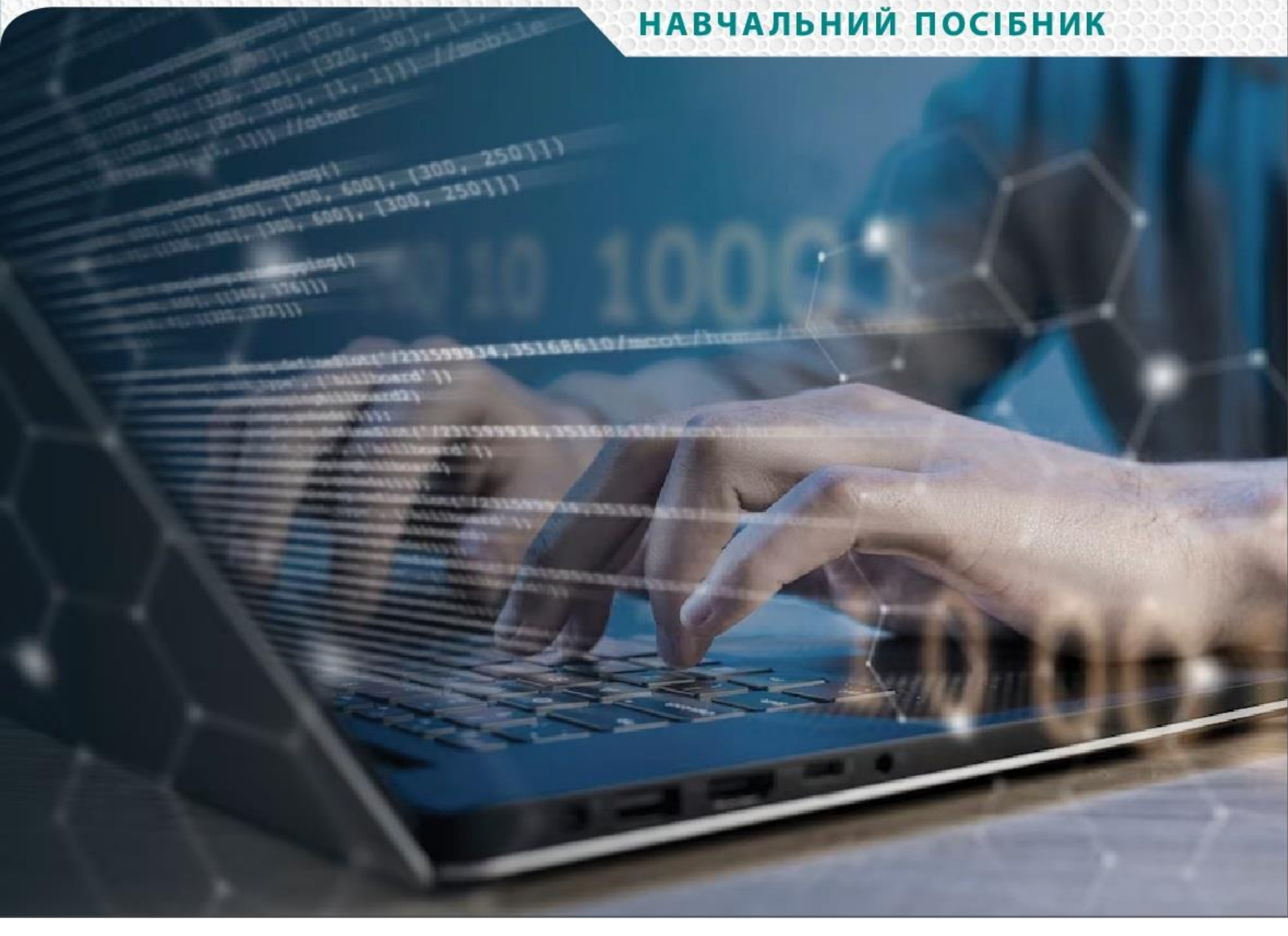
Сергій Ментинський
Ярослав Пелех

ЗБІРНИК ЗАДАЧ З ОСНОВ АЛГОРИТМІЗАЦІЇ ТА ПРОГРАМУВАННЯ

НАВЧАЛЬНИЙ ПОСІБНИК



СЕРГІЙ МЕНТИНСЬКИЙ

ЯРОСЛАВ ПЕЛЕХ

ЗБІРНИК ЗАДАЧ З ОСНОВ АЛГОРИТМІЗАЦІЇ ТА ПРОГРАМУВАННЯ.

НАВЧАЛЬНИЙ ПОСІБНИК
з курсів
«Обчислювальна техніка та програмування», «Інформатика»,
«Основи інформатики і програмування»,
для студентів технічних спеціальностей
для першого (бакалаврського) рівня освіти

Львів 2023

Видавництво ТзОВ «Колір ПРО»

УДК 004.438 (075.8)
М 50
ORCID 0000-0001-8229-429X
ORCID 0000-0002-4339-8093

Рекомендовано до друку Науково-методичною радою

Національного університету «Львівська політехніка»

(протокол № 66 від 14 грудня 2022 р.)

Рецензенти: **Малачівський П. С.** – професор, доктор технічних наук, завідувач відділу математичного моделювання нерівноважних процесів Центру математичного моделювання Інституту прикладних проблем механіки і математики ім. Я.С. Підстригача НАН України, м. Львів

Сеник В. В. – доцент, кандидат технічних наук, завідувач кафедри інформаційного та аналітичного забезпечення діяльності правоохоронних органів Львівського державного університету внутрішніх справ, м. Львів

Гладун В.Р. – доцент, кандидат фізико-математичних наук, доцент кафедри прикладної математики Національного університету «Львівська політехніка», м. Львів

Ментинський С.М., Пелех Я.М.

М 50 Збірник задач з основ алгоритмізації та програмування. Навчальний посібник з курсів «Обчислювальна техніка та програмування», «Інформатика», «Основи інформатики і програмування», для студентів технічних спеціальностей для першого (бакалаврського) рівня освіти / С. М. Ментинський, Я. М. Пелех. – Львів: Галицька Видавнича Спілка, 2023. – 320 с.

ISBN 978-966-2501-91-9

Збірник містить 600 задач з основних розділів програмування, що викладаються у межах традиційних курсів інформатики для технічних спеціальностей у вищих навчальних закладах.

Задачі супроводжуються прикладами програм на різних алгоритмічних мовах (Pascal, C++, Java, Python і ін.) з детальним поясненням алгоритму їх розв’язування.

Для студентів молодших курсів університетів і викладачів, що проводять заняття з основ інформатики та програмування.

УДК 004.438(075.8)

ISBN 978-966-2501-91-9

ЗМІСТ

Передмова

Вивчення основ алгоритмізації та програмування у фундаментальних курсах інформатики для технічних спеціальностей має на меті насамперед виховання у майбутнього інженера “звички до алгоритмічного мислення” при прийнятті рішень та проектуванні технологічних процесів, тому є важливим фактором у формуванні його фахових компетентностей. Викладання цього розділу у вищій школі часто супроводжується труднощами, пов’язаними з різним рівнем базової підготовки студентів з шкільного курсу інформатики та вузькими часовими рамками, відведеними для його вивчення навчальними планами та програмами. З огляду на це, важливим є оптимальний добір навчального матеріалу для практичної та самостійної роботи студента.

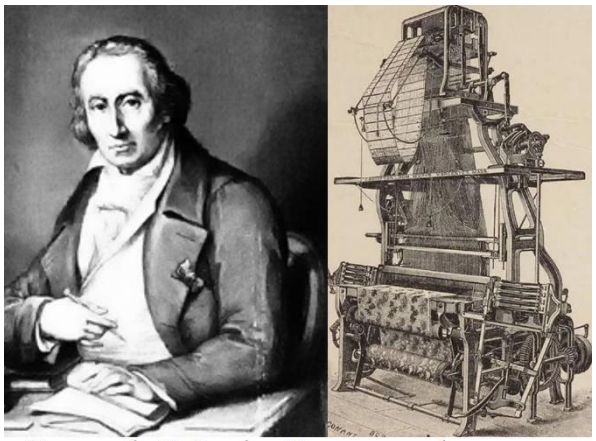
Формат пропонованого навчального посібника у вигляді збірника задач з прикладами програм на різних мовах програмування та детальним поясненням алгоритму розв’язування є спробою акцентувати увагу студента на самій задачі, її формулюванні та процесі її розв’язування виконавцем алгоритму. Такий підхід покликаний власне до формування згаданого вище “алгоритмічного мислення”, незалежно від вибору технології та мови програмування.

Пропонований збірник задач призначений для постановки та проведення лабораторних робіт з розділу основ алгоритмізації та програмування у програмах фундаментальних курсів інформатики для студентів різних технічних спеціальностей. Він містить групи однотипних задач, які можна використовувати для формування варіантів завдань для практичних, лабораторних чи розрахункових робіт. Кожна група задач супроводжується детальним аналізом процесу розв’язування однієї з задач цього типу, описом алгоритму та прикладами програм на декількох мовах програмування (Pascal, C++, Java, Python та ін.).

Посібник можна використовувати також для самостійного поглиблення знань з основ алгоритмізації та програмування, зокрема, на основі порівняння реалізації базових алгоритмічних елементів у різних технологіях програмування.

ВСТУП. Огляд алгоритмічних мов програмування.

Заглиблюючись в історію науково-технічного прогресу можемо зустріти багато згадок про вдалі (чи не надто вдалі) спроби «програмного» керування технологічними процесами, або програмного вирішення технічних задач. Зокрема, з історією виникнення програмування часто пов'язують розробку французького винахідника Жозефа Марі Жаккара, який у 1804 році створив ткацький верстат, в якому візерунок на килимі формувався за допомогою спеціальної «програми». Для тканння килимів вн використав перфоровану (з отворами) стрічку, на якій за допомогою отворів вказував верстату які нитки слід підняти, а які опустити для отримання потрібного візерунка. Сама ідея, як свідчать історики належить іншому французу - Жаку де Вокансону, який застосував її в своєму ткацькому верстаті на пів століття раніше. Не вдаючись до історичних подробиць зазначимо лише, що ця ідея набула значної популярності у середині XX століття як засіб введення даних та коду програм в електронно-обчислювальних машинах перших поколінь.



Joseph Marie Jacquard

Аналізуючи суть ідеї французьких ткачів можна зауважити, що робота їх ткацького верстата керується послідовністю команд, заданих в двійковій (отвір/відсутність отвору) формі. Цей підхід, вже в XX-му столітті, був закріплений в теорії комп'ютерної техніки колективом лабораторії під керівництвом американського математика угорського походження Джона фон Неймана, яка на замовлення уряду США розробляла теоретичні положення функціонування електронно-обчислювальної техніки напередодні створення перших ЕОМ. Серед положень Фон Неймана зустрічаємо таке: «роботу машини контролює програма, що складається з набору команд, які виконуються послідовно одна за одною» та «дані та команди мають бути представлені двійковим (бінарним) числовим кодом».

John von Neumann

Програмне керування перших комп'ютерів (***електронно-обчислювальних машин***, що були розроблені в 40-х роках XX-го століття) і справді було дуже схоже на програмування ткацького верстата Жаккара, – програма та дані для її роботи в двійковому коді наносилася за допомогою отворів на перфокарти та зчитувалася спеціальними пристроями. Зрозуміло, що такий

спосіб функціонування обчислювальної техніки був не надто зручним для розв'язування великих обчислювальних задач, насамперед, з огляду на складність процесу написання самої програми. І вже наприкінці першого десятиліття існування ЕОМ почали з'являтися перші мови програмування. Так для програмування роботи створеної у 1949 році під керівництвом Моріса Вілкса ЕОМ EDSAC розробники використали систему позначень, в якій кожну машинну команду записували однією великою літерою. Така схема позначень разом з використаною в EDSAC бібліотекою підпрограм, розташованих за певною адресою в пам'яті отримала назву «збиральна система» (***assembly system***) та започаткувала сімейство машинно-орієнтованих мов програмування під назвою ***Assembler***.


Sir Maurice V. Wilkes

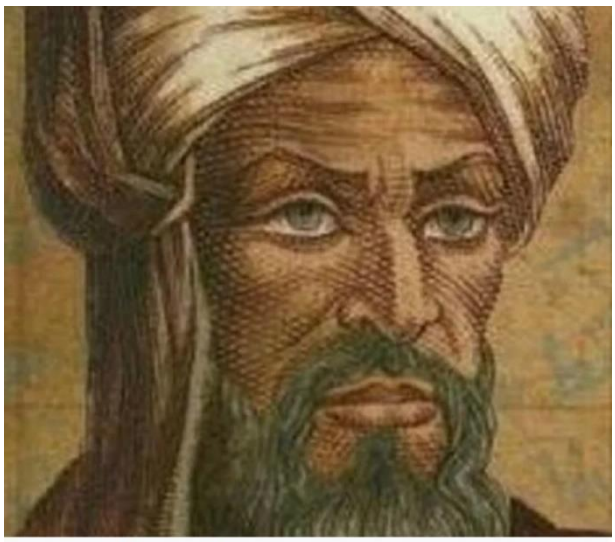
Мови програмування дозволили записувати послідовність команд програми вже не у вигляді отворів на перфокарті, а принаймні, у формі символьних позначень, як в assembly system. ***Мова програмування*** – штучна мова для запису команд, які виконує машина, зазвичай, електронно-обчислювальна (ЕОМ), тобто комп'ютер. Застосування мов програмування аж ніяк не суперечить принципу Фон Неймана щодо двійкового коду: створення програми переходить на вищий рівень абстракції, але її виконання на комп'ютері і далі потребує перекладу програми в умовні «отвори на перфокарті». Для цього використовують спеціальні програми – ***транслятори***. ***Транслятори*** розробляються і вдосконалюються разом з ***мовами програмування*** і бувають двох типів: ***інтерпретатори*** і ***компілятори***.

Інтерпретатор транслює команди програми, читаючи і перекладаючи їх послідовно одна за одною. Команда перекладається на мову комп'ютера та виконується, у випадку її успішного виконання інтерпретатор переходить до опрацювання наступної команди. Сам процес трансляції коду програми інтерпретатором називаємо ***інтерпретацією***. До прикладу, команди сценаріїв JavaScript інтерпретуються і виконуються і процесі завантаження вебсторінки, на якій вони записані. Інтерпретатори простіші у виготовленні та використанні, проте виконати програму на інтерпретованій мові програмування у випадку відсутності чи вимкнення інтерпретатора не вдасться.

Процес трансляції, при якому код усієї програми відразу транслюється в окремий виконавчий модуль називаємо ***компіляцією***. Безперечною перевагою компіляторів є те, що готовий скомпільований код можна багаторазово виконувати навіть за відсутності ***компілятора***. Компіляцію застосовують для розробки програмного забезпечення, що може працювати окремо від середовища його розробки, більшість програм нашого ПК написані, скомпільовані і надані нам для використання вже в готовому вигляді. Щоправда, розробка ***компіля-***

торів є складнішою через потребу додаткового аналізу програмного коду на наявність помилок.

Делегувавши формування двійкового коду трансляторам, програмісти змогли зосередитися на ***алгоритмах*** розв'язування задач, тобто на самій послідовності команд програми. Назва "алгоритм" походить від латинізованого відтворення арабського імені узбецького математика Аль-Хорезмі, який жив наприкінці VIII - на початку IX ст. Він першим сформулював правила, що дозволяють систематично складати і та розв'язувати квадратні рівняння. Під ***алгоритмом*** розуміють набір правил, що однозначно розкривають зміст скінченної послідовності операцій, виконання якої забезпечує успішне розв'язування певного класу задач. Формулювання ***алгоритму*** розв'язування задач є одним з початкових етапів складання програми.

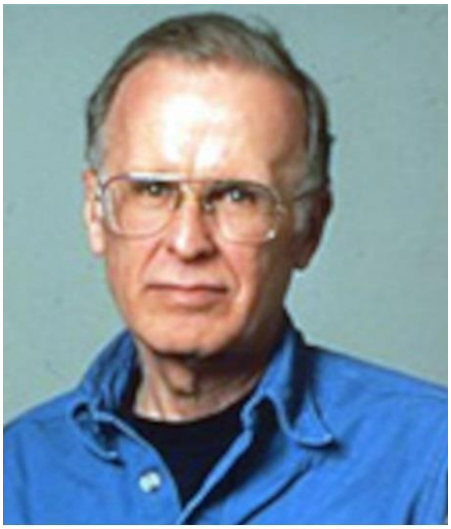

محمد بن موسى الخوارزمي
(Мухамед ібн Муса Аль-Хорезмі)

Важливим чинником, що впливає на формулювання та використання алгоритму є його ***виконавець*** – пристрій або особа, що розв'язуватиме поставлену задачу, використовуючи сформульовану послідовність інструкцій. Алгоритми при цьому повинні відповідати певним вимогам, які ще називають ***властивостями алгоритмів***:

- ✓ **результативність** – *виконання послідовності команд алгоритму має забезпечувати отримання коректного розв'язку задачі;*
- ✓ **формальність** – *інструкції алгоритму можна записувати формально у вигляді деякої обумовленої системи знаків та символів;*
- ✓ **дискретність** – *кожен крок послідовності має бути сформульований у вигляді окремої інструкції зі своїм частковим результатом;*
- ✓ **визначеність** – *кожна команда має бути «зрозумілою» і однозначно трактуватися виконавцем алгоритму;*
- ✓ **масовість** – *алгоритм має бути придатний до розв'язування типової задачі при будь-яких значеннях вхідних даних.*

Використання ***компіляторів*** та ***інтерпретаторів*** дало програмуванню свободу вибору системи знаків та позначень для запису програм. Така свобода відразу поставила розробників нових мов програмування перед дилемою: на що орієнтуватися в формуванні синтаксису мови, – на особливості функціонування ***обчислювальної машини***, чи на обчислювальні конструкції, з яких складаються ***алгоритми*** розв'язування задач. Алгоритми, як свідчить історія, перемогли, а мови програмування почали поділяти на машинні, машинно-орієнтовані, та алгоритмічні (мови програмування високого рівня). Алгоритмічні мови програмування використовують синтаксичні конструкції, що міс-

тять слова та вирази людської мови і є більш зручними для написання коду людиною.


John Backus

З машинно-орієнтованих мов до наших часів розвивається і застосовується для програмування різних технічних пристроїв згаданий вище ***Assembler***. Першою ж алгоритмічною мовою високого рівня став ***Fortran*** (скорочення Formula Translation) презентований у 1957 році IBM та розроблений групою інженерів під керівництвом 30-річного американського математика Джона Бекуса.

У 60-х роках XX століття в Європі активно розроблялася та використовувалася мова програмування ***ALGOL*** (скорочення від Algorythmic Language). Мова була створена міжнародним комітетом, до складу якого входили європейські та американські вчені, зокрема, Джон Бекус, Пітер Наур, Ніклаус Вірт. Той же Ніклаус Вірт у 1970 розробив мову програмування для вивчення процедурного програмування, яку назвав на честь французького фізика й математика Блеза Паскаля. Задум автора був вдалим, – ***Pascal*** вже пів століття не покидає навчальні програми шкіл та технічних вишів. Враховуючи таку особливість цієї мови, використовуватимемо її для програмування алгоритмів розв'язування задач в усіх прикладах до збірника.


Niklaus Emil Wirth

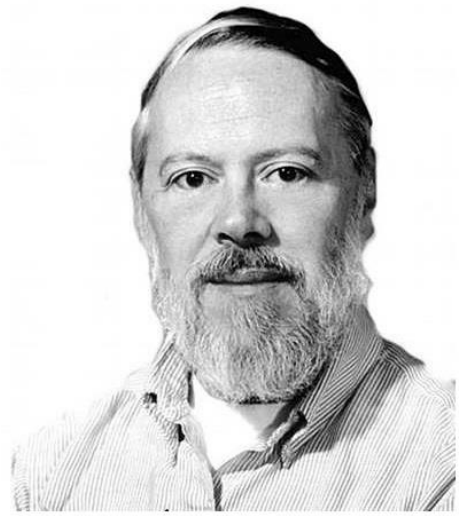
Інтенсивно розвивалося програмування у цей період, разом з обчислювальною тех-нікою, і на теренах України. Насамперед, у Києві. Найвагоміший внесок в розвиток українських обчислювальних технологій належить науковцям створеної у 1957 році лабораторії обчислювальної математики та техніки Інституту математики АН УРСР під керівництвом академіка Віктора Глушкова, яка з 1962 року стала Інститутом кібернетики АН УРСР. Сьогодні це Інститут кібернетики ім. В. М. Глушкова НАН України. Науковий доробок співробітників інституту, що стосувався не тільки обчислювальної техніки та програмування, але й теоретичних досліджень у царині штучного інтелекту, ліг в основу різних технологій програмування світового масштабу.


Віктор Михайлович Глушков

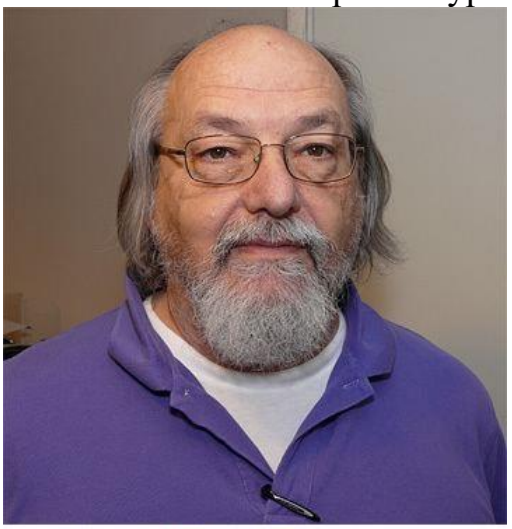
Серед відомих співробітників Інституту кібернетики зустрічаємо постать Катерини Логвинівни Ющенко – першого в СРСР доктора фізико-математичних наук в галузі інформаційних технологій. Українська науковиця є розробником ***адресної мови програмування***, що за часом виникнення (1955 р.) випереджує ***Fortran.*** Ця формальна мова програмування дещо пізніше була реалізована у багатьох моделях ЕОМ, які виготовлялися на території СРСР та постачалися в країни східної Європи (ЕОМ «Київ» та «Дніпро», на комп'ютерах М-20, «Урал» та «Мінськ» і ін.).

Катерина Логвинівна Ющенко

Цікаво в ракурсі поділу мов програмування на машинно-залежні та алгоритмічні виглядає історія виникнення мови програмування ***C***, розробленої на початку 70-х років минулого століття в стінах американської дослідницької компанії Bell Labs. Програмісти цієї компанії Денніс Рітчі та Кеннет Томпсон насправді займалися розробкою нової операційної системи для свого комп'ютера на асемблері. Для зручності винахідники вирішили згрупувати окремі команди асемблера в більші команди (***макрокоманди***) і отримали нову мову програмування високого рівня з властивостями машинно-орієнтованої розробки. Це дозволило поєднати переваги ***машинно-орієнтованих*** мов в плані максимального використання можливостей конкретної обчислювальної архітектури з перевагами ***алгоритмічних мов програмування***. Ця особливість мови ***C*** зробила її у свій час найпопулярнішим засобом розробки програмного забезпечення, як системного, так і прикладного.



Dennis Ritchie



Kenneth Lane Thompson

Алгоритмічна мова ***C++*** виникла на основі мови ***C*** шляхом додавання до неї елементів ***об'єктно-орієнтованого програмування*** (***ООП***). Про це свідчить навіть її початкова назва "Сі з класами" (***клас*** – одне з фундаментальних понять в ООП), яку дав їй її засновник Б'ярне Страуструп. Увібравши разом з базовим функціоналом ***C*** усі переваги своєї попередниці ***C++*** перейняла і її провідну роль в розробці програмного забезпечення на наступному етапі розвитку технологій програмування. Цьому

етапу притаманна потреба в ООП для оптимального структурування значного обсягу програмного коду, що власне і спонукало перехід розробників від ***C*** до ***C++***.


Bjarne Stroustrup

Велика популярність ***C/C++*** серед розробників стала головним чинником того, що багато новітніх мов програмування, які виникли в період інтенсивного розвитку мережевих технологій наприкінці XX ст., такі як ***Java***, ***PHP***, ***JavaScript***, ***C#***, отримали С-подібний синтаксис алгоритмічних конструкцій. Тому сьогодні більшість майбутніх програмістів розпочинають своє навчання саме з вивчення програмування на ***C/C++***. Усі приклади в нашому посібнику також супроводжуються кодом на ***C++*** з детальним його поясненням.

Об'єктно-орієнтоване програмування використовується більшістю сучасних технологій в якості ключової парадигми. Попри те, що задачі збірника не потребують для свого розв'язування об'єктно-орієнтованого підходу, для повноти нашого екскурсу в програмування алгоритми розв'язування прикладів подаються «в перекладі» на об'єктно-орієнтовану мову (здебільшого ***Java***, подекуди можна зустріти також код на ***C#***).


Patrick Naughton

Окрім ефективного впровадження ***об'єктно-орієнтованого підходу*** технології ***Java*** та ***.net (C#)*** принесли в розробку програмного забезпечення ще одну важливу та актуальну сьогодні концепцію – ***крос-платформність*** програм. Вона полягає в можливості програми однаково працювати на пристроях з різними операційними системами та різною конфігурацією апаратного забезпечення.

Зокрема, створення ***Java*** розпочинається з історії про невдоволення одного з провідних іженерів компанії ***Sun Microsystems*** Патріка Ноутона потребою в перекомпіляції програмних кодів ***C++***-програм для різних пристроїв, над програмним керуванням якими в той час працювала компанія.

Так виникає та впроваджується ідея використання проміжного коду (байт-коду) та виконання програм під керуванням віртуальної машини (***Java Virtual Machine*** – ***JVM***). Спочатку ці підходи знайшли втілення у випущеній всередині 90-х років минулого століття компанією Sun Microsystems технології ***Hot Java***, загальновідомій під назвою ***Java***. Вже на початку нового тисячоліття ці ж ідеї у поєднанні з одночасною підтримкою кількох мов

програмування (в т.ч. ***C++***, ***C#*** та ***Visual Basic***) лягли в основу нової технології програмування ***.NET Framework*** від компанії ***Microsoft***.

Компіляція програмного коду перетворює його в програму, що виконується комп'ютером, більшість програм, встановлених на нашому ПК складаються із раніше скомпільованих бібліотек та компонентів. Інтерпретовані програми, як вже згадувалося раніше, можуть виконуватися лише в системах, які мають встановлений відповідний ***інтерпретатор***. На перший погляд, ця особливість мала б робити ***інтерпретовані*** програми менш популярними та продуктивними за ***компільовані***. Про те, що така думка є помилковою свідчить понад 30-річна історія розробленої нідерландським програмістом Гвідо ван Россумом мови програмування ***Python***. Саме ця мова обрана авторами задачника для прикладів коду програм, що транслюються за допомогою ***інтерпретації***. Для цієї технології, так само як для Java чи .Net характерна крос-платформність, яка полягає в тому, що програми на ***Python*** працюватимуть всюди, де встановлено інтерпретатор мови. Зокрема, інтерпретатор ***Python*** вбудовано в сучасні випуски операційної системи ***Linux***.


Guido van Rossum

Кожна програма, – від програм зі студентських лабораторних робіт і до повнофункціональних десктопних додатків чи мережевих сервісів в своєму «житті» проходить низку технологічних етапів. Технологія створення програмних продуктів і рішень, зазвичай, включає:

- *постановку задачі;*
- *побудову математичної моделі;*
- *складання алгоритму розв'язування;*
- *кодування (переклад алгоритму на мову програмування);*
- *відлагодження та тестування програми;*
- *експлуатацію програми та її супровід.*

Не вдаючись в подробиці діяльності розробника ПЗ на кожному з перелічених етапів, зазначимо лише, що подібна схема буде використовуватися нами в прикладах розв'язування задач збірника. Спочатку ми аналізуватимемо умову задачі для з'ясування яку мету матиме виконання нашої програми, та з якими даними вона працюватиме. Потім встановлюватимемо закономірності та формули для розв'язування задачі, аж тоді переходитимемо до формулювання самого алгоритму. Далі записуватимемо алгоритм на обраній мові програмування. На завершення, де це потрібно, будемо формулювати тестові набори вхідних даних з відомим результатом, щоб перевірити коректність роботи створеної програми.

Розділ 1. Програмування алгоритмів лінійної структури

Створення та виконання простої програми. Лінійна алгоритмічна конструкція.

1. Задано два кола зі спільним центром і радіусами R_1 і R_2 ($R_1 > R_2$). Знайти площі цих кіл S_1 та S_2, а також площу S_3 кільця, зовнішній радіус якого дорівнює R_1, а внутрішній радіус дорівнює R_2. Як значення π використовувати 3.14.

2. Задано координати двох протилежних вершин прямокутника: (x_1, y_1), (x_2, y_2). Сторони прямокутника паралельні осям координат. Знайти периметр і площу даного прямокутника.

3. Знайти відстань між двома точками із заданими координатами (x_1, y_1) і (x_2, y_2) на площині.

4. Задано координати трьох вершин трикутника: (x_1, y_1), (x_2, y_2), (x_3, y_3). Знайти його периметр, використовуючи формулу для відстані між двома точками на площині.

5. Задано сторони прямокутника a і b. Знайти його площу S і периметр P.

6. Задано довжини ребер A, B, C прямокутного паралелепіпеда. Знайти його об’єм та площу поверхні.

7. Задано число A. Обчислити A^{15}, використовуючи дві допоміжні змінні і п'ять операцій множення. Для цього послідовно знаходити A^2, A^3, A^5, A^{10}, A^{15}. Вивести всі знайдені степені числа A.

8. Задано значення температури T в градусах Фаренгейта. Визначити значення цієї ж температури в градусах Цельсія. Температура за Цельсієм TC і температура TF за Фаренгейтом зв'язані наступним співвідношенням: TC = (TF - 32) • 5/9.

9. Задано значення температури T в градусах Цельсія. Визначити значення цієї ж температури в градусах Фаренгейта. Температура за Цельсієм TC і температура за Фаренгейтом TF пов'язані наступним співвідношенням: TC = (TF - 32) • 5/9.

10. Відомо, що X кг шоколадних цукерок коштує A гривень, а Y кг ірисок - B гривень. Визначити, скільки коштує 1 кг шоколадних цукерок, 1 кг ірисок, а також у скільки разів шоколадні цукерки дорожчі ірисок.

11. Задано два ненульових числа. Знайти суму, різницю, добуток і частку їх квадратів.

12. Задано два ненульових числа. Знайти суму, різницю, добуток і частку їх модулів.

13. Задано катети прямокутного трикутника a і b. Знайти його гіпотенузу c і периметр P.

14. Швидкість першого автомобіля V_1 км / год, другого - V_2 км / год, відстань між ними S км. Визначити відстань між ними через T годин, якщо вони рухаються в одному напрямку.

15. Швидкість першого автомобіля V_1 км / год, другого - V_2 км / год, відстань між ними S км. Визначити відстань між ними через T годин, якщо автомобілі рухаються назустріч один одному.

16. Задано довжину кола L. Визначити його радіус R та площу S круга, який це коло обмежує, враховуючи, що $L = 2\pi R$, $S = \pi R^2$.

17. Задано площу круга S. Визначити його радіус R та довжину кола L, яке його обмежує, враховуючи, що $S = \pi R^2$, $L = 2\pi R$.

18. Знайти потенціальну енергію тіла масою m, піднятого на висоту h над поверхнею Землі ($E = mgh$).

19. Знайти об'єм V та площу бічної поверхні $S_б$ циліндра, якщо відомо радіус R його основи та висоту H.

20. Знайти об'єм V та площу бічної поверхні $S_б$ конуса, якщо відомо радіус R його основи та висоту H.

21. Тіло починає рухатися без початкової швидкості з прискоренням a. Визначити швидкість v, яку розвине тіло за t сек., та шлях S, який воно пройде за цей час.

22. За заданим радіусом кулі обчислити її об’єм та площу поверхні за формулами $V = \frac{4}{3}\pi R^3$, $S = 4\pi R^2$.

23. Задано об’єм кулі. Визначити її радіус та площу поверхні, враховуючи, що $V = \frac{4}{3}\pi R^3$, $S = 4\pi R^2$.

24. Задана площа ділянки в гектарах. Вивести площу ділянки спочатку в метрах квадратних, а потім кілометрах квадратних.

25. Задана маса вантажу в центнерах. Вивести цю масу спочатку в кілограмах, а потім в тонах.

26. Задана маса вантажу в фунтах. Вивести цю масу спочатку в кілограмах, а потім в центнерах, якщо 1 фунт = 0,45359237 кілограма.

27. Водій на автозаправці залив у бак X літрів пального. Визначити суму до сплати, якщо відома ціна C за літр пального та відсоток P фіксованої (не залежить від кількості пального) знижки.

28. Виріб збирають з трьох вузлів собівартістю A, B та C грн. відповідно. На оплату складання одного виробу затрачають X грн. Визначити ціну виробу, якщо від його продажу планується отримання прибутку P%.

29. Задано довжини сторін трикутника $a,\ b$ та с. Обчислити довжини його медіан $m_a = \frac{\sqrt{2b^2+2c^2-a^2}}{2}$, $m_b = \frac{\sqrt{2a^2+2c^2-b^2}}{2}$, $m_c = \frac{\sqrt{2a^2+2b^2-c^2}}{2}$.

30. Металічний куб з довжиною ребра a переплавили в кулю. Визначити радіус R кулі, враховуючи, що об'єм кулі $V = \frac{4}{3}\pi R^3$, а об'єм куба $V = a^3$.

В задачах 31 - 60 запрограмувати обчислення виразу за заданими формулами для введених користувачем вхідних даних:

31. $y = |v^2 - u^2|^{2z+3}$, де $u = \sin t + \cos t$, $v = |\sin t|^{\cos t}$;

32. $y = \frac{v^{\sin p} - v^{\cos p}}{e^t}$, де $p = x^3 + 3x^2 + 18$, $v = \sin t + x$, $x > 1$;

33. $y = \frac{e^{\sin p} - e^{\cos p}}{ln|z|}$, де $p = |x^3 tgt|$, $z = \frac{x+t}{xt}$;

34. $z = ln(v^2 + u^2)$, де $u = \sin^2 t + xt$, $v = \frac{5x^3}{t^2+2t}$;

35. $z = tg(p^3 + 12q)$, де $p = log_5 x + log_7 y$, $q = 5x^y$, $x, y > 0$;

36. $z = lg(p^2 + 12|q|)$, де $p = lg\, x - y^x\, ln|x + e^{-4y}|$, $q = (x+1)^{y+x}$, $x, y > 0$;

37. $z = cos(p^{q+2} + 2pq^p)$, де $q = ln\, x + |ctg\, y|$, $p = y^x + x^y$, $x. y > 1$;

38. $p = lg\, x - y^q\, ln\left|e^{xq^2} v + e^{-qv}\right|$, де $v = lg\, x + y$, $q = y^3 - x^5$, $x, y > 0$;

39. $p = \frac{ln\left|e^{sy^2} x + e^{-ty}\right|}{tg\, x - y^t}$, де $x = lg\, t + s^2$, $y = e^{2t} - e^{2s}$, $t > s$;

40. $p = \frac{\left|ye^{x^2} + e^{-3y}\right|}{log_3(t+s)}$, де у$x = \cos t + \sin s$, $y = s^{3t} - t^{5s}$, $t, s > 0$;

41. $v = tg(p^3 + 15q)$, де $p = lg\, x - y^x\, ln|x + e^{-4y}|$, $q = 5x^y$, $x > 1$;

42. $v = lg(p^{2x} + 12|q|)$, де $p = log_5 x + log_7 y$, $q = (x+1)^{y+x}$, $x, y > 1$;

43. $z = |v^{2t} - u^{2t}|^{2x+3}$, де $u = \sin t + \cos t$, $v = \sin t + x$;

44. $z = \frac{p^{\sin p} - v^{\cos p}}{e^t}$, де $p = x^3 + 3x^2 + 18$, $v = |\sin t|^{\cos t}$, $x > 0$;

45. $y = \frac{e^{\sin p} - e^{\cos p}}{log_3 z}$, де $p = x^2 \sin t$, $z = \left|\frac{x-t}{xt}\right|$;

46. h$a = lg(p^3 + 12|pq|)$, де $p = lg\, x - y^x\, ln(x + e^y)$, $q = (x+1)^{y+x}$, $x > 1$, $y > 0$;

47. $b = tg(p^{2x} + 12|q|)$, де $p = log_5 x + log_7 y$, $q = (x+1)^{y+x}$, $x, y > 1$;

48. $p = lg\, x - y^q\, ln|e^x v + e^{-qv}|$, де $v = tg\, x + y$, $q = y + x^5$, $x, y > 0$;

49. $z = \frac{p^{\sin t} - v^{\cos t}}{e^{p+v}}$, де $p = x^3 + 3t^2$, $v = |\sin x|^{\cos t}$, $x > 0$;

50. $y = (v^2 + u^2)^{2u+3v}$, де $u = \sin t - \cos t$, $v = sin(z^2 + t^2)$;

51. $z = cos(p^{q+2} + 2p - q^p)$, де $p = |x\ ctgy|$, $q = |y^x - x^y|$, $x, y > 0$;

52. $q = \frac{v^{sin\,p} - v^{cos\,p}}{v^{cos\,t}}$, де $yp = x^3 + 3x^2 + 18$, $v = sin\,t + x$, $x > 1$;

53. $z = ln(v^2 + u^2) + cos\,u$, де $u = sin^2\,t + xt$, $v = \frac{5x^3}{x^2+2x}$;

54. $y = tg(p^{3x} + qy)$, де $p = log_5\,x + log_7\,y$, $q = 5x^y$, $x, y > 1$;

55. $v = lg|p^3 - 15q|$, де $p = lg\,x - y^x\,ln|-4y|$ $q = 5x^3 + 4y^2$, $x, y > 0$;

56. $y = |v^{2t} - u^{2t}|^{2u+3v}$, де $u = sin\,t - cos\,t$, $v = sin\,t + x$;

57. $y = \frac{p^{sin\,x} - p^{cos\,t}}{ln|z|}$, де $p = |x^3 tgt|$, $z = \frac{x+t}{xt}$;

58. $y = \frac{e^{sin\,p} - e^{cos\,p}}{log_3\,z}$, де $p = x + sin^2\,t$, $z = \left|\frac{xt}{x-t}\right|$;

59. $v = \frac{lg\left|e^{sy^2}x + e^{-ty}\right|}{x - y^4}$, де $x = lg\,t + s^2$, $y = e^{2t} - e^{2s}$, $t > 0$;

60. $p = \frac{\left|e^{u^2} + e^{-3u}\right|}{log_3\,v}$, де $u = cos\,x + sin\,y$, $v = y^{3x} + x^{5y}$, $x, y > 0$.

61. Користувач вводить чотирицифрове натуральне число. Знайти суму та добуток його цифр.

62. Задано трицифрове натуральне число. Вивести число, отримане перестановкою його цифр у зворотному порядку.

63. Задано трицифрове ціле число. У ньому закреслили першу цифру ліворуч та дописали її ж праворуч. Вивести отримане число.

64. Задано двоцифрове число. Вивести суму квадратів та добуток його цифр.

65. Задано трицифрове ціле число. Вивести число, отримане при перестановці цифр сотень та десятків заданого числа (наприклад, з отриманого числа 376 утвориться число 736).

66. Задано чотирицифрове натуральне число. Знайти різницю добутку його крайніх цифр та добутку внутрішніх цифр.

67. Задано трицифрове ціле число. Вивести число, отримане перестановкою цифр десятків та одиниць вихідного числа (наприклад, 123 перейде до 132).

68. Задано ціле число, більше за 999. Використовуючи лише одну операцію ділення та одну операцію отримання остачі від ділення, знайти цифру, що задає кількість сотень цього числа.

69. Задано трицифрове ціле число. Вивести спочатку його останню цифру (одиниці), а потім його середню цифру (десятки).

70. Задано ціле число, що означає відстань L у міліметрах. Використовуючи операції цілочисельного ділення та остачі від ділення, записати відстань L у метрах, сантиметрах та міліметрах.

71. Масу M у кілограмах задано цілим числом, більшим за 1000. Використовуючи ділення націло та остачу від ділення, записати масу M в тонах, центнерах та кілограмах (наприклад 12345 кг = 12 тон 3 центнери і 45 кг).

72. Наведено розмір файлу в байтах. Визначити кількість повних кілобайтів, які займає даний файл (1 кілобайт = 1024 байти).

73. Задано трицифрове ціле число. У ньому закреслили першу праворуч цифру та дописали її ліворуч. Вивести отримане число.

74. Задано двоцифрове число. Знайти суму та добуток його цифр.

75. Задано двоцифрове число. Вивести число, отримане перестановкою його цифр.

76. Дні тижня пронумеровані таким чином: 0 – неділя, 1 – понеділок, 2 – вівторок, …, 6 – субота. Ціле число K з відрізка [1;365] задає порядковий номер дня в році. Визначити номер дня тижня для K-го дня року, якщо відомо, що 1 січня цього року був понеділок.

77. Дні тижня пронумеровані таким чином: 0 – неділя, 1 – понеділок, 2 – вівторок, …, 6 – субота. Ціле число K з відрізка [1;365] задає порядковий номер дня в році. Визначити номер дня тижня для K-го дня року, якщо відомо, що 1 січня цього року була субота.

78. Дні тижня пронумеровані таким чином: 0 – неділя, 1 – понеділок, 2 – вівторок, …, 6 – субота. Ціле число K з відрізка [1;365] задає порядковий номер дня в році. Визначити номер дня тижня для K-го дня року, якщо відомо, що 13 січня цього року була п’ятниця.

79. Дні тижня пронумеровані таким чином: 1 – понеділок, 2 – вівторок, …, 6 – субота, 7 – неділя. Ціле число K з відрізка [1;365] задає порядковий номер дня в році. Визначити номер дня тижня для K-го дня року, якщо відомо, що 1 січня цього року була середа.

80. Дні тижня пронумеровані таким чином: 1 – понеділок, 2 – вівторок, …, 6 – субота, 7 – неділя. Ціле число K з відрізка [1;365] задає порядковий номер дня в році. Визначити номер дня тижня для K-го дня року, якщо відомо, що 1 січня цього року був вівторок.

81. Задано ціле число, більше за 999. Використовуючи лише одну операцію ділення та одну операцію отримання остачі від ділення, знайти цифру, що задає кількість десятків цього числа.

82. Задано цілі додатні числа A, B, C. В прямокутнику розміру A × B розміщено максимально можливу кількість квадратів зі стороною C (без накладень). Знайти кількість квадратів, розміщених на прямокутнику, а також площу незайнятої частини прямокутника.

83. Користувач вводить два двоцифрові числа. Утворити з них чотирицифрове число за правилом: 1 цифра = 1 цифра першого числа, 2 цифра = 1 цифра другого числа, 3 цифра = 2 цифра першого числа, 4 цифра = 2 цифра другого числа.

84. Задано п’ятицифрове натуральне число. Визначити добуток його першої, третьої та останньої цифр.

85. Задано шестицифрове натуральне число. Визначити суму та добуток його першої та останньої цифри.

86. Користувач вводить два двоцифрові числа. Утворити з них чотирицифрове число шляхом дописування після першого числа цифр другого числа в зворотному порядку.

87. Користувач вводить два трицифрові числа. Утворити з них чотирицифрове число за правилом: 1 цифра = остання цифра першого числа, 2 цифра = остання цифра другого числа, 3 цифра = 1 цифра першого числа, 4 цифра = 1 цифра другого числа.

88. У трицифровому числі k закреслили першу цифру. Якщо отримане число помножити на 10 і добуток додати до першої цифри числа k, то отримаємо число n. За заданим числом n, $1< n <999$ знайти число k.

89. Від трицифрового числа k відняли його останню цифру. Після ділення результату на 10 до частки ліворуч дописали останню цифру числа k та отримали число m. За заданим числом m знайти число k, вважаючи, що число m належить відрізку [100;999], а середня цифра не дорівнює нулю.

90. Задана кількість секунд, що пройшла від початку доби до поточного моменту. Вивести поточний момент часу в годинах, хвилинах та секундах.

Практикум до розділу 1. Програмування алгоритмів лінійної структури

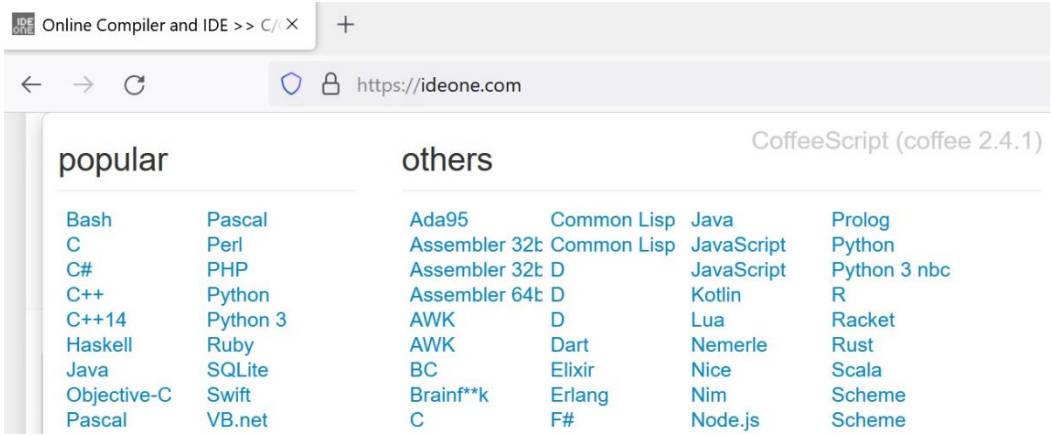
Всі задачі цього розділу передбачають програму, що реалізує просту послідовність дій:

- *на початку роботи програми користувач вводить вхідні дані для розв'язування задачі;*
- *програма обчислює розв'язок задачі за формулами, заданими їй програмістом;*
- *програма повідомляє користувачеві отриманий результат і завершує свою роботу.*

Таку конструкцію алгоритму прийнято називати лінійною. В ній команди алгоритму виконуються послідовно одна за одною від початку до закінчення алгоритму.

В середовищі програмістів поширений такий напівжарт: якщо ти навчився на деякій мові програмування написати програму, що виведе на екрані фразу «Hello World!», то ти вже вмієш програмувати цією мовою. Попри те що ця байка достатньо далека від істини, в більшості навчальних курсів з програмування в якості першої програми традиційно використовується програма «Hello World!». Цінність цієї програми в навчальному процесі безперечна, – вона дозволяє майбутньому програмісту відчути три фундаментальні моменти: які засоби йому необхідні для створення програм, яку структуру матиме код програми на обраній ним мові і які засоби має ця мова для «спілкування із зовнішнім світом». Завдання цього розділу сприяють формуванню трохи більшого числа компетентностей, додаючи до трьох перелічених розуміння механізму вводу та зберігання даних, а також виконання елементарних обчислень і отримання їх результату.

Сучасні засоби розробки програм доволі розмаїті, практично кожна мова програмування має кількадесят інструментів для написання та виконання коду програм. Маючи на меті застосування до розв'язування задач декількох мов програмування, будемо наразі використовувати мультимовні середовища розробки програм. Найпростіше скористатися для цього одним з online-

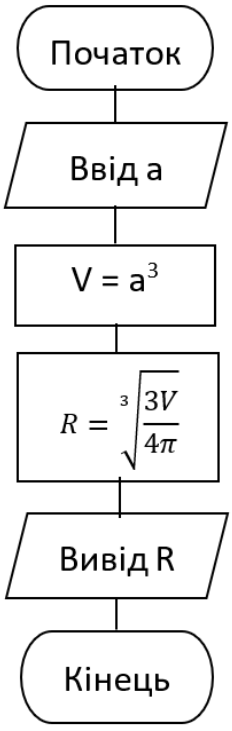


сервісів доступних в мережі ***Internet***, наприклад, ***ideone.com***. Цей ресурс дає можливість обирати для написання та виконання програми одну з кількох десятків популярних мов програмування, а також використовувати готові приклади та ділитися власними програмами з іншими користувачами в мережі.

Перейдемо до розв'язування задач.

30. Металічний куб з довжиною ребра a переплавили в кулю. Визначити радіус R кулі, враховуючи, що об'єм кулі $V = \frac{4}{3}\pi R^3$, а об'єм куба $V = a^3$.

Вхідними даними тут є довжина ребра куба a, її значення програма повинна отримати від користувача напочатку, далі можна знайти об'єм куба за формулою $V = a^3$. Якщо вважати, що при переплавці металу його об'єм не змінюється, то отримане значення V буде об'ємом кулі. А от для знаходження її радіуса формулу, задану в умові, слід записати у вигляді $R = \sqrt[3]{\frac{3V}{4\pi}}$, виразивши радіус R через об'єм V. Ця формула і є формулою, за якою програма обчислюватиме розв'язок задачі.



Більшість технологій програмування має вбудовані засоби отримання значення числа π, але не будемо на них відволікатися, а обмежимося заданням наближеного значення $\pi \approx 3{,}1416$ безпосередньо в коді програми.

Отож, алгоритм розв'язування задачі буде таким:

- *на початку роботи програми користувач вводить число, що задає довжину ребра куба a;*
- *програма знаходить об'єм куба (а, отже, і об'єм кулі) за формулою* $V = a^3$;
- *за отриманим значенням об'єму* V *та заданим в програмі значенням числа* π *програма обчислює величину радіуса кулі за фомулою* $R = \sqrt[3]{\frac{3V}{4\pi}}$;
- *обчислену довжину радіуса кулі програма повідомляє користувачеві і завершує свою роботу.*

Графічне зображення сформульованого вище словесно алгоритму подано у вигляді блок-схеми на малюнку.

В більшості мов програмування базові бібліотеки не мають функції для обчислення кореня кубічного, тому формулу для знаходження R використовуємо у вигляді

$$\mathrm{R} = \sqrt[3]{\frac{3\mathrm{V}}{4\pi}} = \left(\frac{3\mathrm{V}}{4\pi}\right)^{\frac{1}{3}}.$$

Програма на Pascal.

Мова програмування ***Pascal*** належить до класичних алгоритмічних мов і є типовим представником структурної парадигми програмування. Програма на цій мові має чітко регламентовану структуру, що складається з розділів: опису констант, опису змінних, функцій та процедур та блоку операторів, тобто тіла самої програми.

Основні правила синтаксису ***Pascal***-програми:

- ✓ *програма починається зі службового слова* **program**, *за яким вказують її назву;*
- ✓ *всі змінні та константи, що використовуються в програмі повинні бути оголошені у відповідному розділі оголошень, до блоку команд;*
- ✓ *блок команд, які виконує програма, записують після розділів опису, він починається службовим словом* **begin** *і завершується словом* **end** *з крапкою;*
- ✓ *інструкції програми відділяються одна від одної знаком ";" і можуть записуватися як в одному так і в різних рядках;*
- ✓ *коментарі, тобто, текст програми, який не виконується, а використовується лише для пояснень, записують у фігурних дужках;*
- ✓ *Pascal регістро-незалежна мова, тобто команди можна записувати як малими так і великими літерами, а змінні* **x** *та* **X** *взагалі позначатимуть в програмі одну і ту ж величину;*
- ✓ *для виконання обчислень та запису отриманого значення у змінну використовується оператор присвоєння, що позначається знаком ":=" (сам знак «дорівнює» використовується для порівняння даних на рівність).*

Далі подаємо запис алгоритму розв'язання задачі 30 мовою ***Pascal***, пояснення дії інструкцій записано в коментарях. В ***Pascal*** відсутній арифметичний оператор піднесення до степеня, так само як і відповідна функція, тому тут використовуємо формулу $a^x = e^{x \cdot \ln a}$.

```
program task30;

const PI = 3.141593; {Константа, значення числа пі}

var a,v,r3,r:real; {Оголошення змінних – дійсних чисел}

begin

    readln(a); {Зчитуємо введене з клавіатури значення }

    v:=a*a*a; {Обчислюємо об'єм куба}

    r3:= 3*v/(4*PI); {Куб радіуса}

    r:=exp(1/3*ln(r3)); {Корінь кубічний з r3}

    writeln("Радіус кулі ",r) {Вивід результату}

end.
```

Програма на Fortran.

Програми на ***Фортрані*** (скорочення від ***Formula Translator***) досить схожі на паскаль-програми. Ці дві мови мають багато спільного як в часовому вимірі, так і щодо функціональності та реалізованих підходів до програмування. Fortran, здебільшого, використовувався для різноманітних наукових розрахунків. Напрацьовані десятиліттями спеціалізовані бібліотеки забезпечують йому популярність в наукових колах і донині.

Основні правила синтаксису Fortran-програми:

- ✓ *програма починається зі службового слова program, за яким вказують її назву і завершується службовим словом end;*
- ✓ *всі змінні та константи, що використовуються в програмі повинні бути оголошені на початку програми, до першої інструкції виконання;*
- ✓ *для створення іменованих констант використовується службове слово parameter, записане після типу константи та оператор "::";*
- ✓ *інструкції програми відділяються одна від одної переходом на нову стрічку або знаком ";", якщо вони розташовані в одній стрічці;*
- ✓ *коментарі, що використовується для пояснень, записують у рядку, що починається зі знаку оклику;*
- ✓ *для присвоєння змінній значення використовується знак «дорівнює» (для порівняння значень – знак "= =").*

Решта особливостей фортран-програми пропонуємо розглянути в коді розв'язання задачі 30. ***Fortran***, на відміну від ***Pascal***-я, має арифметичний оператор піднесення до степеня, він записується знаком "**". Для збереження даних дійсного типу, як і в ***Pascal***, можна використовувати тип ***real***.

```
program TEST
!Опис констант та змінних
    real, parameter :: PI = 3.141593
    real a,v,r3,r
!Розділ операторів
!Процедура вводу зі стандартного пристрою
    read *,a
!Команди відділяють переходом на !наступну стрічку, або ";"
    v = a*a*a; r3 = 3*v/(4*PI); r = r3**(1.0/3)
!Процедура виводу на стандартний пристрій
!у форматі за замовчуванням
    print *, 'Радіус кулі ',r
end
```

Програма на C++.

Мова програмування ***C*** створена на основі макрокоманд асемблера і центральною її парадигмою процедурне програмування. Програми на ***C*** утворюються з функцій. Мова ***C***++ виникла як об'єктно-орієнтоване продовження мови ***C***. Тобто, ***C***++ підтримує дві основні парадигми: процедрне та об'єктно-орієнтоване програмування. В процедурному ж підході, що застосовується в рамках цього навчального посібника мови ***C*** та ***C***++ здебільшого співпадають.

Процедурний підхід обумовлює наступні концепції синтаксису ***C***++:

- ✓ *програма складається з функцій, функції можуть викликати одна одну, але не допускається вкладення функції у функцію;*
- ✓ *виконання програми починається з головної функції («точки входу») з назвою* **`main`***;*
- ✓ *кожна функція складається з заголовку та тіла, заголовок містить назву функції, перед якою вказується тип результату функції, а після, – список параметрів функції в круглих дужках;*
- ✓ *тіло функції складається з послідовності інструкцій програми, у фігурних дужках, кожна інструкція закінчується знаком "*`;`*", поділ коду на стрічки, взагалі кажучи, до уваги не береться;*
- ✓ *виконання функції завершується командою* **`return`***, після якої вказується значення, що буде повернене в якості результату функції;*
- ✓ *функції можуть не повертати жодного значення, в такому випадку тип результату у заголовку функції задається службовим словом* **`void`***;*
- ✓ *в тілі функції, яка не повертає значення (***`void`***) команда return не обов'язкова, за її відсутності виконання функції завершиться після досягнення закриваючої фігурної дужки, що обмежує тіло функції;*
- ✓ *змінні та константи, що використовуються в тілі функції можуть бути оголошені будь-де, але до їх першого використання для присвоєння чи вводу та виводу даних;*
- ✓ *для створення іменованих констант використовується службове слово* **`const`***, записане перед типом константи;*
- ✓ *коментарі, тобто, пояснення до програми, записують у рядку, починаючи з пари знаків «слеш» ("*`//`*"), або між парами символів "*`/*`*" та "*`*/`*";*
- ✓ *для присвоєння змінній значення використовується знак «дорівнює» (а для порівняння значень – знак "*`= =`*").*

Операція піднесення до степеня в ***C/C***++ відсутня, але для обчислення a^x можна скористатися бібліотечною функцією ***`pow(a, x)`***.

Сучасні стандарти ***C***++ використовують простори імен (***`namespace`***), зокрема, більшість бібліотечних класів, об'єктів та функцій «занурені» в простір імен ***`std`***. Тому для звертання до стандартних потоків вводу-виводу ***`cin`*** та ***`cout`*** в коді нижче використано так званий «оператор дозволу області видимості» "***`::`***" для цього простору імен. Програма-розв'язок задачі 30 мовою C++:

```
//заголовковий файл з бібліотекою функцій вводу-виводу
#include <iostream>
//заголовковий файл cmath містить математичні
//функції, зокрема, pow(x,y)
#include <math.h>
//головна функція (програма)
int main() {
//оголошення змінної дійсного типу
    /*стала пі*/ const float PI = 3.141593;
    float a; //довжина ребра куба
    std::cin >> a;//інструкція вводу даних
    float v;//оголошення змінної для об'єму
    v = a * a * a;
/*змінні в C/C++ можна оголошувати будь-де
 в коді функції, але до їх використання*/
    float r3, r;
    r3 = 3 * v /(4 * PI);
    r = pow(r3, 1.0/3.0);
//вивід результату
    std::cout<<"Радіус кулі "<<r<<std::endl;
     return 0;// у випадку успішного виконання програми
// на C/C++ повертають значення 0
}
```

Програма на Java.

Розробники мови ***Java*** для її створення запозичили синтаксичні конструкції з ***C++.*** Це було зроблено для зручності програмістів, більшість з яких на час створення Java користувалися ***C*** чи ***C++***. Тому, в програмах на ***Java*** ми зустрінемо багато коду, схожого на ***C++*** (оголошення змінних, запис коментарів, арифметичні оператори та алгоритмічні конструкції, тощо). Проте, ***Java*** є об'єктно-орієнтованою мовою, в java-коді ми не зустрінемо функцій, як в процедурній парадигмі ***C++***. Тут усі функції належать класам та об'єктам, і можуть бути використані лише від імені класу/об'єкта.

«Приладнати» об'єктно-орієнтовану ***Java*** до структурного розв'язування простої обчислювальної задачі можна, дотримуючись наступних засад:

- ✓ *код програми записується в класі, клас має ім'я, що записується після службового* ***class*** *та тіло у фігурних дужках, що містить поля (дані) та методи (функції) класу;*
- ✓ *для того, щоб код класу можна було виконати як самостійну програму він повинен містити «головний метод» з сигнатурою* ***public static void main (String[] args)****, з якого розпочнеться виконання програми;*
- ✓ *тіло методу складається з послідовності інструкцій програми, поміщеної в фігурні дужки, кожна команда закінчується знаком "****;****";*
- ✓ *виконання методу може завершується командою* ***return****, після якої вказується значення, що буде повернене в якості результату методу, метод може не повертати жодного значення, в такому випадку тип його результату позначається службовим словом* ***void****;*
- ✓ *в тілі методу, що не повертає значення (****void****) команда* ***return*** *не обов'язкова, за її відсутності виконання методу завершиться після виконання усіх його команд;*
- ✓ *змінні та константи, що використовуються тілі методу можуть бути оголошені будь-де, але до їх першого використання, змінні оголошені в класі за межами методу називають полями класу, вони доступні в усіх методах класу;*
- ✓ *для створення іменованих констант використовується службове слово* ***final****, записане перед типом константи;*
- ✓ *коментарі, тобто, пояснення до програми, записують у рядку, після пари знаків «слеш» ("****//****"), або між парами символів "****/****" та "*****/****";*
- ✓ *для присвоєння змінній значення використовується знак «дорівнює» (для порівняння значень – знак "****= =****").*

Решта особливостей Java пропонуємо розглянути в наступному коді:

```
//пакет класу Scanner, використаного для вводу даних
import java.util.*;
//класичний компілятор пакет lang підключає автоматично
import java.lang.*;//потрібно лише для ideone.com
import java.io.*;// пакет класів вводу-виводу

class Task30 {
  public static void main (String[] args) {
    final double PI = 3.141593;
    double a; //довжина ребра куба
```

```
    // Створюємо об’єкт класу Scanner для вводу даних
    Scanner scanner = new Scanner(System.in);
    a = scanner.nextDouble();//інструкція вводу даних
    double v;//оголошення змінної для об'єму
    v = a * a * a;
    double r3, r;
    r3 = 3 * v /(4 * PI);
/* усі математичні функції, в тому числі, піднесення
 до степеня є статичними методами бібліотечного кладсу
Math, тому викликаються від його імені*/
    r = Math.pow(r3, 1.0/3.0);
//вивід результату
    System.out.println("Радіус кулі " + r);
  }
}
```

Програма на Python.

Найкоротшим в групі прикладів розв’язування цієї задачі буде код на ***Python***. Причина такої лаконічності криється, можливо, в тому, що ***Python*** – інтерпретована скриптова мова, тоді як інші, розглянуті тут, технології використовують для трансляції коду компіляцію. Щоправда, програми на інтерпретованих мовах виконуються на комп’ютері лише в тому випадку, якщо на ньому встановлено відповідний інтерпретатор.

Основні правила запису програмного коду мовою ***Python***:

- ✓ *код програми складається з інструкцій записаних в окремих стрічках, – інтерпретатор зчитує чергову стрічку коду, перекладає на мову комп’ютера і намагається її виконати;*
- ✓ *у випадку успішного виконання чергової інструкції інтерпретатор переходить до опрацювання наступної, якщо трапилася помилка, повідомляє про неї і зупиняє виконання програми;*
- ✓ *окрім поділу на стрічки, для структурування коду використовуються горизонтальні відступи (для вкладених блоків);*
- ✓ *створення змінних відбувається відразу під час присвоєння їм значень за допомогою знаку “=”, тип змінної визначається відповідно до типу переданого їй значення;*

- ✓ *значення, що вводяться з клавіатури за допомогою функції* ***input*** *мають тип* ***string*** *(рядок тексту), для вводу числових даних слід виконувати їх приведення до потрібного типу;*
- ✓ *створення іменованих констант в Python не передбачено;*
- ✓ *коментарі записують у рядку, починаючи зі знаку "#" та пробілу;*
- ✓ *для присвоєння змінній значення використовується знак «дорівнює» (для порівняння значень – знак "= =").*

Код програми для розв'язування задачі 30 на мові ***Python***:

```
PI = 3.141593

# Ввід дійсного числа - довжини ребра куба

a = float(input()) # текстовий результат функції input

# приводиться до типу float – дійсне число

v = a**3

r3 = 3*v/(4*PI)

r = r3**(1/3)# піднесення до степеня 1/3, тобто,

# корінь кубічний

print("Радіус кулі: ", r)
```

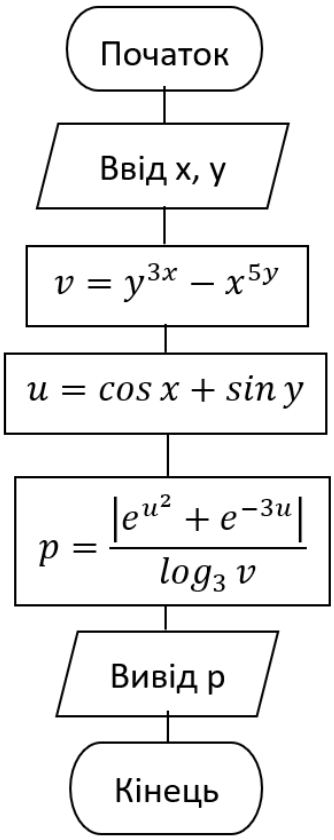
60. Запрограмувати обчислення виразу за заданими формулами для введених користувачем вхідних даних:

$$p = \frac{\left|e^{u^2} + e^{-3u}\right|}{log_3 v}, \text{ де } u = cos\, x + sin\, y,\ v = y^{3x} + x^{5y},\ x, y > 0.$$

Задача 60 відрізняється від попередньої хіба-що тим, що усі формули для обчислення задані вже в умові, а самі вирази не мають геометричного чи фізичного когнтексту. Єдине, що може викликати роздуми, – які саме змінні слугують вхідними даними, а які є результатом обчислень, а також звідки брати значення числа e.

Щодо числа e, то знати його значення не потрібно, математичні бібліотеки усіх мов програмування містять функцію exp(x) («експонента») для обчислення виразу e^x. За потреби ж дізнатися значення самого числа e достатньо обчислити значення цієї функції при $x = 1$.

Щоб встановити вхідні дані для обчислень слід уважно розглянути формули для обчислення p. Окрім стандартних елементарних функцій тут фігурують дві невідомі величини u та v. Для знаходження u та v задано два наступні вирази, які, очевидно є функціями від невідомих x та y. Отже, задавши деякі значення для x та y, ми зможемо обчислити відповідні значення u та v, а за ними знайти результат p.



Тобто, схематично задачу можна сформулювати так: за заданими значеннями x та y обчислити значення p, використовуючи наступні формули. Алгоритм розв'язання задачі такий:

- *користувач вводить два числа – значення невідомих x та y;*
- *програма знаходить значення «проміж-них» змінних u та v за відповідними формулами;*
- *за отриманими значеннями u та v програма обчислює результат, тобто значення величини p;*
- *обчислений результат програма повідомляє користувачеві і завершує свою роботу.*

Графічне зображення сформульованого вище словесно алгоритму подано у вигляді блок-схеми на малюнку.

Програма на Pascal.

Враховуючи усі зауваження щодо програмування на ***Pascal***, зроблені при розгляді задачі 30, нагадаємо, що операція піднесення до степеня реалізовується за формулою $a^x = e^{x \cdot ln\, a}$. Окрім цього, у формулах для

обчислень фігурує логарифм за основою 3, його обчислюють за формулою переходу до нової основи $log_3 x = \frac{lnx}{ln3}$.

Реалізація описаного алгоритму на паскалі виглядатиме так:

```
program task60;

var p,u,v,x,y:real;

begin

     readln(x,y);

     u := cos(x) + sin(y);

     v := exp((3*x)*ln(y)) + exp((5*y)*ln(x));

     p:= abs(exp(u*u) +

          exp(-3*u))/(ln(v)/ln(3));

     { p:12:5 вказує, що дійсне число p при виводі на

     екран потрібно показати з 5-ма цифрами після

     коми, а всього для виводу числа p використати

     12 знаків, включно з пробілами на початку}

     writeln("Значення виразу ",p:12:5);

end.
```

Щодо виводу дійсних чисел засоби виводу різних мов, мають ті чи інші стандартні способи їх представлення на консолі. ***Pascal***, зокрема, подає їх у так званій «експоненційній формі», тобто в канонічному вигляді. Так, наприклад дійсне число x = 77.51, при виводі на екран функцією ***writeln(x)***, буде виглядати як 7.751000000000000e+01. Це не дуже зручно для розуміння числових результатів людиною. Для таких випадків передбачено формат виводу даних у функції writeln, – перше ціле число, записане через двокрапку після змінної в списку означає загальну довжину (у символах) стрічки на екрані, на якій буде виведено значення. Для дійсних значень формат задають двома цілими числами, друге вказує кількість десяткових знаків після коми.

Програма на C++.

У C++ потік виводу на консоль, що представлений об’єктом cout, більш гнучко вибирає стандартний спосіб виводу тих чи інших даних. Дійсне число x = 77.51 командою ***cout<<x;*** буде виведено як 77.51. Але для виведення, наприклад, результатів наближених обчислень форматування виводу може знадобитися. В C/C++ для цього зручно використовувати функцію ***printf()***, першим аргументом якої є так звана «форматна стрічка». Вона показує загальний вигляд тексту, що буде виведений, а місця, куди потрібно вставити значення змінних зі списку виводу резервуються спеціальними форматними

покажчиками, що починаються знаком відсотка. Наприклад, форматний специфікатор "***%10.4f***" в наступному коді вказує, що значення p слід відобразити у фіксованому (f – fixed) форматі з чотирма цифрами після коми, заповнивши при цьому стрічку з 10 символів числом, включно з пробілами на початку.

```
#include <iostream>
#include <math.h>
using namespace std;


int main() {
     double x,y;
     cin>>x>>y;
     //змінні описуємо по мірі їх використання
     double u,v;
     u = cos(x) + sin(y);
     v = exp((3*x)*log(y)) +
          exp((5*y)*log(x));
     double p;
     p= abs(exp(u*u) +
          exp(-3*u))/(log(v)/log(3));
     //форматований вивід p на 10 позицій
     //з 4-ма знаками після коми
     printf("Значення виразу %10.4f\n",p);
     return 0;
}
```

Програма на Go.

Компільована мова програмування Go зі статичною типізацією розроблена в компанії Google побачила світ наприкінці 2007 року. Заснована на C++, Java та Python, підтримує імперативну та процедурні парадигми, має оригінальну та просту концепцію ООП та засоби паралельного виконання за допомогою потоків.

В структурі програми на Go, попри оригінальний синтаксис, проглядаються базові концепції C/C++:

- ✓ *програма складається з функцій, функції можуть викликати одна одну, функції розміщуються у пакетах, (часто пакет = файл коду, але не обов'язково);*
- ✓ *виконання програми починається з головної функції («точки входу») з назвою* **main** *з пакету* **main**;
- ✓ *функція починається з ключового слова* **fun** *складається з заголовку та тіла, заголовок містить назву функції та список параметрів функції в круглих дужках, якщо функція не приймає параметрів, дужки залишають порожніми, якщо функція повертає результат, його тип вказують після списку параметрів;*
- ✓ *тіло функції складається з послідовності інструкцій програми, поміщеної в фігурні дужки, кожна інструкція поміщається в окремому рядку, дві інструкції, розміщені в одному рядку, розділяють знаком "**;**";*
- ✓ *виконання функції завершується командою* **return**, *після якої вказується значення, що буде передано в якості результату функції;*
- ✓ *функції можуть не повертати жодного значення, в такому випадку тип результату у заголовку не вказують, а команда* **return** *необов'язкова;*
- ✓ *змінні та константи, що використовуються тілі функції можуть бути оголошені будь-де, але до їх першого використання для вводу чи присвоєння даних;*
- ✓ *для створення іменованих змінних використовується службове слово* **var** *для констант –* **const**, *тип змінної (константи) вказується після її імені (або списку імен);*
- ✓ *коментарі записують у рядку, починаючи з пари знаків «слеш» ("**//**"), або між парами символів "**/***" та "***/**";*
- ✓ *для присвоєння змінній значення використовується знак «дорівнює», ініціалізацію змінної можна сумістити відразу з її створенням за допомогою вказання значення після знаку "**:=**".*

Застосування цих правил та загальну структуру програми на ***Go*** (***Golang***) пропонуємо розглянути в поданому нижче коді:

```
package main

import "fmt"

import "math"

func main(){

    var x,y float64

    fmt.Scan(&x); fmt.Scan(&y) // змінні можна

    // створювати ініціалізуючи їх значеннями

    // за допомогою знака ":="

    u := math.Cos(x) + math.Sin(y);
```

```
    v := math.Pow(y,(3*x)) + math.Pow(x,(5*y))
    p := math.Abs(math.Exp(u*u) +
        math.Exp(-3*u))/(math.Log(v)/math.Log(3))
    //форматований вивід p на 10 позицій
    //з 4-ма знаками після коми
    fmt.Printf("Значення виразу %10.4f\n",p)
}
```

Для дійсних змінних в прикладі використано тип даних ***float64*** – аналог типу ***double*** в C++, що вміщає дійсні числа, використовуючи 8 байтів пам'яті. Бібліотечні функції викликають через крапку від імені відповідного пакету. Форматований вивід числових даних аналогічний до C++, відрізняється хіба що способом виклику функції ***Printf()***.

Програма на Python.

Щодо обчислювальних можливостей ***Python*** зазначимо, що тут присутній оператор піднесення до степеня, а більшість елементарних математичних функцій викликаються від імені модуля ***math***, який попередньо слід підключити інструкцією ***import***:

```
# Підключення модуля елементарних
import math # математичних функцій
x = float(input()) # ввід дійсних значень та створення
y = float(input()) # змінних x, y типу float
u = math.cos(x) + math.sin(y)
v = y ** (3*x) + x ** (5*y)
p = abs(math.exp(u*u) +
math.exp(-3*u))/(math.log(v)/math.log(3))
# Вивід значення p з чотирма знаками
# після коми на 10 позицій
print("Значення виразу %10.4f\n" % p)
```

Для коректної роботи вводу даних в ideone.com слід вводити дані в stdin по одному числу в рядку.

Програма на VB.net.

BASIC (Beginners' All-purpose Symbolic Instruction Code) – один з найдавніших представників інтерпретованих мов програмування. Порівняно з іншими мовами, ця мова, за свою більш ніж півстолітню історію, зазнала найбільше з видозмін та мала багато різних діалектів. Перші її версії розпочиналися з класичної імперативної парадигми, сучасні підтримують ООП і за можливостями можуть конкурувати з C++ чи Java. Актуальний діалект бейсика ***Visual Basic .net*** є частиною мультимовної платформи ***.Net*** від компанії Microsoft, а інший варіант – ***Visual Basic for Application*** (***VBA***) вбудовано в програми пакету Microsoft Office. Тут ми розглянемо розв'язок задачі 60 на ***VB.net*** з використонням онлайн-середовища ***ideone.com***.

Основні правила запису програмного коду мовою ***Visual Basic***:

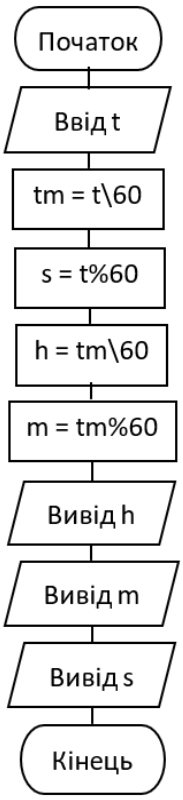
- ✓ *код програми складається з процедур* **`Sub`** *та функцій* **`Function`**, *розміщених в модулі (або в класі, як у прикладі далі), виконання програми починається з процедури з назвою* **`Main`**;
- ✓ *інструкції процедур та функцій записують в окремих стрічках, починаючи від заголовку процедури/функції і до інструкції її завершення* **`End Sub`**(**`End Function`**);
- ✓ *інтерпретатор зчитує та виконує чергову стрічку коду, у випадку успішного її виконання інтерпретатор переходить до опрацювання наступної, якщо трапилася помилка, повідомляє про неї і зупиняє виконання програми;*
- ✓ *для розміщення двох інструкцій в одній стрічці використовується розділювач "*`:`*", розбити одну інструкцію на стрічки можна лише за допомогою спеціального знака переносу, що утворюється з пробілу та знака підкреслення "* `_`*";*
- ✓ *для оголошення змінних використовується інструкція опису змінної* **`Dim`** *в якій вказують назву змінної та її тип після службового слова* **`As`**, *при оголошенні декількох змінних в одній інструкції опису, тип необхідно вказувати після кожної змінної окремо;*
- ✓ *для створення іменованих констант використовується інструкція, що починається службовим словом* **`Const`**, *опис кожної константи завершується заданням її значення після знаку "*`=`*";*
- ✓ *коментарі записують у рядку, що починається з одинарної верхньої лапки.*

Код програми для розв’язування задачі 60 на мові ***Visual Basic*** в онлайн-середовищі ideone.com:

```
Imports System

Public Class Test 'таку назву класу використовує ideone

    Public Shared Sub Main()

        'Оголошуємо змінні дійсного типу

        Dim x As Double : Dim y As Double

        'Ввід даних з консолі та перетворення

        ' їх до відповідного типу

        x = Double.Parse(Console.ReadLine())

        y = Double.Parse(Console.ReadLine())

        Dim u As Double, v As Double, p As Double

        u = Math.Cos(x) + Math.Sin(y)

        v = y ^ (3 * x) + x ^ (5 * y)

        p = Math.Abs(Math.Exp(u * u) + _

        Math.Exp(-3 * u)) / (Math.Log(v) / Math.Log(3))

        'Вивід значення p з чотирма знаками

        'після коми на 10 позицій

        Console.WriteLine( _

                 String.Format("p = {0,10:N4}", p))

        Console.ReadLine()

    End Sub

End Class
```

Форматований вивід дійсних чисел у ***Visual Basic*** доволі складний. Перший параметр методу ***Format*** класу ***String***, як і в інших мовах задає «форматну стрічку». Проте, в цій стрічці вказують також і номер змінної зі списку виводу. Тобто, порядок виводу змінних зі списку функції керується самою форматною стрічкою, включно з повторами одного і того ж значення.

90. Задана кількість секунд, що пройшла від початку доби до поточного моменту. Вивести поточний момент часу в годинах, хвилинах та секундах.



Задачі третього блоку цього розділу (задачі 61-90) відрізняються від попередніх тим, що передбачають виконання обчислень з цілими числами, зокрема ділення з остачею. Відповідно, в програмах розв'язування задачі 90 слід використовувати так звані «цілочисельні» арифметичні операції обраної мови програмування. Сам алгоритм розв'язування залишається лінійним:

- *користувач вводить ціле число* t*, що задає кількість секунд, яка пройшла від опівночі;*
- *програма знаходить цілу частину від ділення заданого числа на 60, число* tm*, що буде означати кількість повних хвилин, що пройшла від опівночі;*
- *програма знаходить остачу* s *від ділення введеного користувачем числа* t *на 60, яка показує кількість секунд, що пройшла з останньої повної хвилини;*
- *обчислену на другому кроці цього алгоритму кількість хвилин* tm *знову ділимо на 60: знайдена ціла частина* h *буде вказувати кількість годин від опівночі, а остача* m*, – кількість хвилин, що минула від початку поточної години;*
- *знайдені кількості годин, хвилин і секунд програма повинна послідовно повідомити користувачеві і після цього завершити свою роботу.*

Графічне зображення сформульованого вище словесного опису алгоритму подано у вигляді блок-схеми на малюнку. Для позначення ділення націло у Visual Basic використано оператор "\", а для знаходження остачі від ділення – "%" (як в C++, C#, Python, Java).

Програма на Pascal.

В Pascal цілочисельні операції позначаються за допомогою службових слів: ***div*** – ділення націло та ***mod*** – остача від ділення. Ввід цілого числа здійснюється процедурою ***readln()*** без додаткових маніпуляцій в коді:

```
program mytime;

var t,tm: integer;

h,m,s:integer;

begin

    readln(t);

    tm := t div 60;

    s := t mod 60;
```

```
    h := tm div 60;
    m := tm mod 60;
    writeln("Час ", h:2,":", m:2,":", s:2);
end.
```

Формат “***:2***” вказує вивід цілого числа двома цифрами, а одноцифрові числа будуть виведені з пропуском зліва.

Програма на C++.

В C++ операція ділення “***/***” виконується по різному, залежно від її операндів, – цілі числа діляться націло, числа з дробовою при діленні дають числа з дробовою частиною. Така особливість ділення в C/C++ (C#, Java) деколи призводить до колізій: наприклад, запис “***1/3***”, який математично сприймається, як число «одна третя» в коді на C/C++ означає ділення цілого числа ***1*** на ціле число ***3*** і дорівнює нулю. Але у випадку нашої задачі нас цікавить саме таке ділення. Для знаходження остачі від ділення в C/C++ використовується оператор “***%***”.

```
#include <iostream>
using namespace std;

int main() {
    int t, t1;
    cin>>t;
    t1 = t / 60;
    int s = t % 60;
    int h = t1 / 60;
    int m = t1 % 60;
    printf("Час %02d:%02d:%02d\n", h, m, s);
}
```

Форматний дескриптор “***%02d***” забезпечує вивід цілого числа двома цифрами, причому до одноцифрового числа буде дописано нуль зліва.

Програма на C#.

Ще один об'єктно-орієнтований «нащадок» ***C/C++*** – ***C#*** багато в чому схожий на ***Java***: код так само структурується у вигляді класів та їх методів, програми виконуються на віртуальній машині, що виділяє та розподіляє ресурси і забезпечує кросплатформність програм. Правила написання коду також аналогічні до ***Java***:

- ✓ *код програми записується в класі, у вигляді методів, клас має ім'я, що записується після службового* **`class`** *та тіло у фігурних дужках з полями (дані) та методами (функції) класу;*
- ✓ *для того, щоб код класу можна було виконати як самостійну програму він повинен містити «головний метод» з сигнатурою* **`public static void Main (String[] args)`**, *з нього розпочинається виконання програми;*
- ✓ *тіло методу складається з послідовності інструкцій програми, кожна команда закінчується знаком "`;`", поділ коду на стрічки, взагалі кажучи, до уваги не береться;*
- ✓ *виконання методу може завершується командою* **`return`**, *після якої вказується значення, що буде передано в якості результату методу;*
- ✓ *метод може не повертати жодного значення, в такому випадку тип результату у заголовку задається службовим словом* **`void`**;
- ✓ *в тілі методу, що не повертає значення (*`void`*) команда* **`return`** *не обов'язкова, за її відсутності виконання функції завершиться після виконання останньої інструкції тіла;*
- ✓ *змінні та константи, що використовуються тілі методу можуть бути оголошені будь-де, але до їх першого використання для присвоєння чи вводу даних;*
- ✓ *змінні оголошені в класі за межами методу називають полями класу, вони доступні в усіх методах класу;*
- ✓ *для створення іменованих констант використовується службове слово* **`const`**, *записане перед типом константи;*
- ✓ *коментарі, тобто, пояснення до програми, записують у рядку, що починається з пари знаків «слеш» ("`//`"), або між парами символів "`/*`" та "`*/`";*
- ✓ *для присвоєння змінній значення використовується знак «дорівнює» (для порівняння значень – знак "`==`").*

Код програми на C# для поділу часу на години, хвилини, секунди:

```
using System;
public class Test// таку назву класу використовує ideone
{
     public static void Main()
     {
          int t, tm;
          // зчитування цілого числа з консолі
          t = int.Parse(Console.ReadLine());
          tm = t / 60;
          int s = t % 60;
          int h = tm / 60;
          int m = tm % 60;
          Console.WriteLine(
          string.Format("Час {0,2:D2}:{1,2:D2}:{2,2:D2}",
                        h, m, s));//форматний вивід
     }
}
```

Форматування виводу відрізняється від C++, технологія ***Microsoft .NET*** використовує для форматування тексту з даними так звані, інтерполяційні стрічки. В прикладі результати виводять в форматі D2, що вказує вивід цілого числа двома цифрами.

Програма на VB.net.

Для ділення націло в ***Visual Basic .net*** визначена окрема операція, що позначається знаком “****”, а остачу від ділення шукають оператором ***Mod***, як в Pascal. Попри оригінальний синтаксис та трансляцію інтерпретатором, програма на ***VB.net***, за структурою та засобами вводу-виводу нагадує ***C#***. Це природньо, адже обидві мови належать до однієї технології технологія ***Microsoft .Net***.

Код програми розв’язування задачі 90:

```
Imports System

Public Class Test

    Public Shared Sub Main()

        Dim T As Integer, TM As Integer, _

        H As Integer, M As Integer, S As Integer

        T = Integer.Parse(Console.ReadLine)

        TM = T \ 60 : S = T Mod 60

        H = TM \ 60 : M = TM Mod 60

        Console.WriteLine(

        "Час {0,2:D2}:{1,2:D2}:{2,2:D2}", H, M, S)

    End Sub

End Class
```

Програма на Python.

Для ділення націло натуральних чисел в ***Python*** можна використати оператор “***//***”. Насправді, цей оператор виконує ділення з заокругленням до меншого цілого, але для додатних чисел така дія співпадає з діленням націло. Знаходження остачі від ділення позначається знаком відсотка “**%**”, як і в С-подібних мовах.

Код мовою ***Python***:

```
t = int(input())

tm = t // 60

s = t % 60

h = tm // 60

m = tm % 60

print("Час %02d:%02d:%02d\n" % (h, m, s))
```

Як і в попередніх задачах, код на Python найкоротший, бо містить лише команди програми, що виконуються інтерпретатором.

Розділ 2. Програмування алгоритмів розгалуженої структури

Алгоритмічна конструкція розгалуження. Повна та коротка форма розгалуження. Перемикачі.

91. Задано три цілі числа. Знайти кількість додатних чисел в цьому наборі.

92. Задано три дійсні числа. Знайти кількість від'ємних чисел в цьому наборі.

93. Задано дві змінні дійсного типу: A, B. Перерозподілити значення даних змінних так, щоб A виявилось меншим із заданих значень, а B - більше. Вивести нові значення змінних A і B.

94. Задано дві змінні цілого типу: A і B. Якщо їхні значення не рівні, то присвоїти кожній змінній суму цих значень, а якщо рівні, то присвоїти змінним нульове значення. Вивести нові значення змінних.

95. Задано два цілих числа: A і B. Якщо обидва числа парні, то зменшити їх значення вдвічі, якщо обидва непарні, то присвоїти кожному півсуму їх значень. Вивести нові значення змінних.

96. У магазині при покупці товару на суму від S_1 грн діє знижка p_1% та p_2%, – при покупці на суму від S_2 грн ($S_1 < S_2$). На покупки сумою до S_1 знижка не поширюється. Скласти програму, яка для заданих значень S_1, S_2, p_1 та p_2, за введеною з клавіатури вартістю покупки розраховує суму до сплати з урахуванням знижки.

97. У платіжній системі комісія за переказ коштів складає 10 грн, якщо сума переказу менша за 1000 грн, 1%, – на перекази сумою від 1000 до 10000 грн включно і 0,5% на суми, більші за 10000 грн. Скласти програму, яка для заданої суми переказу S розраховує розмір комісії та суму до сплати з урахуванням комісії.

98. Задано три цілі числа. Знайти найменше з них.

99. Задано три цілі числа. Визначити скільки серед них парних.

100. Задано три дійсні числа. Знайти середнє з них (тобто число, розташоване між найменшим і найбільшим).

101. Задано координати точки, що не лежить на координатних осях OX та OY. Визначити номер координатної чверті, в якій знаходиться дана точка.

102. Задано цілочисельні координати трьох вершин прямокутника, сторони якого паралельні координатним осям. Знайти координати його четвертої вершини.

103. Перевірити чи задане шестицифрове натуральне число є паліндромом (число, утворене записом цифр заданого числа у зворотному порядку дорівнює самому числу).

104. Тіло масою m з об'ємом V занурюють в рідину густиною ρ. Дослідити чи тіло плаватиме в рідині і вивести відповідне повідомлення.

105. Для введених з клавіатури двох цілих чисел визначити найбільшу величину серед їх суми, різниці та добутку.

106. На ділянці автошляху з обмеженням швидкості M км/год автомобіль рухається зі швидкістю V км/год . Якщо швидкість автомобіля перевищує обмеження не більше, ніж на $p\%$, патрульний повинен зупинити автомобіль і зробити водієві авта усне попередження. Якщо ж перевищення швидкості більше за $p\%$ від обмеження, патрульний повинен накласти на водія штраф у сумі S грн. Написати програму, яка моделює алгоритм дій патрульного.

107. У торгівельній мережі для зменшення втрат пов'язаних з термінами зберігання продуктів запроваджено гнучку систему ціноутворення: на продукти, до кінця зберігання яких залишилося менше половини терміну придатності діє знижка у розмірі p%, а товари, термін придатності яких спливає наступної доби продають за ціною p% від початкової. Скласти програму, яка за заданою ціною товару, терміном його придатності та кількістю діб, що пройшла з дня його виготовлення визначатиме поточну його ціну, за вказаним вище правилом.

108. Для введених з клавіатури двох дійсних чисел визначити найбільшу величину серед їх суми, добутку та частки.

109. Задано три дійсних числа. Знайти суму двох менших з них.

110. Задано три цілі числа. Знайти добуток двох менших з них.

111. Оплата за телефонний дзвінок стягується за таким тарифом: p грн за першу хвилину розмови, незалежно від тривалості дзвінка, q грн за кожну наступну хвилину з посекундною тарифікацією, після 10 хв розмови плата не стягується взагалі. Скласти програму, яка для заданої тривалості дзвінка за заданими тарифами p та q розраховує його вартість.

112. Задано три цілі числа. Визначити скільки з них є двоцифровими.

113. Задано три дійсні числа. Знайти суму двох більших з них.

114. Підприємець сплачує фіксовану суму податку у розмірі X грн, якщо сума його місячного прибутку менша за S_1 грн. Якщо ні, то X грн плюс $p\%$ від суми, яка перевищує S_1. Якщо ж сума прибутків підприємця перевищує S_2 ($S_2 > S_1$), то фіксована сума не стягується, а нараховується $q\%$ податку від загальної суми прибутків. Скласти програму для обчислення розміру податку на вказану суму прибутку.

115. Фільми на диску зберігаються у трьох різних папках, залежно від їх тривалості. Фільми тривалістю до 35 хв записують у папку «коротко-

метражні», фільми тривалістю від 35 хв до однієї години, – в папку «телефільми», а фільми, довші за 1 годину, в папку «кіно». Скласти програму, яка за введеною з клавіатури тривалістю фільму повідомить назву папки, до якої слід завантажити фільм.

116. Задано три цілі числа. Вивести спочатку найменше, а потім найбільше з даних чисел.

117. На координатній площині побудоване коло радіуса R з центром в початку координат та точка $A(x; y)$. За заданим радіусом кола та координатами точки встановити де розміщена точка: всередині кола, поза колом, чи на колі.

118. На координатній площині задано пряму $y = kx + b$ $(k \neq 0)$ та точку $A(x; y)$. За заданими параметрами рівняння та координатами точки A встановити де розміщена точка: над прямою $y > kx + b$, під прямою $y < kx + b$, чи на прямій.

119. Задано три цілі числа. Знайти різницю між найбільшим і найменшим з них.

120. Задано натуральне число з діапазону 1 – 9999. Вивести його опис у вигляді: «одноцифрове число», «двоцифрове число» і т. д.

121. За заданими коефіцієнтами p та q знайти корені квадратного рівняння $x^2 = px + q$. Якщо рівняння має два різні корені x_1 та x_2, то знайти суму їх кубів. Якщо рівняння має один корінь x_1, то обчислити величину $\lg|x_1|$. А якщо рівняння коренів не має, то вивести відповідне повідомлення, а також обчислити відношення $\frac{p+q}{pq}$.

122. За заданими коефіцієнтами a, b та c знайти корені квадратного рівняння $ax^2 + bx = c$. Якщо рівняння має два різні корені x_1 та x_2, то знайти суму їх модулів. Якщо рівняння має один корінь x_1, то обчислити величину $sin\left(\sqrt{|x_1|}\right)$. А якщо ріняння коренів не має, то вивести відповідне повідомлення, а також обчислити відношення $\frac{-b}{2a}$.

123. За заданими коефіцієнтами p та q знайти корені квадратного рівняння $qx^2 + x + p = 0$. Якщо рівняння має два різні корені x_1 та x_2, то знайти різницю $x_1^4 - x_2^3$. Якщо рівняння має один корінь x_1, то обчислити величину $\log_7|pqx_1|$. А якщо ріняння коренів не має, то вивести відповідне повідомлення, а також обчислити відношення $\frac{-q}{p}$.

124. За заданими коефіцієнтами p та q знайти корені квадратного рівняння $x^2 - px - q = 0$. Якщо рівняння має два різні корені x_1 та x_2, то знайти різницю $x_1^3 - x_2^4$. Якщо рівняння має один корінь x_1, то обчислити величину $sin|qx_1|$. А якщо ріняння коренів не має, то вивести відповідне повідомлення, а також обчислити відношення $\frac{q-p}{p}$.

125. За заданими коефіцієнтами p та q знайти корені квадратного рівняння $x^2 + px = q$. Якщо рівняння має два різні корені x_1 та x_2, то знайти суму їх кубів. Якщо рівняння має один корінь x_1, то обчислити величину $cos|x_1|$. А якщо ріняння коренів не має, то вивести відповідне повідомлення, а також обчислити відношення $\frac{q^3}{p}$.

126. За заданими коефіцієнтами a, b та c знайти корені квадратного рівняння $ax^2 + bx + c = 0$. Якщо рівняння має два різні корені x_1 та x_2, то знайти суму їх модулів. Якщо рівняння має один корінь x_1, то обчислити величину $ln\left(\sqrt{|x_1|}\right)$. А якщо ріняння коренів не має, то вивести відповідне повідомлення, а також обчислити відношення $\frac{c+b}{3a}$.

127. За заданими коефіцієнтами p та q знайти корені квадратного рівняння $x^2 = px - q$. Якщо рівняння має два різні корені x_1 та x_2, то знайти добуток $x_2 \cdot \ln|x_1|$. Якщо рівняння має один корінь x_1, то обчислити величину $\mathrm{tg}(x_1)$. А якщо ріняння коренів не має, то вивести відповідне повідомлення а також обчислити відношення $\frac{pq}{p-q}$.

128. За заданими коефіцієнтами a, b та c знайти корені квадратного рівняння $ax^2 - bx = c$. Якщо рівняння має два різні корені x_1 та x_2, то знайти $sin\left(\sqrt{|x_1 + x_2|}\right)$. Якщо рівняння має один корінь x_1, то обчислити величину $\sqrt[5]{x_1^4}0$. А якщо ріняння коренів не має, то вивести відповідне повідомлення а також обчислити відношення $\frac{b+c}{4a}$.

129. За заданими коефіцієнтами p та q знайти корені квадратного рівняння $qx^2 - x + p = 0$. Якщо рівняння має два різні корені x_1 та x_2, то знайти різницю $\sqrt[3]{x_1^4} - x_2^3$. Якщо рівняння має один корінь x_1, то обчислити величину $\log_8|qx_1|$. А якщо ріняння коренів не має, то вивести відповідне повідомлення а також обчислити відношення $\frac{q}{p^3}$.

130. За заданими коефіцієнтами p та q знайти корені квадратного рівняння $x^2 - px = q$. Якщо рівняння має два різні корені x_1 та x_2, то знайти різницю $\sin x_1 - \cos x_2$. Якщо рівняння має один корінь x_1, то обчислити величину $p \cdot sin|x_1|$. А якщо ріняння коренів не має, то вивести відповідне повідомлення а також обчислити відношення $\frac{q^2+p^2}{3}$.

131. За заданими коефіцієнтами p та q знайти корені квадратного рівняння $x^2 + px - q = 0$. Якщо рівняння має два різні корені x_1 та x_2, то знайти суму $tg\ x_1 + ctg\ x_2$. Якщо рівняння має один корінь x_1, то обчислити величину $arctg|x_1|$. А якщо ріняння коренів не має, то вивести відповідне повідомлення а також обчислити відношення $\frac{q^2}{p^3}$.

132. За заданими коефіцієнтами a, b та c знайти корені квадратного рівняння $ax^2 + bx + c = 0$. Якщо рівняння має два різні корені x_1 та x_2, то знайти $\frac{x_1+x_2}{3ac}$. Якщо рівняння має один корінь x_1, то обчислити величину $log_3\left(\sqrt{|x_1|}\right)$. А якщо ріняння коренів не має, то вивести відповідне повідомлення а також обчислити відношення $\frac{a+b}{a-b}$.

133. За заданими коефіцієнтами a, b та c знайти корені квадратного рівняння $ax^2 + bx - c = 0$. Якщо рівняння має два різні корені x_1 та x_2, то знайти різницю їх кубів. Якщо рівняння має один корінь x_1, то обчислити величину $cos^2 x_1$. А якщо ріняння коренів не має, то вивести відповідне повідомлення а також обчислити відношення $\frac{a-b}{c}$.

134. За заданими коефіцієнтами p та q знайти корені квадратного рівняння $qx^2 + x + p = 0$. Якщо рівняння має два різні корені x_1 та x_2, то знайти різницю $x_1^2 - x_2^5$. Якщо рівняння має один корінь x_1, то обчислити величину $\log_3\left|\frac{x_1}{pq}\right|$. А якщо ріняння коренів не має, то вивести відповідне повідомлення а також обчислити відношення $\frac{\sqrt{|q|}}{p}$.

135. За заданими коефіцієнтами p та q знайти корені квадратного рівняння $x^2 - px - q = 0$. Якщо рівняння має два різні корені x_1 та x_2, то знайти різницю $x_1^{-1} - x_2^{-2}$. Якщо рівняння має один корінь x_1, то обчислити величину $sin^2 x_1$. А якщо ріняння коренів не має, то вивести відповідне повідомлення а також обчислити відношення $\frac{7}{pq}$.

В задачах 136 – 150 скласти програму для обчислення значення заданої функції $y(x)$ за введеним з клавіатури значенням змінної x:

136. $y = \begin{cases} \sqrt{cos\, x + ln\, x}\,, \ \textit{якщо } x = 3, \textit{або } x = 7. \\ \sqrt[5]{a}x + tgx, \textit{при } 3 < x < 7, \\ \frac{x}{x+a} \ \textit{у решті випадків.} \end{cases}$

137. $y = \begin{cases} a\, lg\, x + \sqrt[3]{|x|}, \ \textit{якщо } x = 8, \textit{або } x = 10, \\ 2a\, cos\, x + 3x^2, \ \textit{якщо } 8 < x < 12, \\ \frac{x+a}{a} \ \textit{у решті випадків.} \end{cases}$

138. $y = \begin{cases} \sqrt{ax^2 + sin\, x}\,, \ \textit{якщо } x = 3 \textit{ або } x = 7, \\ arccos\, x\,, \textit{при } |x| < 1, \\ log_2|x| \textit{ у решті випадків.} \end{cases}$

139. $y = \begin{cases} \sqrt{a + lg\, x}\,, \ \textit{якщо } x > 1, \\ arcsin\, x\,, \ \textit{якщо} |x| < 1, \\ x^a \textit{ у решті випадків.} \end{cases}$

140. $y = \begin{cases} ax^8 \ln x\,, \text{ якщо } 1 \le x \le 2, \\ 1, \text{ якщо } x < 1, \\ e^{ax} \cos ax \text{ якщо } x > 2. \end{cases}$

141. $y = \begin{cases} \pi x^2 - 7/x^2\,, \text{ якщо } x < 1{,}3, \\ ax^6 + 7/\sqrt{x}\,, \text{ якщо } x = 1{,}3, \\ lg(x + 7\sqrt{x}), \text{ якщо } x > 1{,}3. \end{cases}$

142. $y = \begin{cases} ax^5 + bx + c, \text{ якщо } x < 1{,}2, \\ a/x + \sqrt{x^2 + 1}\,, \text{ якщо } x = 1{,}2, \\ (a + bx)/\sqrt{x^{2\cdot} + 1}\,, \text{ якщо } x > 1{,}2. \end{cases}$

143. $y = \begin{cases} x\sqrt[3]{x - a}, \text{ якщо } x > a, \\ x \sin a\, x, \text{ якщо } x = a, \\ e^{-x} \cos a\, x, \text{ якщо } x < a. \end{cases}$

144. $y = \begin{cases} \frac{a+b}{e^x + \cos x}, \text{ якщо } x < 2{,}8, \\ (a + b)/(x + 1)\,, \text{ якщо } 2{,}8 \le x \le 6, \\ e^{-x} \cos a\, x \text{ якщо}, \ x > 6. \end{cases}$

145. $y = \begin{cases} bx - lg\, b\, x, \text{ якщо } x < 1, \\ 1 + x^{10}, \text{ якщо } x = 1, \\ bx + \ln bx \text{ якщо } x > 1. \end{cases}$

146. $y = \begin{cases} \frac{\ln^3 x + x^2}{\sqrt{x+t}}, \text{ якщо } x < 0{,}5, \\ \frac{\sqrt{x+t}+1}{x}, \text{ якщо } x = 0{,}5, \\ \cos x + t \sin^2 x, \text{ якщо } x > 0{,}5. \end{cases}$

147. $y = \begin{cases} \sqrt{ax^2 + \sin x}\,, \text{ якщо } x = -5, \text{ або } x = 2, \\ arcsin\, x\,, \text{ якщо} |x| < 1, \\ log_3 |x|, \text{ у решті випадків.} \end{cases}$

148. $y = \begin{cases} \pi x^2 - 7/x^2\,, \text{ якщо } x = 5, \\ ax^6 + 7/x\,, \text{ якщо } x < 5, \\ lg(x + 7\sqrt{x}), \text{ якщо } x > 5. \end{cases}$

149. $y = \begin{cases} x\sqrt[3]{3x - a}, \text{ якщо } x < a, \\ x \sin a\, x, \text{ якщо } x = a, \\ e^{-x} \cos a\, x \text{ якщо } x > a. \end{cases}$

150. $y = \begin{cases} x\sqrt[3]{ax^3 + \sin x}\,, \text{ якщо } 0 \le x \le 5; \\ a \cdot \sin^3 x\,, \text{ якщо } x < 0; \\ log_x |x^3 - e^x|\,, \text{ якщо } x > 5. \end{cases}$

В задачах 151 – 165 скласти програму для розв'язування алгебраїчної нерівності (системи нерівностей) згідно варіанту. Програма повинна для

заданих значень числових коефіцієнтів a, b та c визначати розв'язок у вигляді проміжків на числовій осі. Виконати програму для різних значень коефіцієнтів a, b і c так, щоб продемонструвати її роботу за усіма можливими гілками розгалуження.

151. $\frac{ax}{x^2+bx+c} \leq 0;$

152. $\begin{cases} x - a \geq 0, \\ x^2 + bx + c > 0; \end{cases}$

153. $\frac{x^2}{ax+b} < c;$

154. $\frac{x-a}{x^2+bx+c} \geq 0;$

155. $\frac{x-a}{x^2+bx+c} > 0;$

156. $\begin{cases} x + a > 0, \\ x^2 + bx + c \leq 0; \end{cases}$

157. $\begin{cases} x - a < 0, \\ x^2 + bx + c > 0; \end{cases}$

158. $\frac{x^2-ax+b}{x+c} > 0;$

159. $\frac{ax}{x^2+bx+c} > 0;$

160. $\frac{x^2-bx-c}{ax} \leq 0;$

161. $\frac{x^2}{ax+b} \geq c;$

162. $\frac{x-a}{x^2+bx+c} < 0;$

163. $(x - a)(x^2 + bx + c) < 0;$

164. $(x + a)(x^2 - bx + c) > 0;$

165. $\begin{cases} x - a \geq 0, \\ x^2 + bx + c < 0. \end{cases}$

166. Задане ціле число d з відрізка [1; 7] задає день тижня починаючи з понеділка (1 – «понеділок», 2 – «вівторок» і т. д.). Скласти програму, яка за введеним числом d виведе відповідну назву дня тижня.

167. Задане ціле число p, що означає оцінку в 12-бальній шкалі. Вивести опис відповідного рівня оцінки: 1-3 – «початковий рівень», 4-6 – «середній рівень», 7-9 – «достатній рівень», 10-12 – «високий рівень». Якщо ж p не належить відрізку [1; 12], вивести рядок «помилка».

168. Ціле число з діапазону 0-24 визначає поточну годину доби. Скласти програму, яка для вказаної години визначає пору доби: від 0 годин до 6 – «ніч», 6-11 – «ранок», 12-18 – «день», 19-24 – «вечір».

169. За введеним порядковим номером місяця в році вивести назву відповідної пори року (наприклад, 1 – «зима», 5 – «весна», 7 – «літо» і т.п.).

170. Ціле число в діапазоні 10-40 вказує кількість тестових завдань з певної теми. Вивести текстовий опис зазначеної кількості завдань з правильним узгодженням числа зі словосполученням «тестове завдання». Наприклад, «18 тестових завдань», «23 тестові завдання», «31 тестове завдання».

171. Написати програму «калькулятор», що працює наступним чином: користувач вводить два дійсні числа, а потім символ арифметичної операції («+» – додавання, «-» – віднімання, «*» – множення, «:» – ділення, «^» – піднесення до степеня). Програма виконує обрану користувачем арифметичну дію і виводить на екран повідомлення, в якому вказано задані користувачем операнди, з’єднані знаком виконаної дії та результат обчислення після знаку «=».

172. Користувач задає коректну сьогоднішню дату за допомогою двох цілих чисел: D (день) і M (місяць). Обчислити значення D (день) і M (місяць) для вчорашньої дати, вважаючи, що рік не високосний.

173. Ціле число з діапазону 20-69 визначає вік відвідувача у роках. Вивести рядок-опис зазначеного віку, додавши до віку слово «рік» у відповідному відмінку множини. Наприклад: «20 років», «32 роки», «41 рік».

174. Задане ціле число p, що задає оцінку в 5-бальній шкалі. Вивести текстову назву відповідної оцінки: 1 або 2 – «незадовільно», 3 – «задовільно»,4 – «добре», 5 – «відмінно». Якщо ж p не належить відрізку [1; 5], вивести рядок повідомлення про помилку.

175. Ціле число в діапазоні 1-20 вказує кількість академічних годин, відведених на вивчення певного розділу. Вивести текстовий опис зазначеної кількості годин з правильним узгодженням числа із множиною іменника «година». Наприклад, «18 годин», «4 години», «1 година».

176. Ціле число з діапазону 0-24 визначає поточну годину доби. Скласти програму, яка для вказаної години виводить фразу для вітання: від 0 годин до 6 – «Доброї ночі!», 6-11 – «Доброго ранку!», 12-18 – «Доброго дня!» 19-24 – «Доброго вечора!».

177. Ціле число з діапазону 20-99 визначає суму переказу у гривнях. Вивести опис зазначеної суми, додавши слово «гривня» у відповідному відмінку множини. Наприклад: «20 гривень», «32 гривні», «41 гривня».

178. Користувач задає коректну вчорашню дату за допомогою двох цілих чисел: D (день) і M (місяць). Обчислити значення D (день) і M (місяць) для сьогоднішньої дати, вважаючи, що рік високосний.

179. Задане ціле d число з відрізка [1; 7] позначає день тижня починаючи з неділі. Скласти програму, яка за введеним числом d виведе відповідну англійську назву дня тижня (1 – «Sunday», 2 – «Monday», і.т.д.)

180. За заданим номером місяця (ціле число з відрізка [1; 12]) визначити кількість днів в цьому місяці, за умови, що рік не високосний.

Практикум до розділу 2. Програмування алгоритмів розгалуженої структури

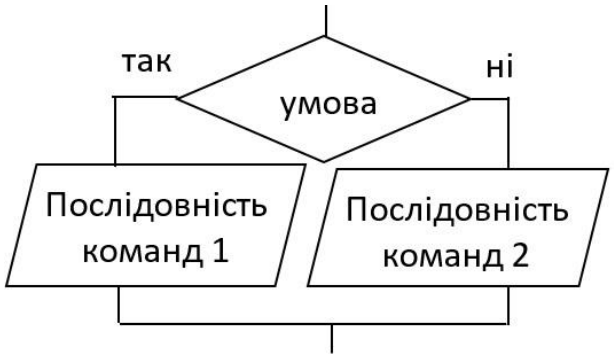
Програми для розв'язування задач другого розділу повинні, вцілому, реалізовувати таку ж послідовність дій, як і програми до задач розділу 1:

- *користувач вводить вхідні дані для розв'язування задачі;*
- *програма обчислює розв'язок задачі за обраними формулами;*
- *отриманий результат програма повідомляє користувачеві і завершує свою роботу.*

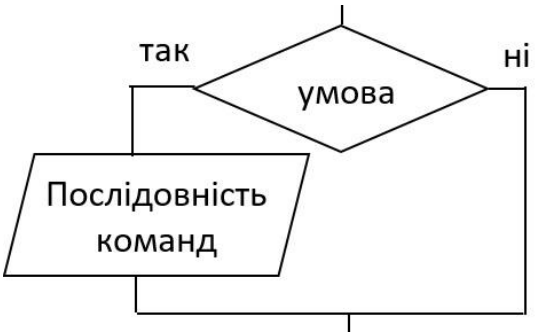
Проте, на етапі обчислення розв'язку, програмі доведеться обирати послідовність команд, які вона виконуватиме для отримання результату, залежно від того чи дані, задані користувачем відповідають тим чи іншим критеріям. Така ситуація в програмуванні називається ***розгалуженням***. Тут частина команд алгоритму може виконуватися чи не виконуватися залежно від результату перевірки деякої умови (значення ***логічного виразу***).

Розгалуження зустрічаються у двох формах: ***повна форма розгалуження*** передбачає дві послідовності команд, одна з яких виконуватиметься, якщо значення логічного виразу істинне, інша, – якщо ні; ***коротка форма розгалуження*** дозволяє виконати деяку послідовність команд, якщо значення логічного виразу істинне, і не виконувати нічого у протилежному випадку. Графічне зображення обидвох форм подано нижче.

Повна форма розгалуження



Коротка форма розгалуження



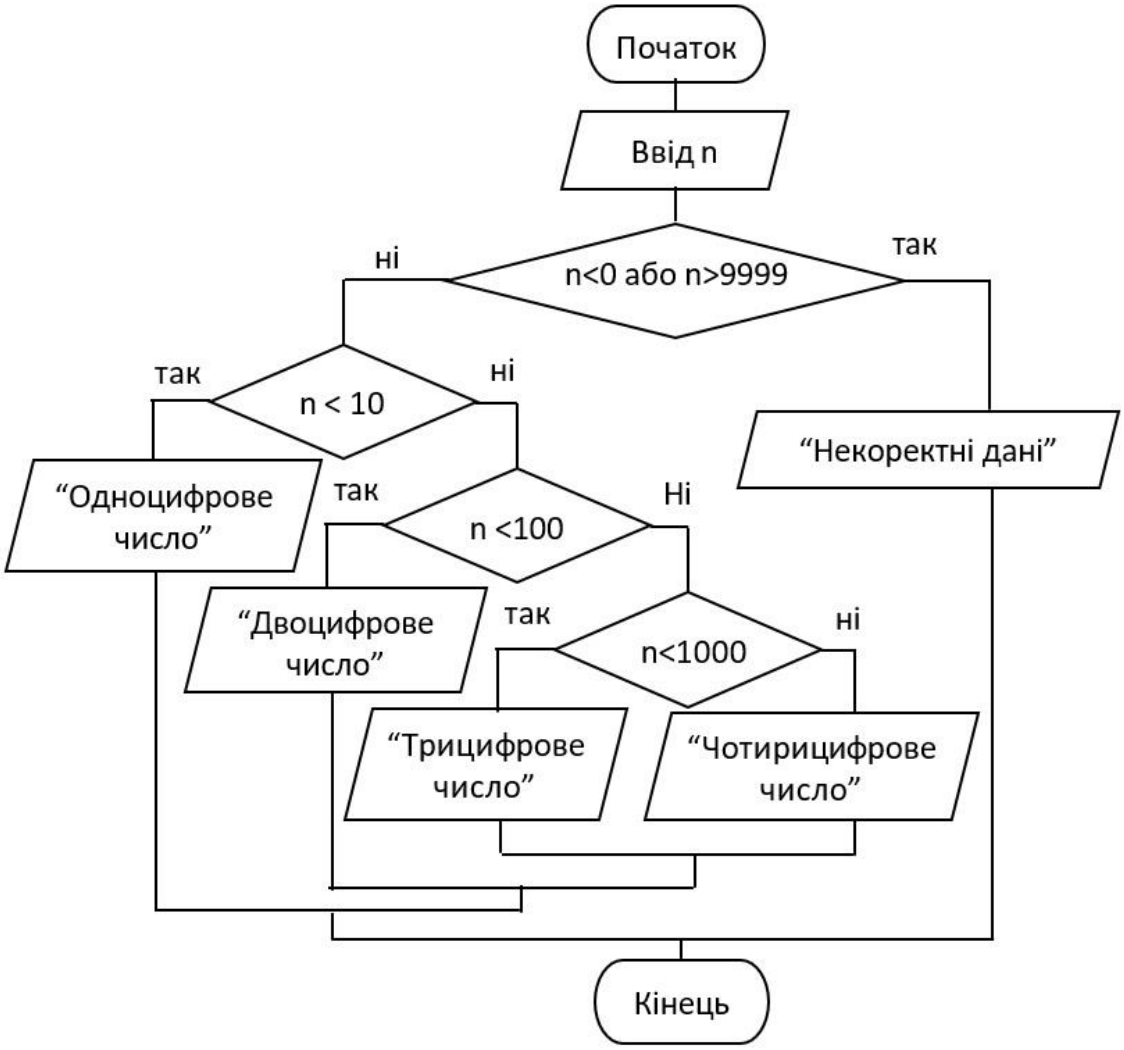
У цьому розділі для написання та відлагодження програм продовжуємо використовувати online-сервіси, – тут скористаємося іншим мультимовним online-компілятором ***onlinegdb.com.*** За кількістю підтримуваних мов програмування він поступається ***ideone.com,*** проте має можливість діалогового вводу даних користувачем, що робить його більш схожим до повноцінних стаціонарних середовищ розробки ПЗ.

120. Задано натуральне число з діапазону 1 – 9999. Вивести його опис у вигляді: «одноцифрове число», «двоцифрове число» і т. д.

Можна побудувати декілька варіантів алгоритму розв'язання сформульованої задачі. Зупинимося на варіанті, що використовує лише розгалуження у повній формі. Для коректної роботи такого алгоритму при довільних значеннях вхідних даних, спочатку слід перевірити чи введене ціле число відповідає умові задачі, тобто належить діапазону [1; 9999]. Якщо введене число відповідає умові, то його класифікацію можна реалізувати за схемою:

- *числа менші за 10 – одноцифрові,*
- *інші числа, але менші за 100, – двоцифрові,*
- *всі інші числа, менші за 1000 – трицифрові,*
- *решта чисел – чотирицифрові.*

Графічне зображення такого підходу подано нижче.



Програма на Pascal.

Алгоритмічна конструкція розгалуження в мові ***Pascal*** реалізовується за допомогою умовного оператора:

```
if <умова> then <оператор 1> else <оператор 2>;
```

Поділ коду на стрічки не береться до уваги компілятором, проте використовується програмістами з метою оформлення коду у зручній для читання формі. Виконання оператора починається з перевірки ***умови***, якщо її результат є істинним, то виконується ***оператор1***, якщо ні, то ***оператор2***.

Вище подано варіант розгалуження у повній формі, коротка форма:

```
if <умова> then <оператор 1>;
```

відрізняється відсутністю блоку ***else*** та тим, що у випадку хибного результату перевірки умови програма одразу переходить до наступної команди.

Вітки розгалуження позначені службовими словами ***then*** та ***else*** можуть містити лише по одному оператору. Якщо в цей оператор потрібно вкласти багато команд, використовують складений оператор:

```
begin <оператор 1>; <оператор 2>; ... <оператор N> end;
```

Розділювач “;”, який в мові Pascal відділяє оператори один від одного, перед ***end*** не ставлять, хоча компілятор не вважає це помилкою, а просто додає до програми ще один порожній оператор, який не виконує жодної дії. А от крапка з комою перед ***else*** є синтаксичною помилкою оскільки розриває суцільну синтаксичну конструкцію повної форми умовного оператора.

Реалізація описаного вище алгоритму розв’язування задачі 120 на мові ***Pascal*** може бути такою:

```
program Task120;

var n: integer;

begin

  write ('Введіть натуральне число від 1 до 9999');

  readln(n);

  if (n > 0) and (n < 10000) then {перевірка того, чи

введене число відповідає умові задачі}

    begin

      if n < 10 then writeln(n, ' - одноцифрове число')

        { після if-then можна ставити лише один оператор}

      else

        begin
```

```
            if n < 100 then
                 writeln(n, ' - двоцифрове число')
            { розбиття операторів на кілька стрічок
              компілятор Pascal до уваги не бере }
            else
              if (n < 1000) then
                  writeln(n, ' - трицифрове число')
              else
                  writeln(n, ' - чотирицифрове число')
            { if-then-else вважається суцільною
              синтаксичною конструкцією, тому
              begin .. end можна не ставити }
          end
    end {знак ; ставити перед else не можна}
  else
    writeln('Число не належить вказаному відрізку');
end.
```

Програма на C++.

Реалізація алгоритмічної конструкції повного розгалуження в C++ складається, формально, з двох операторів: ***if*** та ***else*** і, зазвичай, записується в такій формі:

```
if (<умова>)
{
    <послідовність команд 1>;
}
else
{
    <послідовність команд 2>;
}
```

Хоча поділ коду на стрічки не береться до уваги, і цю конструкцію можна записати навіть в одну стрічку, таке оформлення коду є зручним для читання,

воно використовується більшістю середовищ розробки, а його використання є правилом хорошого тону в спільноті С-програмістів.

Оператор ***else*** формально вважається окремим оператором, тому в ***C/C++***, на відміну від ***Pascal***, крапка з комою перед ***else*** ставиться. Попри це, оператор ***else*** без оператора ***if*** не вживається.

А от ***if*** без ***else*** вживається, і дуже часто, оскільки є реалізацією короткої форми розгалуження:

```
if (<умова>)
{
    <послідовність команд 1>;
}
```

Фігурні дужки для блоків команд операторів ***if*** (***else***) можна не ставити, якщо блок містить єдину команду. Решта особливостей синтаксису умовного оператора С++ пропонуємо розглянути на прикладі коду для розв'язування задачі 120:

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout<<"Введіть натуральне число "
        <<"від 1 до 9999 включно\n";
    cin>>n;
    //первірка, чи користувач задав коректне число
    if (n > 0 && n<10000)
    {
        if(n < 10) cout<<n<<" - одноцифрове число\n";
        //одну інструкцію після if можна ставити без {}

        else
```

```
        {
            if (n < 100)
            {
            //для зручного читання коду опрацювання
            //вітки розгалуження після if та else
            //форматують, зазвичай, окремим блоком
                cout<<n<<" - двоцифрове число\n";
            }
            else
            {
                if (n < 1000)
                {
                    cout<<n<<" – трицифрове число.\n";
                }
                //else з однією командою можна писати так
                else cout<<n<<" – чотирицифрове число.\n";
            }
        }
    }
    else
    {
        cout<<"Ви ввели число, що не належить "
                <<"вказаному відрізку\n";
    }
    return 0;
}
```

Програма на Java.

Як відомо, синтаксичні конструкції ***Java*** запозичені з ***C++***, тому усе сказане вище про оператор розгалуження стосуватиметься як коду на ***C++***, так і коду на ***Java***. Проте є одна цікава відмінність: при форматуванні блоків коду розробники ***Java*** вирішили записувати відкриваючі фігурні відразу після службового слова ***if*** (***else***), до якого стосується блок, а не з нової стрічки, як в C++. Повну форму розгалуження на ***Java*** форматують так:

```
if (<умова>){
    <послідовність команд 1>;
} else {
    <послідовність команд 2>;
}
```

Решта особливостей умовних конструкцій в Java пропонуємо розглянути в наступному коді:

```
//пакет класу Scanner, використаного для вводу даних
import java.util.*;

public class Main{
  public static void main(String[] args) {
    System.out.println("Введіть натуральне число "+
                          "від 1 до 9999 включно");
    int n;
    Scanner input = new Scanner(System.in);
    n = input.nextInt();
    //первірка, чи користувач ввів коректне число
    if(n > 0 && n<10000){
      if(n < 10)System.out.printf(
                       "%5d - одноцифрове число\n",n);
      //одну інструкцію після if можна ставити без {}
      else {
        if(n < 100) {
          System.out.printf("%5d - двоцифрове число\n",n);
        }
        else {
```

```
            if (n < 1000){
              System.out.printf(
                    "%5d - трицифрове число\n", n);
            }
            else System.out.printf(
                     "%5d - чотирицифрове число\n", n);
          }
        }
      }
      else System.out.println("Ви ввели число, " +
                     "що не належить вказаному відрізку");
  }
}
```

Програма на Python.

Враховуючи, що в синтаксичних конструкціях ***Python*** беруть участь не лише вертикальні, а й горизонтальні розділювачі тексту, можна припустити, що синтаксис умовного оператора тут складний і заплутаний. Насправді, це не зовсім так, але конструкцій оператора ***if*** в ***Python*** доволі багато. З іншого боку, розмаїття синтаксичних форм умовного оператора дозволяє писати на ***Python*** добре структурований і лаконічний код. В цьому розділі звернемо увагу лише на ті форми оператора ***if***, що використовуються в програмі розв'язування задачі 120.

Повна форма умовного оператора в ***Python*** має структуру:

```
if <умова> :
   <блок інструкцій 1>
else :
   <блок інструкцій 2>
```

Поділ коду на стрічки в цій конструкції важливий, насамперед, до уваги береться вертикальне розташування початку стрічок з кодом. Якщо не дотримуватися наступних правил запису умовного оператора ***Python*** може не впоратися з інтерпретацією коду і виводити повідомлення про помилки.

- ✓ *пов'язані стрічки коду з* **if** *та* **else** *повинні починатися з однакової кількості відступів зліва (пробілів, чи знаків табуляції), це вказує інтерпретатору на один рівень вкладеності цих команд;*
- ✓ *блоки команд після* **if** *чи* **else** *повинні складатися зі стрічок з коду з однаковою кількістю відступів зліва, але більшою ніж у оператора, до я*

кого вони належать (вкладаються), дужок для об'єднання блоку чи додаткових розділових знаків при цьому не потрібно;

- ✓ *якщо блок команд після* **if** *чи* **else** *містить лише одну команду, його можна записати в тій же стрічці, після знаку* **":"**;
- ✓ *послідовність інструкцій одного рівня вкладеності можна розмістити в одній стрічці, розділяючи їх знаком* **";"**;
- ✓ *блок інструкцій після* **if** *чи* **else**, *записаний в одну стрічку з розділювачем* **";"** *можна помістити в тій же стрічці, після знаку* **":"**.

Коротка форма розгалуження, очевидно, утворюється з описаної вище конструкції відкиданням ***else*** та відповідного блоку інструкцій.

При кодуванні розгалужень з багатьма вітками, як у цій задачі, доводиться використовувати вкладені умовні конструкції, враховуючи використання горизонтальних відступів таке вкладення може бути заплутаним та важким для читання. Уникнути цього можна за рахунок команди ***elif***, що поєднує в собі дію команди ***else*** та подальшої перевірки умови для наступної вітки розгалуження. Ознайомитися з записом цієї конструкції пропонуємо на прикладі програми для розв'язування задачі 120:

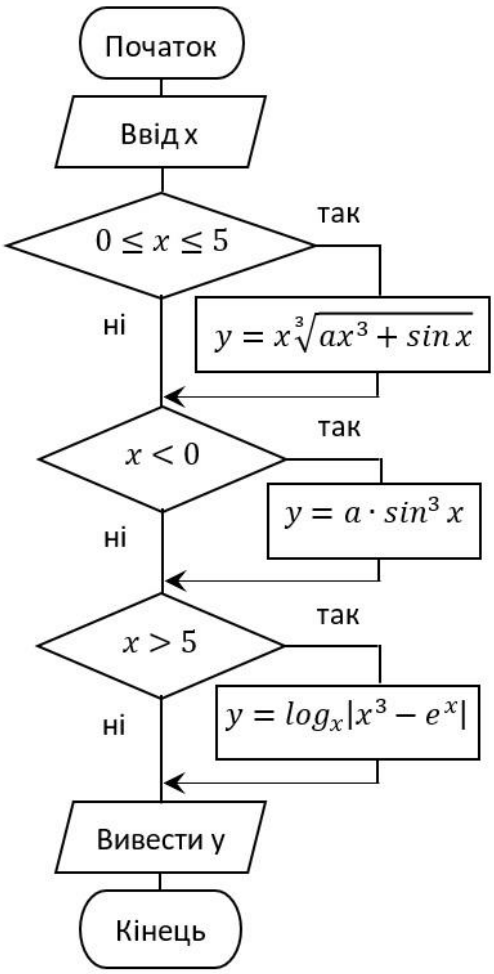
```
print ('Введіть натуральне число від 1 до 9999 включно:')
n = int(input())
if n > 0 and n < 10000:
    if n < 10 :
        print('Число одноцифрове')
    elif n < 100 :
        print('Число двоцифрове')
    elif n < 1000 :
        print('Число трицифрове')
    else :
        print('Число чотирицифрове')
else:
    print('Ви ввели некоректне число')
    print('Число повинно належати відрізку [1;9999]')
```

150. Скласти програму для обчислення значення заданої функції $y(x)$ за введеним з клавіатури значенням змінної x:

$$y = \begin{cases} x\sqrt[3]{ax^3 + \sin x}, & \text{якщо } 0 \le x \le 5; \\ arcsin^3 x, & \text{якщо } x < 0; \\ log_x|x^3 - e^x|, & \text{якщо } x > 5. \end{cases}$$

Це типова задача, що полягає в обчисленні значення функції, заданої окремими формулами на різних проміжках числової осі. З використанням повної форми розгалуження схема її розв'язання могла б бути такою:

- *якщо* $x < 0$ *обчислюємо значення функції* y *за формулою* $y = a \cdot sin^3 x$;
- *інакше, але якщо* $x > 5$ *обчислюємо значення функції за формулою* $y = log_x|x^3 - e^x|$;
- y *решті випадків значення функції обчислюємо за формулою* $y = x\sqrt[3]{ax^3 + \sin x}$.



Попри те, що використання повної форми розгалуження дозволяє виконувати менше перевірок (обчислень логічних виразів), розробники надають перевагу короткій формі. Використання короткої форми розгалуження робить код програми зручнішим для читання, а логіку її роботи більш прозорою.

Тому в програмах для розв'язування цієї задачі використовуватимемо короткі розгалуження.

Алгоритм випливає безпосередньо з формули для обчислення y, його графічне зображення подано на малюнку.

Параметр a, що фігурує в формулах вважатимемо константою, яку задамо в коді програми за допомогою іменованої константи, присвоївши їй деяке довільне значення.

Програма на Pascal.

Для побудови простих умов (логічних виразів) при програмуванні розгалужень використовуються оператори порівняння (відношення), в ***Pascal*** це “=” – дорівнює, “< >” – не рівне, “>” – більше, “<” – менше, “>=” – більше або дорівнює, “<=” – менше або дорівнює.

Для побудови більш складних умов, з простих відношень утворюють логічні вирази за допомогою логічних операторів, що задають логічні операції “і”, “або” чи “не”. До прикладу, в цій задачі виконується перевірка належності $x \in [0;5]$, яка полягає в перевірці одночасного виконання двох умов ***x>=0*** і ***x<=5***.

Для позначення логічних операторів “і”, “або” та “не” в ***Pascal*** використовують спеціальні службові слова ***and***, ***or*** та ***not***. При цьому операнди цих логічних операцій обов’язково беруть в круглі дужки.

Реалізація алгоритму розв’язання задачі 150 мовою ***Pascal*** може бути такою:

```
program Task150;

const  a = 7.7; {параметр a описуємо як константу}

var x,y: real;

begin

  writeln('x = ');

  readln(x);

  if(x >= 0) and (x <= 5) then  { умова xЄ[0;5] }

    y := x*exp(1/3*ln(a*x*x*x + sin(x)));

  if x < 0 then

    y := a*sin(x)*sin(x)*sin(x);

  if x > 5 then

    y := ln(abs(x*x*x-exp(x)))/ln(x);

  writeln('y = ', y:12:5);

end.
```

Програма на C++.

Оператори порівняння (відношення), в ***C++*** задаються знаками “= =” – дорівнює, “!=” – не рівне, “>” – більше, “<” – менше, “>=” – більше або дорівнює, “<=” – менше або дорівнює. Для позначення логічних операторів “і”, “або” та “не” в ***C++*** використовують спеціальні знаки ***”&&”***, ***”||”*** та ***”!”***

відповідно. Логічні оператори в ***C++*** мають найнижчий пріоритет, тому яких-небудь додаткових дужок для їх операндів не потрібно.

Програма-розв'язок задачі 150 мовою C++:

```
#include <iostream>
#include <cmath> // містить елементарні математичні ф-ції
using namespace std;

int main()
{
    //параметр a задаємо як константу
    const double a = 7.7;
    cout<<"x = ";
    double x, y;
    cin>>x;
    if(x >= 0 && x <= 5)
    {
        y = x*pow(a*x*x*x + sin(x),1.0/3);
    }
    if(x < 0)
    {
        y = a*sin(x)*sin(x)*sin(x);
    }
    if(x > 5)
    {
        y = log(abs(x*x*x-exp(x)))/log(x);
    }
    printf("y = %12.5f\n", y);
    return 0;
}
```

Програма на Java.

Побудова логічних виразів в ***Java***, в принципі, аналогічна до ***C++.*** Тобто, оператори відношення задаються знаками “= =” – дорівнює, “!=” – не рівне, “>” – більше, “<” – менше, “>=” – більше або дорівнює, “<=” – менше або дорівнює. Логічні оператори “і”, “або” та “не” – знаками ***″&&″***, ***″||″*** та ***″!″***відповідно і мають найнижчий пріоритет.

Щоправда, для логічної операції “і” можна використовувати як знак ***″&&″***, так і знак ***″&″***, а для “або” як знак ***″||″*** , так і знак ***″|″***. При цьому використання «подвійних» чи «одинарних» логічних операторів на сам результат виразу не впливає. Є відмінності лише у процесі виконання таких операторів: «одинарні» оператори виконуються за всіма правилами логіки, «подвійні» ж дозволяють пропустити зайві перевірки операндів, якщо результат операції очевидний. Наприклад, оператор “і”, позначений знаком ***″&&″***, не буде витрачати час на перевірку істинності другого свого операнда, якщо перший операнд є хибним, бо в такій ситуації вже відомо, що результат операції “і” буде хибним.

Код розв’язування задачі 150 на ***Java***:

```
import java.util.*;
import static java.lang.Math.*; //статичний імпорт дозволяє
// використовувати методи класу Math без посилання на нього
public class Main{
    public static void main(String[] args) {
        final double a = 7.7; //параметр a, константа
        Scanner scanner = new Scanner(System.in);
        System.out.print("x = ");
        double x = scanner.nextDouble();
        double y = 0.0;//змінна повинна бути ініціалізована
        //оскільки компілятор вважає, що програма може не
        // виконати жоден з блоків обчислення в if
        if(x >= 0 && x <= 5){
            y = x*pow(a*x*x*x + sin(x),1.0/3);
        }
        if(x < 0){ y = a*sin(x)*sin(x)*sin(x); }
        if(x > 5){
            y = log(abs(x*x*x-exp(x)))/log(x);
```

```
        }
        System.out.printf("y = %12.5f\n",y);
    }
}
```

Програма на Python.

Окрім згаданих у поясненні до задачі 120 форм умовного оператора в ***Python*** є цікава конструкція, – так званий ***короткий if…else***. Його записують в одну стрічку за такою схемою:

```
<інструкція 1> if <умова> else <інструкція 2>
```

Дія його аналогічна до звичайної повної форми розгалуження: виконується перевірка умови, якщо результат – істина, то виконується інструкція зліва від ***if***, якщо ні, – інструкція справа від ***else***. Якщо такою конструкцією виконувати обчислення значення деякої змінної у вітках розгалуження, то присвоєння записують лише в інструкції зліва від ***if***:

```
<змінна> = <значення 1> if <умова> else <значення 2>
```

Короткий if…else допускає вкладення і його можна використати у коді програми для розв'язування задачі 150. Щоправда, тут маємо доволі громіздкі арифметичні вирази і запис їх обчислення у вигляді двох вкладених один в один коротких ***if…else*** в одній стрічці буде надзвичайно довгим. Саме для такого випадку у ***Python*** передбачено спеціальний розділювач "****" який повідомляє інтерпретатору, що наступна стрічка коду є продовженням попередньої. Застосування цих можливостей пропонуємо переглянути у коді нижче:

```
# використовуємо математичні функції
import math
a = 7.7 #задаємо значення параметра
print('Введіть значення x:')
x = float(input())
y = a * math.sin(x) ** 3 if x < 0 else \
math.log(abs(x ** 3 - math.exp(x)))/math.log(x) if x>5 \
else x*((a * x ** 3 + math.sin(x)) ** (1.0/3))
print("Значення виразу %10.4f" % y)
```

165. Скласти програму для розв'язування системи нерівностей.

$$\begin{cases} x - a \geq 0, \\ x^2 + bx + c < 0. \end{cases}$$

Програма повинна для заданих значень числових коефіцієнтів a, b та c визначати розв'язок у вигляді проміжків на числовій осі. Виконати програму для різних значень коефіцієнтів a, b і c так, щоб продемонструвати її роботу за усіма можливими вітками розгалуження.

Головною запорукою правильної роботи цієї програми є ґрунтовний аналіз можливих варіантів процесу розв'язування нерівності, залежно від значень коефіцієнтів a, b та c.

Перша нерівність має розв'язок $x \geq a$, тобто проміжок $[a; +\infty)$. Розв'язок системи залежить від того яким є розв'язок другої нерівності, та як відносно цього розв'язку розташоване на числовій осі число a.

Тобто програма спочатку повинна обчислити значення дискримінанта D квадратного тричлена в лівій частині другої нерівності та встановити наявність-відсутність його коренів.

- *Якщо дискримінант від'ємний, то квадратний тричлен не має коренів, а оскільки коефіцієнт при x^2 додатний, то друга нерівність матиме розв'язком усю множину дійсних чисел R. Отже розв'язком системи буде проміжок $[a; +\infty)$.*

При D=0 квадратний тричлен в другій нерівності має коренем число $x_0 = \frac{-b}{2}$, а сама нерівність – розв'язок $(-\infty; x_0) \cup (x_0; +\infty)$. Тоді розв'язок системи залежить від взаємного розташування на числовій прямій чисел a та x_0:

- *при $x_0 < a$ розв'язком системи залишиться проміжок $[a; +\infty)$;*
- *при $x_0 = a$ число a не є розв'язком системи і результатом буде $x \in (a; +\infty)$ (те ж що і $x \in (x_0; +\infty)$);*
- *при $x_0 > a$ число x_0 слід виключити з проміжка $[a; +\infty)$, отримаємо результат $x \in [a; x_0) \cup (x_0; +\infty)$*

При $D > 0$ знайдемо корені $x_1 = \frac{-b-\sqrt{D}}{2}$ та $x_2 = \frac{-b+\sqrt{D}}{2}$ квадратного тричлена в другій нерівності (очевидно, що $x_1 < x_2$). Друга нерівність матиме розв'язок $(-\infty; x_1) \cup (x_2; +\infty)$ а для розв'язку системи матимемо такі випадки:

- *при $a < x_1$ розв'язок системи $x \in [a; x_1) \cup (x_2; +\infty)$;*
- *при $x_1 \leq a \leq x_2$ розв'язок системи $x \in (x_2; +\infty)$;*
- *при $a > x_2$ розв'язком системи залишиться проміжок $[a; +\infty)$.*

Враховуючи велику кількість розгалужень в отриманому алгоритмі обійдемося без складання блок-схеми. Взявши за основу наведений вище словесний опис алгоритму розв'язування задачі перейдемо одразу до написання коду програм.

Програма на Pascal.

При реалізації описаного алгоритму на мові ***Pascal*** слід пам'ятати про використання складеного оператора ***begin … end*** у вітках розгалужень, що мають декілька команд та не забувати, що конструкція ***if-then-else*** є одним оператором. Це важливо з огляду на велику кількість розгалужень та досить складну структуру алгоритму. Орієнтовний код програми для розв'язування задачі 165 на ***Pascal***:

```
program Task165;
var a, b, c, d : real;
x0, x1, x2 : real;
begin
  writeln('Введіть коефіцієнти нерівності a,b,c');
  readln( a, b, c);
  d := b * b - 4 * c;
  if d < 0 then {друга нерівність не має коренів}
    writeln('[', a:8:2, ';+inf)')
  else
    {d не менше 0, тобто 0, або більше 0}
    if d = 0 then
      begin
        x0 := -b/2;
        if a < x0 then writeln( 'x Є [', a:8:2, ';',
               x0:8:2, ') U (', x0:8:2, '; +inf)')
        else
          if a = x0 then writeln('x Є (', a:8:2, '; +inf)')
          else writeln('x Є [', a:8:2, '; +inf)')
      end
    else  {вітка, що виконується лише коли d > 0}
      begin
        x1 := (-b-sqrt(d))/2;   x2 := (-b+sqrt(d))/2;
        if a < x1 then
          writeln('x Є [', a:8:2, ';', x1:8:2,
```

```
                ')U(', x2:8:2, ';+inf)')
        else
          if a <= x2 then
              writeln('x Є (', x2:8:2, ';+inf)')
          else writeln('x Є [', a:8:2, ';+inf)')
      end
end.
```

Програма на C++.

При реалізації описаного алгоритму на ***C++*** потрібно слідкувати за коректним використанням блоків коду у кожній вітці розгалуження. На відміну від мови ***Pascal***, де конструкція ***if-then-else*** є одним оператором, повне розгалуження в ***C++*** формально складається з двох окремих операторів: ***if*** та ***else***. Тому вкладені розгалуження повної форми прийнято брати у фігурні дужки.

Програма-розв'язок задачі 165 мовою C++:

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
  double a,b,c,d;
  cout<<"Введіть коефіцієнти нерівності a,b,c\n";
  cin >> a >> b >> c;
  d = b * b - 4 * c;
  if (d < 0)//друга нерівність не має коренів
  {
    cout<< "["<<a<<";+inf)\n";
  }
  else
  {//d не менше 0, тобто 0, або більше 0
    if (d == 0)
    {
```

```
      double x0 = -b/2;
      if (a < x0)
      {
        cout<<"x Є ["<<a<< ";"<< x0 <<
          ")U("<<x0<<";+inf)\n";
      }
      else
      {
        if(a == x0) cout<<"x Є ("<<a<<";+inf)\n";
        else  cout<<"x Є ["<<a<<";+inf)\n";
      }
    }
    else
    {//вітка, що виконується лише коли d > 0
      double x1 = (-b-sqrt(d))/2;
      double x2 = (-b+sqrt(d))/2;
      if (a < x1)
      {
        cout<<"x Є ["<<a<<";" << x1 <<")U("
                    <<x2<<";+inf)\n";
      }
      else{
        if (a <= x2) cout<<"x Є ("<<x2<<";+inf)\n";
        else cout<<"x Є ["<<a<<";+inf)\n";
      }
    }
  }
  return 0;
}
```

Програма на Java.

За винятком об'єктно-орієнтованої структури програми та конструкцій вводу-виводу даних програма на ***Java*** мало чим відрізнятиметься від реалізації цього ж алгоритму на ***C++***:

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

      double a,b,c,d;

      System.out.println("Введіть коефіцієнти нерівності");

      Scanner scanner = new Scanner(System.in);

      a = scanner.nextDouble();

      b = scanner.nextDouble();

      c = scanner.nextDouble();

      d = b * b - 4 * c;

      if (d < 0) { //друга нерівність не має коренів

         System.out.println("x Є [" + a + "; +inf)");

      }

      else {

         if (d == 0) {

            double x0 = -b/2;

            if (a < x0) { System.out.printf(

            "x Є [%.3f; %.3f) U (%.3f; +inf)\n",a, x0, x0);

            }

            else {

               if(a == x0) System.out.printf(

                                    "x Є (%.3f; +inf)\n", a);

             else System.out.printf("x Є [%.3f;+inf)\n",a);

            }

         }

         Else {//вітка, що виконується лише коли d > 0

            double x1 = (-b - Math.sqrt(d))/2;

            double x2 = (-b + Math.sqrt(d))/2;
```

```
            //формули для x1 та x2 обрані так, що x1 < x2
            if (a < x1)  {
               System.out.printf(
            "x Є [%.3f; %.3f) U (%.3f; +inf)\n", a, x1, x2);
            }
            else{
               if (a <= x2)
               System.out.printf("x Є (%.3f; +inf)\n", x2);
               else
              System.out.printf("x Є [%.3f; +inf)\n", a);
            }
         }
      }
   }
}
```

Програма на Python.

З огляду на велику кількість вкладених розгалужень, реалізація нашого алгоритму на ***Python*** також вимагає акуратного запису вкладених блоків коду:

```
import math
print('Введіть коефіцієнти нерівності')
a = float(input())
b = float(input())
c = float(input())
d = b * b - 4 * c;
if d < 0: #друга нерівність не має коренів
    print('x Є [%8.2f; +inf)' % a)
else :
    if d == 0 :
        x0 = -b/2
        if a < x0 :
            print('x Є [%8.3f; %8.3f) U (%8.3f; +inf)' \
            %(a, x0, x0))
        elif a == x0 :
            print('x Є (%8.3f; +inf)' % a)
        else :
            print('x Є [%8.3f; +inf)' % a)
    else: #вітка, що виконується лише при d > 0
```

```
x1 = (-b - math.sqrt(d))/2;
x2 = (-b + math.sqrt(d))/2;
# формули для x1 та x2 обрані так, що x1 < x2
if a < x1 :
    print('x Є [%8.3f; %8.3f) U (%8.3f; +inf)' \
    % (a, x1, x2))
elif a <= x2 :
    print("x Є (%8.3f; +inf)", x2);
else :
    print("x Є [%8.3f; +inf)", a);
```

Розробка програми, яка не має синтаксичних помилок в коді і запускається та програми, яка правильно розв'язує поставлену задачу – різні речі. Для того, щоб переконатися, що ви створили правильну програму для розв'язування задачі, її роботу потрібно перевірити. Такій превірці в сучасній розробці прогрного забезпечення присвячений цілий розділ – ***тестування***.

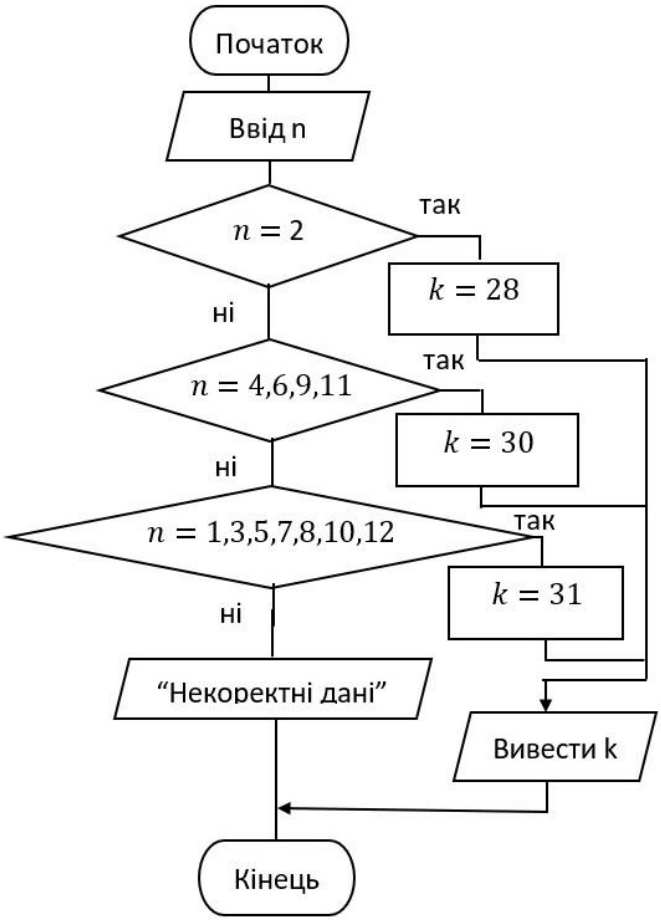
У тестуванні для перевірки якості програми використовують так звані ***тест-кейси***, тобто ситуації при яких результат роботи програми відомий і можна оцінити коректність її роботи. Нижче подано набори вхідних даних, кожен з яких змушує програму виконати одну із запрограмованих віток розгалуження. Отримавши відповідний результат при кожному з поданих наборів значень ***a, b, c***, ми можемо вважати, що наша програма працює коректно.

Значення параметрів			Результат
A	B	c	Роботи
1	1	5	x Є [1.000; +inf)
1	-4	4	x Є [1.000; 2.000)U(2.000; +inf)
2	-4	4	x Є (2.000; +inf)
3	-4	4	x Є [3.000; +inf)
1	-5	6	x Є [1.000; 2.000)U(3.000; +inf)
2	-5	6	x Є (3.000; +inf)
5	-5	6	x Є [5.000; +inf)

Найбільш чутливими до відсутності тестування є програми на ***Python***, оскільки інтерпретатор перевіряє синтаксис лише того коду, який він виконує.

180. За заданим номером місяця (ціле число з відрізка [1; 12]) визначити кількість днів в цьому місяці, за умови, що рік не високосний.

Задачі 176 – 180 розв'язуються за допомогою конструкції розгалуження специфічного виду, де умовою вибору тієї чи іншої вітки виконання програми є перевірка рівності деякого параметра заданому значенню: кожне зі значень параметра визначає окремий спосіб виконання обчислень. Попри те, що реалізувати такий обчислювальний процес можна за допомогою багатократного використання умовного оператора if, більшість технологій програмування надають для нього спеціальну синтаксичну конструкцію, – ***оператор вибору*** або ***перемикач***.



Сам алгоритм розв'язування задачі буде таким:

- *на початку роботи програми користувач вводить число n, що задає порядковий номер місяця, і на основі нього програма робить вибір:*
- *якщо номер місяця 2, то результатом роботи програми має бути відповідь, що місяць має 28 днів*;
- *якщо номер місяця 4, 6, 9 або 11, то місяць має 30 днів;*
- *якщо номер місяця 1, 3, 5, 7, 8, 10 або 12, то місяць має 31 день;*
- *у всіх інших випадках слід повідомити користувача про некоректно заданий ним номер місяця.*

Щодо графічного зображення цієї алгоритмічної конструкції, так само, як і щодо доцільності її використання в кодах програм ІТ-спільнота однозначної позиції не сформувала. Тому, не претендуючи на істину в останній інстанції, подаємо блок-схему розв'язування задачі з використанням однієї з можливих нотацій оператора вибору, – за допоиогою умовних блоків.

Програма на Pascal.

Оператор вибору в ***Pascal*** має такий синтаксис:

```
case (<вираз>) of
   <мітка 1> : <оператор 1>;
   <мітка 2> : <оператор 2>;
   ...
   <мітка n>: <оператор n>;
[   else <оператор else>;]
end;
```

Запис та виконання цього оператора регламентуються такими правилами:

- ✓ *вираз, що використовується в операторі* **case**, *повинен мати цілий інтервальний або перелічувальний тип.*
- ✓ *мітки і вираз повинні бути одного типу, мітки можна задавати одинарними значеннями, переліком значень через кому, діапазоном «від-до» через знак ".. ";*
- ✓ *одне і те ж значення не може міститися в кількох мітках одночасно;*
- ✓ *компілятор обчислює значення виразу, якщо одне зі значень мітки збігається зі значенням виразу, виконується оператор, який слідує за цією міткою і керування передається до наступного оператора;*
- ✓ *якщо жодна мітка не відповідає значенню виразу, виконується інструкція після ключового слова* **else**;
- ✓ *вітка* **else** *може бути відсутня, в цьому випадку, якщо жодна мітка не відповідає значенню виразу, виконнання програми переходить до наступних після case операторів;*
- ✓ *якщо після мітки потрібно записати декілька операторів, то використовують складений оператор* **begin ... end**.

Нижче подано запис алгоритму розв'язання задачі 180 мовою ***Pascal***.

```
program Task180;
var n, k : integer;
begin
  writeln ('Введіть номер місяця:');
  readln(n);
  if (n>0) and (n<13) then
  begin
    case (n) of
      2: k:=28;
```

```
      4, 6, 9, 11: k:=30;
      1, 3, 5, 7, 8, 10, 12: k:=31;
    end;{закінчення оператора case}
    writeln('Кількість днів: ', k);
  end
  else writeln('Нема такого місяця');
end.
```

Програма на C++.

Оператор з багатьма вітками розгалуження в ***C/C++*** називається перемикач, тому що він не обирає вітку для виконання, а просто перемикає виконання на відповідну мітку. Його синтаксис:

```
switch (<вираз>)
{
   case <мітка 1> : <послідовність інструкцій 1>;
   case <мітка 2> : < послідовність інструкцій 2>;
   ...
   case <мітка n>: < послідовність інструкцій n>;
[   deault: <послідовність інструкцій>;]
}
```

Перемикач в ***C/C++*** функціонує за такими правилами:

- ✓ *вираз, що використовується в операторі* **switch**, *повинен мати цілий, перелічувальний або символьний тип.*
- ✓ *мітки і вираз повинні бути одного типу, мітки можна задавати лише одинарними значеннями: літералами, або константами;*
- ✓ *не можна використовувати дві мітки з однаковими значеннями одночасно;*
- ✓ *програма обчислює значення виразу, і переходить до виконання інструкцій, починаючи з місця позначеного міткою з відповідним значенням;*
- ✓ *після переходу до мітки інструкції виконуються послідовно до кінця оператора* **switch**, *таким чином, можуть виконуватися і інструкції наступних міток;*
- ✓ *щоб виконати лише вказану за міткою послідовність інструкцій і програма не заходила в інструкції наступних міток використовують оператор* **break**, *який перериває виконання поточного блоку команд;*
- ✓ *якщо жодна мітка не відповідає значенню виразу, виконується інструкція після ключового слова* **default**;

- ✓ *вітка* **default** *може бути відсутня, тоді, якщо жодна мітка не відповідає значенню виразу, програма преходить до виконння операторів після* **switch**.

Згадані особливості перемикача ***C/C++*** пропонуємо розглянути на прикладі поданого далі коду розв'язування задачі 180:

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Введіть номер місяця:";
    int n, k=0;//значення k виводиться у будь-якому випадку
    //воно залишиться 0 при вводі неправильного номера
    cin >> n;
    switch(n)
    {
        case 2: k=28; break;
        /*мова C/C++ не дозволяє використовувати списки
     значень у пунктах вибору, але ось така конструкція:*/
        case 4: // буде виконано те, що й для
        case 6: // міток 6, 9, 11, тобто код після
        case 9: // мітки 11
        case 11: k=30; break;
        //фактично реалізує      case 4,6,9,11: k=30;
        case 1: case 3: case 5: case 7: case 8: case 10:
        case 12: k=31; break;
        default: cout<<"Нема такого місяця ";
    }
    cout<<"Кількість днів: "<<k<<endl;
    return 0;
}
```

Програма на Java.

Усі правила та зауваження, викладені вище, стосовно структури та принципів функціонування оператора ***switch*** в мові ***C++*** повністю стосуються і перемикача в ***Java***. За винятком хіба однієї особливості, – в ***Java*** перемикач таки вдосконалили, додавши можливість використання у виразах та для міток даних типу ***String***, тобто, фрагментів тексту. Наприклад, в цій задачі програма на ***Java*** могла б запитувати користувача не номер місяця, а його назву та використовувати у відповідних розділах ***case*** назви місяців.

Попри таку можливість ***Java***, залишимо умову задачі 180 без змін та подамо код програми для розв'язування цієї задачі з використанням номера місяця:

```
public class Main{
   public static void main(String[] args) {
      System.out.println("Введіть номер місяця: ");
      int n, k=0;// k виводиться у будь-якому випадку
      //воно залишиться 0 при вводі неправильного номера
      java.util.Scanner scanner =
                    new java.util.Scanner(System.in);
      n = scanner.nextInt();
      switch(n){
         case 2: k=28; break;
         case 4: //перемикач у java дозволяє використуовати
         case 6: //тестові змінні, але списки значень також
         case 9: //не дозволені тому використовуємо групу
         case 11: k=30; break;// case до першого break
         case 1: case 3: case 5:
         case 7: case 8: case 10:
         case 12: k=31; break;
         default: System.out.print("Нема такого місяця ");
      }
      System.out.println("Кількість днів " + k);
   }
}
```

Програма на VB.net.

Для формування більш повної картини розмаїття синтаксичних конструкцій перемикачів (команд вибору) у різних технологіях програмування додамо тут приклад на ***Visual Basic .net***. Оператор вибору (перемикач) тут має наступний синтаксис:

```
Select [ Case ] testexpression
    [ Case expressionlist1
        [ statements1 ] ]
    [ Case expressionlist2
        [ statements2 ] ]
    . . .
    [ Case expressionlistN
        [ statementsN ] ]
    [ Case Else
        [ elsestatements ] ]
End Select
```

Тут вираз ***testexpression*** що використовується для перемикання повинен бути одного з простих типів, хоча допускає використання текстових стрічок (тип String) та навіть дати. Списки ***expressionlist<i>*** в альтернативах ***Case*** можуть містити одне значення, перелік значень через кому, включно з діапазонами значень вигляду ***<початок> To <кінець>***.

Оператор ***Select*** у ***Visual Basic*** виконує саме ***вибір***, а не ***перемикання***, як ***switch*** в ***C++***, – після обчислення значення керуючого виразу програма переходить до виконання блоку інструкцій тієї альтернативи ***Case***, в списку значень якої міститься отримане значення. Після виконання вибраного блоку програма завершує виконання оператора ***Select***, та переходить до виконання наступних після нього операторів. Блок команд при цьому утворюється зі стрічок коду записаних після списку значень вибраного варіанту і до наступного Case.

Цікаво, що інтерпретатор ***Visual Basic*** не контролює входження одного і того ж значення в списки декількох альтернатив ***Case***. В такому випадку програма виконає блок команд після першого списку, в якому знайшлося потрібне значення і цим завершить виконання оператора ***Select***.

Код програми для розв’язування задачі 180 на ***Visual Basic***:

```
Imports System
Module Program
    Sub Main(args As String())
        Dim monthNumber As Integer
        Dim daysCount As Integer
        Dim str As String
        Console.WriteLine("Введіть номер місяця:")
        str = Console.ReadLine()
        ' Вводимо текст з номером місяця і
        ' і конвертуємо його в ціле число
        monthNumber = Convert.ToInt32(str)
        If monthNumber > 0 And monthNumber < 13 Then
            Select Case monthNumber
                Case 2
                    daysCount = 28
                Case 4, 6, 9, 11
                    daysCount = 30
                Case 1, 3, 5, 7, 8, 10, 12
                    daysCount = 31
            End Select
            Console.WriteLine("Місяць має " &
                              daysCount & " днів.")
        Else
            Console.WriteLine("Нема такого місяця")
        End If
    End Sub
End Module
```

Програма на Python.

Упродовж тривалої історії (понад 30 років) існування та розвитку мови ***Python*** такої синтаксичної конструкції як оператор вибору чи перемикач в ній не було. Оператор ***match***, аналогічний до оператора ***switch*** в ***C++*** з'явився лише у версії ***Python 3.10***. В усіх попередніх версіях алгоритми, подібні до нашого, можна було реалізовувати щонайменше трьома способами:

- ✓ *ланцюжком операторів* **if**…**elif**…**else***;*
- ✓ *з використанням спеціальної структури даних* – **словника***;*
- ✓ *засобами об'єктно-орієнтованого програмування.*

Оскільки онлайн-середовище onlinegdb.com, яке ми використовуємо в цьому розділі не підтримує можливості ***Python 3.10,*** реалізуємо наш алгоритм у найпростіший з трьох згаданих альтернативних способів, – за допомогою послідовності умовних операторів. Можливість використовувати оператор ***in***, який перевіряє належність значення до вказаного переліку, робить нашу реалізацію доволі лаконічною:

```
print ('Введіть номер місяця')
m = int(input())
if m==2 :
    n = 28
elif m in (4, 6, 9, 11) :
    n = 30
elif m in (1, 3, 5, 7, 8, 10, 12) :
    n = 31
else :
    print('Неправильно задано номер місяця')
    n = 0
    '''змінну n потрібно оголошувати і ініціалізувати, щоб
    не виникала помилка в підсумковому виводі'''
print('Кількість днів %3d' % n)
```

Розділ 3. Програмування алгоритмів циклічної структури

Циклічні програми з регулярною зміною аргумента. Ітераційні цикли. Вкладені цикли.

181. Дано два цілі числа A та B (A < B). Знайти суму всіх парних цілих чисел від 2A до 4B включно.

182. Для заданого натурального числа N знайти суму

$$\frac{1}{1!}+\frac{2}{2!}+\frac{3}{3!}+\ldots+\frac{N}{N!},$$

за допомогою одного циклу.

183. Для заданого натурального числа N і дійсного числа X знайти суму

$$\frac{1}{X}+\frac{1}{X^2}+\frac{1}{X^3}+\ldots+\frac{1}{X^N},$$

використовуючи лише один цикл.

184. Дано два цілі числа A і B (A < B). Знайти добуток всіх цілих чисел від A до B включно.

185. Вивести таблицю квадратів усіх натуральних чисел, починаючи від заданого натурального K до заданого натурального N (N>K)

186. Дано ціле число N > 0. Послідовність натуральних чиселA_k, задана за допомогою формул $A_1 = 3$, $A_2 = 7$, $A_k = 2A_{k-1} + 3A_{k-2}$, k = 3, 4, … . Вивести елементи A_1,A_2,…,A_N.

187. Дано два цілі числа A и B (A < B). Знайти суму квадратів усіх цілих чисел від A до B включно.

188. Для заданого додатного дійсного числа X знайти суму

$$\frac{X}{1}-\frac{X}{2}+\frac{X}{3}-\frac{X}{4}+\ldots+\frac{X}{N},$$

де N найближче до X натуральне число, менше за X .

189. Задано ціле число N > 0. Знайти суму

$$N^2+(N+1)^2+(N+2)^2+\ldots+(2N)^2$$

190. Задано ціле число N (N > 0). Знайти добуток

$$1.1\cdot 1.2\cdot 1.3\cdot \ldots\ (1+0.1*N).$$

191. Задано ціле число N (N > 0). Знайти квадрат даного числа, використовуючи для його обчислення формулу:

$$N^2=1+3+5+\ldots+(2N+1).$$

Після додавання до суми наступного доданка виводити поточне значення.

192. Задано натуральне число N. Послідовність дійсних чисел A_k, задана за допомогою формул $A_0 = 2$, $A_k = 2 + \frac{3}{A_{k-1}}$, k = 1, 2, … . Вивести елементи $A_1, A_2, \ldots, A_N$.

193. Задано натуральне число N. Послідовність дійсних чиселA_k, задана за допомогою формул $A_0 = 1$, $A_k = \frac{A_{k-1}+1}{k}$, k = 1, 2, … . Вивести елементи $A_1, A_2, \ldots, A_N$.

194. Задано дійсне число A і натуральне число N. Використовуючи один цикл, знайти суму

$$1 + A + A^2 + A^3 + \ldots + A^N.$$

195. Задане дійсне число A та натуральне число N. Використовуючи один цикл і не використовуючи умовного оператора, знайти суму

$$1 - A + A^2 - A^3 + \ldots + (-1)^N A^N.$$

196. Вивести на екран таблицю значень функції $y = \sin x + \sqrt{x}$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n = b$, $x_k = x_{k-1} + h,\ h = const$. Межі відрізка $a \geq 0, b > a$ та кількість точок n ввести з клавіатури, крок h визначити за довжиною відрізка та кількістю точок. Знайти середнє арифметичне обчислених значень функції.

197. Побудувати таблицю значень функції $y = e^x + x^p$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n \leq b$, $x_k = x_{k-1} + h,\ h = const$. Межі відрізка a, b та крок h ввести з клавіатури, кількість точок n визначити за довжиною відрізка та кроком. Параметр p задати константою в коді програми. Обчислити суму отриманих значень функції.

198. Побудувати таблицю значень функції $y = \sin^3 x$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n \leq b$, $x_k = x_{k-1} + h,\ h = const$. Межі відрізка $a, b > a$ та кількість точок n ввести з клавіатури, крок h визначити за довжиною відрізка та кількістю точок. Знайти добуток обчислених значень функції.

199. Вивести на екран таблицю значень функції $y = p^x + x^3$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n = b,\ x_k = x_{k-1} + h,\ h = const$. Межі відрізка a, b та крок h ввести з клавіатури, кількість точок n визначити за довжиною відрізка та кроком. Параметр p задати як константу в коді програми. Обчислити суму натуральних логарифмів від отриманих значень функції.

200. Побудувати таблицю значень функції $y = ln\, x + \frac{1}{\sqrt{x}}$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n = b,\ x_k = x_{k-1} + h,\ h = const$. Межі відрізка $a > 0, b > a$ та кількість точок n ввести з клавіатури, крок h визначити за довжиною відрізка та кількістю точок. Знайти середнє арифметичне кубів обчислених значень функції.

201. Побудувати таблицю значень функції $y = \frac{p^x}{x^p}$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n \leq b,\ x_k = x_{k-1} + h,$ $h = const$. Межі відрізка a, b та крок h ввести з клавіатури, кількість точок n визначити за довжиною відрізка та кроком. Параметр $p > 1$ задати константою в коді програми. Обчислити суму кубів отриманих значень функції.

202. Вивести на екран таблицю значень функції $y = x + cos^3 x$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n = b,$ $x_k = x_{k-1} + h,\ h = const$. Межі відрізка $a, b > a$ та кількість точок n ввести з клавіатури, крок h визначити за довжиною відрізка та кількістю точок. Знайти суму квадратів обчислених значень функції.

203. Вивести на екран таблицю значень функції $y = e^{sinx}$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n = b,\ x_k = x_{k-1} + h,\ h = const$. Межі відрізка $a, b > a$ та кількість точок n ввести з клавіатури, крок h визначити за довжиною відрізка та кількістю точок. Знайти добуток кубів обчислених значень функції.

204. Побудувати таблицю значень функції $y = e^x - px^3$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n \leq b,\ x_k = x_{k-1} + h,\ h = const$. Межі відрізка a, b та крок h ввести з клавіатури, кількість точок n визначити за довжиною відрізка та кроком. Параметр p задати константою в коді програми. Обчислити середнє арифметичне отриманих значень функції.

205. Побудувати таблицю значень функції $y = \sin^3 x + \cos^2 x$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n \leq b,$ $x_k = x_{k-1} + h,\ h = const$. Межі відрізка $a, b > a$ та кількість точок n ввести з клавіатури, крок h визначити за довжиною відрізка та кількістю точок. Знайти добуток кубів обчислених значень функції.

206. Вивести на екран таблицю значень функції $y = \sqrt[3]{p + x^2}$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n = b,$ $x_k = x_{k-1} + h,\ h = const$. Межі відрізка a, b та крок h ввести з клавіатури, кількість точок n визначити за довжиною відрізка та кроком. Параметр p задати як константу в коді програми. Обчислити суму отриманих значень функції.

207. Побудувати таблицю значень функції $y = \frac{1+x^3}{\sqrt{x}}$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n = b,\ x_k = x_{k-1} + h,\ h = const$. Межі відрізка $a > 0, b > a$ та кількість точок n ввести з клавіатури, крок h визначити за довжиною відрізка та кількістю точок. Знайти середнє арифметичне квадратів обчислених значень функції.

208. Побудувати таблицю значень функції $y = \frac{x^3-7}{x^2+5x+9}$ у рівномірно розподілених на відрізку $[a;\ b]$ точках $x_0 = a < x_1 < \cdots < x_n \leq b,\ x_k = x_{k-1} + h,\ h = const$. Межі відрізка a, b та крок h ввести з клавіатури, кількість

точок n визначити за довжиною відрізка та кроком. Обчислити добуток отриманих значень функції.

209. Вивести на екран таблицю значень функції $y = x^5 + lg\, x$ у рівномірно розподілених на числовій осі точках $x_0 < x_1 < \cdots < x_n,\ x_k = x_{k-1} + h,\ h = const$. Початкову точку x_0, крок h та кількість точок n ввести з клавіатури. Знайти середнє арифметичне обчислених значень функції.

210. Вивести на екран таблицю значень функції $y = e^x - x^3$ у рівномірно розподілених на числовій осі точках $x_0 < x_1 < \cdots < x_n,\ x_k = x_{k-1} + h,\ h = const$. Початкову точку x_0, крок h та кількість точок n ввести з клавіатури. Знайти добуток обчислених значень функції.

211. Задано цілі додатні числа m і n ($m > n$). Визначити частку та остачу від ділення націло m на n, використовуючи лише операції віднімання та додавання.

212. Користувач вводить два цілі додатні числа m і n. Перевірити, чи число m є деяким цілим степенем числа n, чи ні.

213. Стартова сума депозиту у банку складає S_0 грн. Кожного місяця банк нараховує власнику $p\%$ від поточного розміру суми депозиту. Окрім цього, кожного місяця власник сам поповнює депозит на фіксовану суму M грн. Визначити через скільки місяців сума депозиту досягне потрібної власнику величини S грн.

214. Задано натуральне число m. Не використовуючи функцію для знаходження кореня квадратного від числа, знайти найменше натуральне число n, квадрат якого перевищує m ($n^2 > m$).

215. Задано натуральне число m. Використовуючи операції ділення націло та остачі від ділення вивести всі його цифри у зворотному порядку а також визначити кількість його цифр.

216. Користувач вводить два натуральні числа a та b. Визначити найбільший натуральний степінь числа a, що не перевищує b, тобто визначити масимальний показник $n \in N$, для якого $a^n \leq$b.

217. Послідовність цілих чисел задається рекурентними співвідношеннями:

$$A_1 = 3, A_2 = 7, A_n = 3A_{n-1} - 2A_{n-2}, n = 3, 4, \ldots$$

Скласти програму для знаходження порядкового номера найбільшого члена послідовності $\{A_n\}$, що не перевищує задане число m.

218. Скласти програму, яка з'ясовує, чи містить задане натуральне число $n \in N$ задану цифру p.

219. Для заданого дійсного числа $A \in [1; 20]$ знайти найменше натуральне число n, для якого сума $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ буде більшою за A. Вивести цю суму.

220. Визначити при якому найменшому значенні n сума n послідовних чисел Фібоначчі $\sum_{k=1}^{n} F_k$ перевищить за величиною задане натуральне число m.

Послідовність чисел Фібоначчі задається співвідношеннями:

$$F_1 = 1, F_2 = 1, F_n = F_{n-1} + F_{n-2}, n = 3, 4, \ldots$$

221. Для заданих натуральних чисел a та b визначити найбільший спільний дільник (НСД), використовуючи алгоритм Евкліда:

$$\text{НСД}(m, n) = \text{НСД}(n, m \bmod n); \text{НСД}(m, 0) = m,$$

де $m \bmod n$ – остача від ділення m на n.

222. Послідовність дійсних чисел задається співвідношеннями:

$$a_1 = 2, a_k = 2 + \frac{1}{a_{k-1}}, k = 2, 3, \ldots$$

Для заданої величини $\varepsilon \in (0; 1)$ визначити перший із номерів k, для якого:

$$|a_k - a_{k-1}| < \varepsilon.$$

223. Послідовність цілих чисел задається рекурентними співвідношеннями:

$$A_1 = 2, A_2 = 3, A_n = 3A_{n-1} + A_{n-2}, n = 3, 4, \ldots$$

Скласти програму для знаходження порядкового номера n, при якому сума n перших членів послідовності $\{A_n\}$ досягне (або перевищить) задану величину m.

224. Задано натуральне число m. Використовуючи операції ділення націло та остачі від ділення, визначити його першу цифру та суму усіх його цифр.

225. Перевірити, чи задане натуральне число m є числом Фібоначчі. Якщо так то вивести його порядковий номер n. Якщо ж ні, то вивести найближче до m число Фібоначчі, більше за m.

Послідовність чисел Фібоначчі задається співвідношеннями:

$$F_1 = 1, F_2 = 1, F_n = F_{n-1} + F_{n-2}, n = 3, 4, \ldots$$

226-240. Скласти програму для обчислення значення елементарної функції в заданій точці за допомогою степеневого ряду з точністю до члена, меншого за 0.00001. Визначити кількість доданків. Порівняти отримане значення суми зі значенням вказаної функції, обчисленим за допомогою бібліотечних функцій.

226. $e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$;

227. $sin^3 x = \frac{1}{4} \cdot \sum_{k=1}^{\infty} (-1)^{k+1} * \frac{3^{2k+1}-3}{(2k+1)!} * x^{2k+1}$;

228. $\ln x = 2\sum_{k=1}^{\infty} \frac{1}{2k-1}\left(\frac{x-1}{x+1}\right)^{2k-1}$;

229. $e^{-x^2} = \sum_{k=0}^{\infty}(-1)^k \frac{x^{2k}}{k!}$;

230. $\cos^2 x = 1 - \sum_{k=1}^{\infty}(-1)^{k+1} \frac{2^{2k-1}x^{2k}}{(2k)!}$

231. $ln\frac{1+x}{1-x} = 2\sum_{k=0}^{\infty} \frac{x^{2k+1}}{2k+1}$;

232. $(1+x)^{-\frac{1}{2}} = 1 - \frac{1}{2}x + \frac{1*3}{2*4}x^2 - \frac{1*3*5}{2*4*6}x^3 + \cdots$;

233. $arcctg\, x = \frac{\pi}{2} - \sum_{k=0}^{\infty}(-1)^k \frac{x^{2k}}{2k+1}$;

234. $\cos^3 x = \frac{1}{4}\sum_{k=0}^{\infty}(-1)^k \frac{(3^{2k}+3)}{(2k)!}x^{2k}$;

235. $sh\, x = \sum_{k=0}^{\infty} \frac{x^{2k+1}}{(2k+1)!}$;

236. $\sin x = \sum_{k=0}^{\infty}(-1)^k \frac{x^{2k+1}}{(2k+1)!}$;

237. $(1+x)^{-\frac{1}{3}} = 1 - \frac{1}{3}x + \frac{1*4}{3*6}x^2 - \frac{1*4*7}{3*6*9}x^3 + \cdots$;

238. $\sin^2 x = \sum_{k=1}^{\infty}(-1)^{k+1} \frac{2^{2k-1}x^{2k}}{(2k)!}$;

239. $ch\, x = \sum_{k=0}^{\infty} \frac{x^{2k}}{(2k)!}$;

240. $(1+x)^{\frac{1}{4}} = 1 + \frac{1}{4}x - \frac{1*3}{4*8}x^2 + \frac{1*3*7}{4*8*12}x^3 - \frac{1*3*7*11}{4*8*12*16}x^4 + \cdots$.

241-255. Побудувати таблицю значень заданої функції двох змінних у заданому користувачем прямокутнику $[a;b]\times[c;d]$. Крок зміни значень змінних $x \in [a;b]$ та $y \in [c;d]$ вибрати таким чином, щоб на кожному з відрізків розміщувалося по 8 точок. Результат обчислень вивести на екран у вигляді прямокутної таблиці, в якій по горизонталі вказані різні значення змінної x, по вертикалі – значення змінної y, а на перехресті стовпця зі значенням змінної x та рядка зі значенням змінної y – відповідне значення функції.

241. $u = 2x^2 + 3y^2$;

242. $u = x^3y + 3y^3x$;

243. $u = sin\, x + cos\, y$;

244. $u = 5\, lg(x+y)$;

245. $u = \sqrt{x^2 + 4y^2}$;

246. $u = x^3 + y^2x$;

247. $u = x^3 + y^2x;$ **248.** $u = 10\,sin(x - y);$

249. $u = 5\,sin(x + y);$ **250.** $u = xy + 3y^3x;$

251. $u = 2x^3 + 3y^2;$ **252.** $u = 5\,tg(x + y);$

253. $u = sin\,x\,y + cos\frac{x}{y};$ **254.** $u = 2x^3 - 3y^2;$

255. $u = \sqrt{2xy + 4y^3}.$

256-270. Написати програму для обчислення суми/добутку. Кількості доданків/множників s та p користувач вводить з клавіатури під час виконання програми:

256. $\sum_{k=1}^{s}\prod_{n=1}^{p}\frac{3k}{2n};$ **257.** $\sum_{n=1}^{p}\sum_{k=1}^{s}\frac{(k+n)^2}{2};$

258. $\prod_{n=1}^{p}\sum_{k=1}^{s}\frac{3k+2n}{7};$ **259.** $\prod_{k=1}^{s}\prod_{n=1}^{p}\frac{k+2n}{3};$

260. $\sum_{k=1}^{p}\sum_{n=1}^{s}\frac{(k-n)^2}{3};$ **261.** $\sum_{k=1}^{s}\prod_{n=1}^{p}\frac{5+n}{2k+3};$

262. $\prod_{k=1}^{s}\sum_{n=1}^{p}\frac{k-2n}{3};$ **263.** $\prod_{k=1}^{p}\sum_{n=1}^{s}\frac{3k-n}{7};$

264. $\sum_{k=1}^{s}\prod_{n=1}^{p}(k+n);$ **265.** $\prod_{n=1}^{s}\prod_{k=1}^{p}\frac{2n+3}{3k+2};$

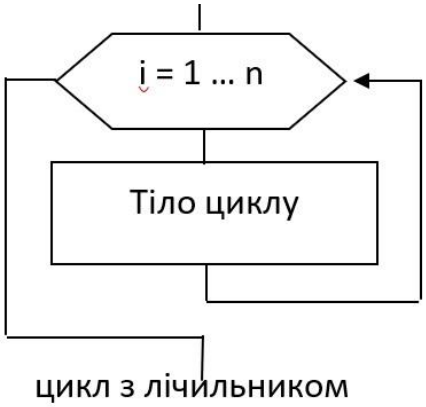
266. $\sum_{k=1}^{s}\sum_{n=1}^{p}(2k+n)^2;$ **267.** $\prod_{k=1}^{p}\sum_{n=1}^{s}\frac{k+2n}{7k};$

268. $\prod_{k=1}^{p}\sum_{n=1}^{s}\frac{3n-k}{n};$ **269.** $\sum_{n=1}^{s}\sum_{k=1}^{p}(k-n)^3;$

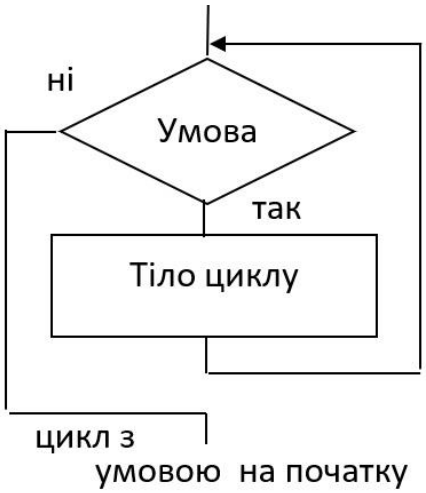
270. $\sum_{k=1}^{s}\prod_{n=1}^{p}\frac{3}{k+n}.$

Практикум до розділу 3. Програмування алгоритмів циклічної структури

Умови задач 181-270 передбачають використання ще однієї алгоритмічної конструкції – ***циклу***. Цикл це повторення деякої послідовності інструкцій алгоритму. Кількість повторів може задаватися явно, або регламентуватися виконанням деякої умови. Послідовність команд, які повторюються в циклі прийнято називати ***тілом циклу***, а одне виконання ***тіла циклу***, зазвичай, називають ***ітерацією***. Цикл разом з розгалуженням та лінійною алгоритмічною конструкцією, розглянутими у попередніх розділах, складають базовий набір конструкцій для побудови алгоритмів будь-якої складності.



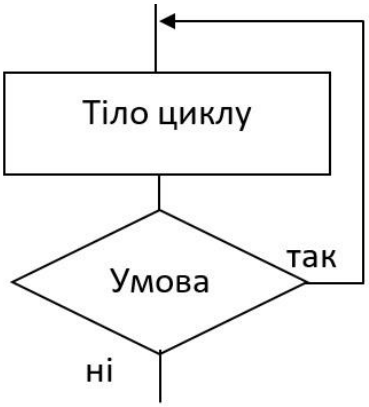
цикл з лічильником



цикл з умовою на початку

Залежно від способу організації ітерацій виділяють три базові конструкції циклу:

- ✓ *цикл з лічильником (параметром) виконує послідовність команд для кожного значення які пробігає деяка змінна-лічильник;*
- ✓ *цикл з умовою на початку перевіряє потребу виконання тіла циклу перед кожною ітерацією, якщо умова виконання хибна вже на початку циклу, то тіло не виконується жодного разу;*
- ✓ *цикл з умовою після тіла виконує перевірку умови після кожної ітерації і вирішує чи потрібно повернутися і повторити інструкції ще раз. Тут команди тіла циклу виконаються обов'язково хоча б один раз.*

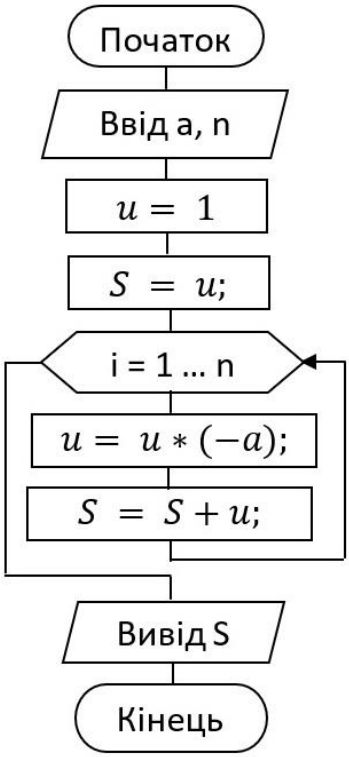


цикл з умовою після тіла

Такий поділ суто теоретичним, реалізації зазначених концепцій в різних технологіях програмування, як можна бачити далі, доволі розмаїті. Окремі синтаксичні конструкції, як, наприклад, оператор **`for`** в С-подібних мовах мають засоби для реалізації одразу двох концепцій (цикл з лічильником та цикл з умовою на початку). З іншого боку, цикли з умовами за рахунок допоміжних змінних та контролю їх значень дозволяють реалізовувати цикли з лічильником. І, взагалі, до кожної практичної задачі, що розв'язується за допомогою циклу, можна застосувати будь-яку із зазначених форм циклу. Розглянемо ці конструкції на конкретних прикладах.

195. Задане дійсне число A та натуральне число N. Використовуючи один цикл і не використовуючи умовного оператора, знайти суму

$$1 - A + A^2 - A^3 + \ldots + (-1)^N A^N.$$



Умови задач 181-210, зокрема і задачі табулювання функцій передбачають відому наперед кількість ітерацій циклу. Ця кількість або задана явно умовою задачі в якості вхідного параметра (як у випадку цієї задачі), або обчислюється через інші вхідні дані. Тобто, тут оптимальним є застосування конструкцій циклів з лічильником.

Зазвичай, в програмах для обчислення суми запроваджують допоміжну змінну для зберігання поточного доданка, і в тілі циклу додають її значення до суми. В нашій задачі шукаємо суму доданків з почерговою зміною знаку, в початківців така ситуація часто породжує ідеї на кшталт перевірки парності номера доданка і вибору відповідного знаку через умовний оператор. В цьому немає жодної потреби, про що і наголошується в умові задачі. Оскільки кожен наступний доданок відрізняється від попереднього, окрім протилежного знака, на одиницю більшим степенем числа A, то для обчислення чергового доданка

$$(-1)^k A^k = (-1) \cdot (-1)^{k-1} \cdot A \cdot A^{k-1} = -A \cdot (-1)^{k-1} A^{k-1}$$

достатньо домножити попередній доданок на $-A$. Початкове значення змінної (в алгоритмі позначено u) для поточного доданка за умовою задачі дорівнює 1. Подібний підхід застосовується також і в задачах 225 - 240 про обчислення значень елементарних функцій за допомогою степеневих рядів.

Алгоритм розв'язування задачі буде таким:

- *спочатку програми отримує від користувача задає число a, для якого обчислюється сума, та кількість доданків n;*
- *програма задає початкові значення суми, та поточного доданка: враховуючи, що сума в умові задачі має $n + 1$ доданок, додамо до суми перший з них $u = 1$ перед циклом, а наступні знаходитимемо і додаватимемо в тілі циклу;*
- *n разів програма повторює виконання таких дій: знаходить значення наступного доданка, домноживши попередній на -a та додає його до суми;*
- *після завершення n ітерацій циклу обчислену суму програма виводить на екран і завершує свою роботу.*

Графічне зображення сформульованого вище словесного опису алгоритму подано у вигляді блок-схеми на малюнку.

Програма на Pascal.

Для розробки та відлагодження Pascal-програм продовжуємо користуватися онлайн-IDE, наприклад, https://rextester.com/l/pascal_online_compiler, чи ideone.com, або onlinegdb.com, як у попередніх розділах.

Pascal має синтаксичні конструкції для реалізації усіх трьох форм циклу, зокрема для циклу з лічильником використовуються конструкції:

```
for <параметр>:= <поч. знач.> to <кінц. знач.> do <команда>
```

та

```
for <параметр>:= <поч. зн.> downto <кінц. зн.> do <команда>
```

Основні правила використання цієї конструкції:

- ✓ *команда виконується для кожного із значень параметра, яких він набуває, пробігаючи відрізок від початкового до кінцевого значення включно;*
- ✓ *параметр може бути лише цілого чи перелічуваного типу;*
- ✓ *зміна значення параметра може відбуватися лише з кроком 1, якщо в записі оператора використовується ключове слово* **to**, *та -1, – якщо використовується ключове слово* **downto**;
- ✓ *команда в тілі циклу може бути тільки одна, якщо потрібно в тілі циклу виконувати декілька інструкцій, то використовується складений оператор* **begin**…**end.**

Запис алгоритму розв’язання задачі 195 мовою ***Pascal***:

```
program Task195;

var a, s, u: real; n, i : integer;

begin

  u := 1;

  writeln('Введіть значення a = ');   readln(a);

  writeln('Введіть значення n = ');   readln(n);

  s := u;

  for i:=1 to n do

  begin

    u := u * -a; s := s + u;

  end;

  writeln('Шукана сума s = ', s:10:3);

end.
```

Програма на С++.

Мови програмування, що використовують ***С***- подібний синтаксис коду (C++, Java, PHP, C#, тощо) також наділені можливостями для реалізації усіх варіантів циклічних алгоритмів. Зокрема, для реалізації циклу з лічильником тут використовується конструкція:

```
for ( <ініціалізація >; <умова>; <дії після ітерації>)
{
    <команди>
}
```

Така конструкція охоплює значно більше можливостей, аніж, скажімо, оператор ***for*** в ***Pascal***. Оскільки кожна з трьох частин заголовку цього оператора є необов'язковою, то реалізувавши лише середню з них можна отримати цикл ***for*** з умовою на початку:

```
for ( ; <умова>; )
{
    <команди тіла циклу>
}
```

Основні правила запису та функціонування циклу ***for*** в ***C/C++***:

- ✓ *перша частина заголовку оператора* **for** *(ініціалізація) оголошує дії, що виконуються перед початком циклу, тут можна оголошувати нові змінні, але їх область існування буде обмежена тілом циклу;*
- ✓ *ініціалізація циклу може містити декілька команд, тоді їх записують підряд, розділяючи комами;*
- ✓ *у другій частині заголовку* **for** *записують умову, яка перевіряється перед кожним виконанням тіла циклу (ітерацією), якщо результат перевірки хибний, то ітерація не виконується і програма переходить до наступного після циклу оператора;*
- ✓ *третя частина заголовку описує дії, що потрібно виконати після завершення окремої ітерації, якщо їх декілька, то їх записують підряд, розділяючи комами;*
- ✓ *кожна з трьох частин заголовку за потреби може бути пропущена, зокрема, конструкція* **for(; ;){<інструкції>}** *є синтаксично коректною і виконуватиме "вічний цикл";*
- ✓ *тіло циклу, що складається з однієї команди можна записувати після заголовку без фігурних дужок.*

Перелічені вище правила декларують різні можливі способи застосування оператора ***for***. Реалізація простого циклу з лічильником, що повторює тіло циклу задану кількість разів за допомогою оператора ***for*** набагато простіша,

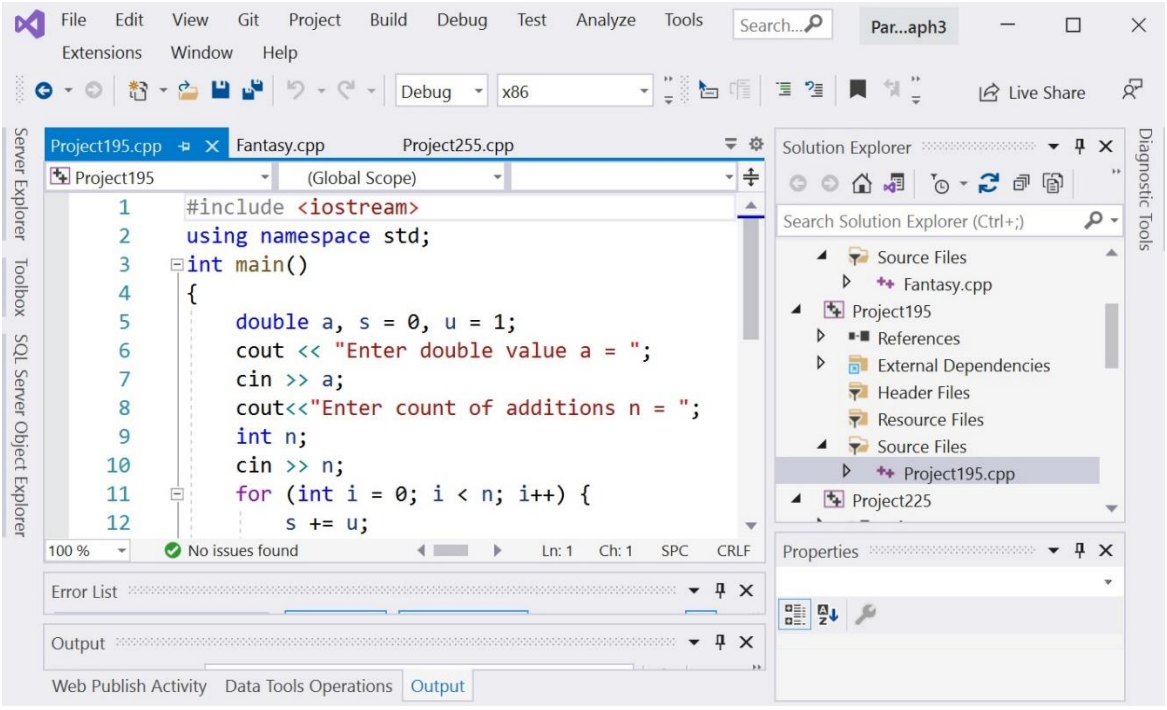
пропонуємо розглянути її безпосередньо в прикладі програми для розв'язування задачі 195:

```
#include <iostream>

using namespace std;

int main()

{

    double a, u = 1, s = u;

    cout << "Enter double value a = ";

    cin >> a;

    cout<<"Enter integer value n = ";

    int n;

    cin >> n;

    //цикл повторює виконання тіла n разів (i=0,1,…,n-1)

    for (int i = 0; i < n; i++) {

        u *= -a;  s += u;

    }

    printf("Summ s = %.3f", s);

}
```

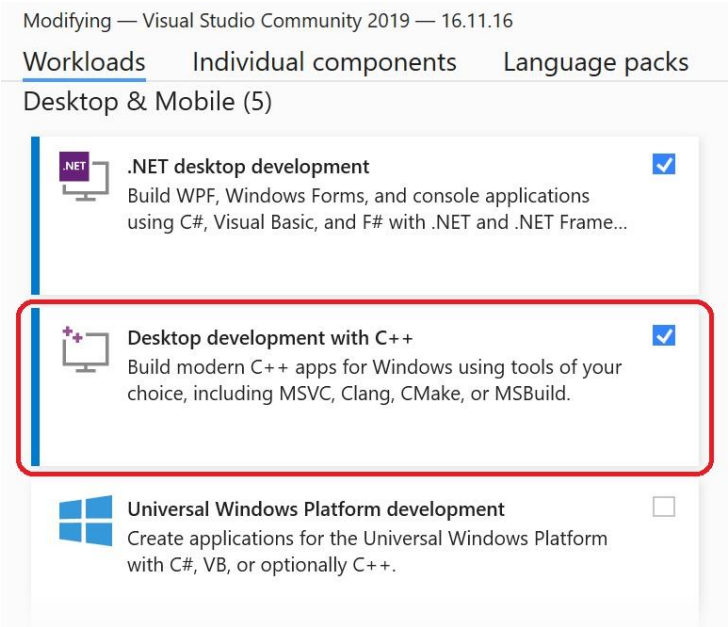
Використання онлайн-компіляторів, попри очевидні переваги, не підходить для професійної розробки. Стаціонарні ***IDE*** мають більш потужний набір інструментів, що допомагають розробникові швидко писати якісний код, підключати до програм потрібні бібліотеки, збирати з них готові програмні продукти, координувати свою роботу з іншими розробниками в колективних проєктах. Повноцінні середовища розробки є незамінним інструментом IT-фахівців. В цьому та наступних розділах розглянемо деякі з популярних інтегрованих середовищ розробки програм на мовах програмування, які тут використовуються.

Для розробки програм на ***C++*** тут і далі використовуватимемо IDE ***MS Visual Studio*** від компанії ***Microsoft***. Для професійного (комерційного) використання цього програмного продукту необхідно придбати ліцензію, але для навчальних цілей Microsoft розповсюджує «полегшену» версію – ***Community Edition***. Для використання Microsoft Visual Studio Community Edition достатньо мати лише ***обліковий запис Microsoft***, його можна зареєструвати безкоштовно, або скористатися раніше створеним, наприклад, корпоративним Microsoft-профілем, що надають студентам навчальні заклади.

В поточному моменті корпорація Microsoft надає дві версій цього програмного продукту ***MS Visual Studio 2019*** та ***MS Visual Studio 2022***. Поданий далі огляд зроблено на основі ***MS Visual Studio 2019.***



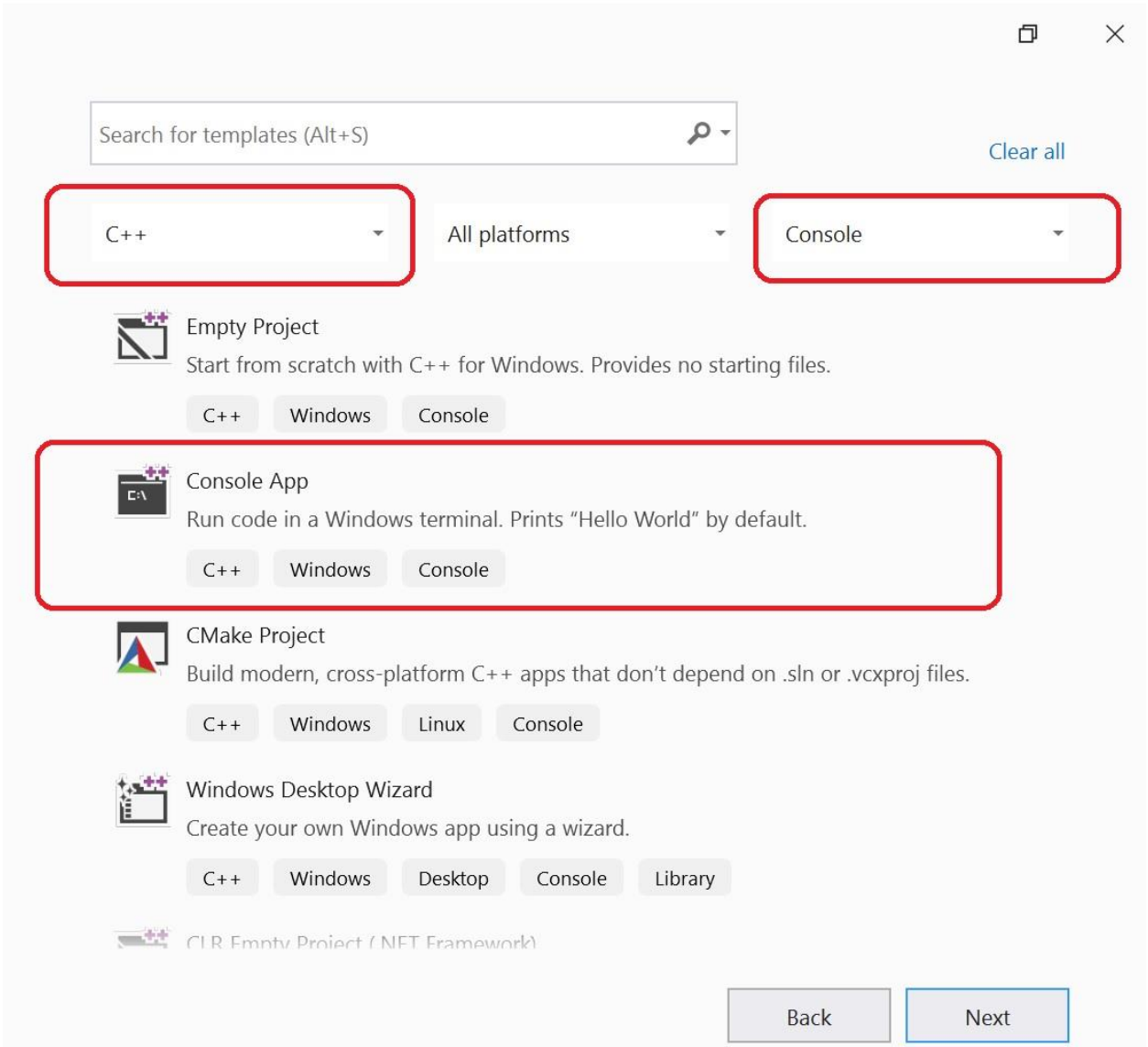
MS Visual Studio підтримує розробку програм декількома мовами, в тому числі ***C++***, ***C#*** та ***Python***. Керування засобами для розробки, а також оновленнями самого програмного продукту здійснюється за допомогою спеціальної програми ***Visual Studio Installer***, що встановлюється разом із ***Visual Studio***. Ця програма дозволяє встановлювати так звані «робочі навантаження», що містять інструменти для використання тієї, чи іншої технології. Для того, щоб працювати з класичними проєктами на C++ потрібно встановити Desktop development with C++.



Visual Studio структурує код у вигляді проектів. ***Проєкт*** це, по суті, окрема програма, яку можна виконати. Проєкт може складатися з декількох файлів з кодом, усі наші попередні приклади на C++ (як і більшість наступних) містили один файл програмного коду. MS Visual Studio зберігає проєкти в так званих «рішеннях», рішення може містити один або декілька проєктів.

Найшвидший спосіб створити та виконати просту програму на C++ у MS Visual Studio 2019 буде приблизно таким:

- *запустити Visual Studio, якщо потрібно увійти з Visual Studio у свій профіль Microsoft;*
- *з меню програми виконати команду File=>New=>Project, або обрати пункт Create a new project на стартовому екрані програми;*
- *у полях для пошуку шаблону задати C++ та Console, як показано на малюнку, обрати тип проєкту Console App та натиснути Next;*



- *на наступному кроці задати назву проєкту та папку на диску, в якій він буде зберігатися і натиснути Create;*
- *після успішного створення проєкту Visual Studio створить файл з заготовкою програми;*
- *згенерований код можна замінити на власний, наприклад, код розв'язування задачі 195, поданий вище.*

Програма на Java.

Синтаксичні конструкції циклів у ***Java*** практично нічим не відрізняються від аналогічних конструкцій ***C++.*** Зіславшись на описану вище концепцію побудови циклу з лічильником в ***C++***, та врахувавши особливості вводу-виводу даних і структуру консольної програми на ***Java*** отримаємо таку реалізацію алгоритму розв'язування задачі 195:

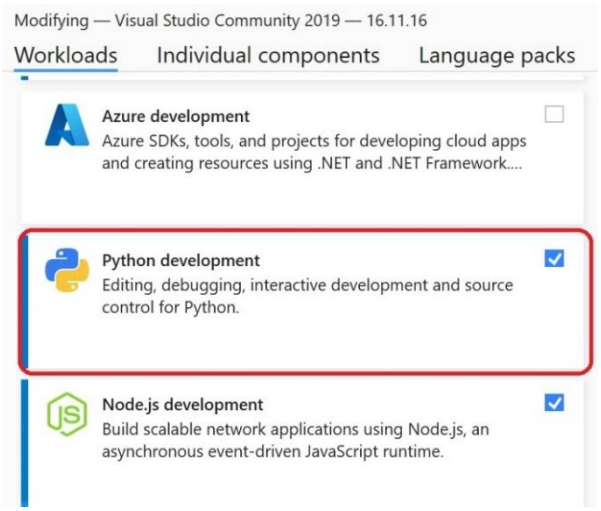
```
import java.util.Scanner;

public class Task195 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        double a, u = 1, s = u;

        System.out.println("Введіть значення a");

        a = scanner.nextDouble();

        System.out.println("Введіть значення n");

        int n = scanner.nextInt();

        for (int i = 0; i < n; i++) {

            u *= -a;

            s += u;

        }

        System.out.printf("Значення суми s = %.3f", s);

    }

}
```

Програма на Python.

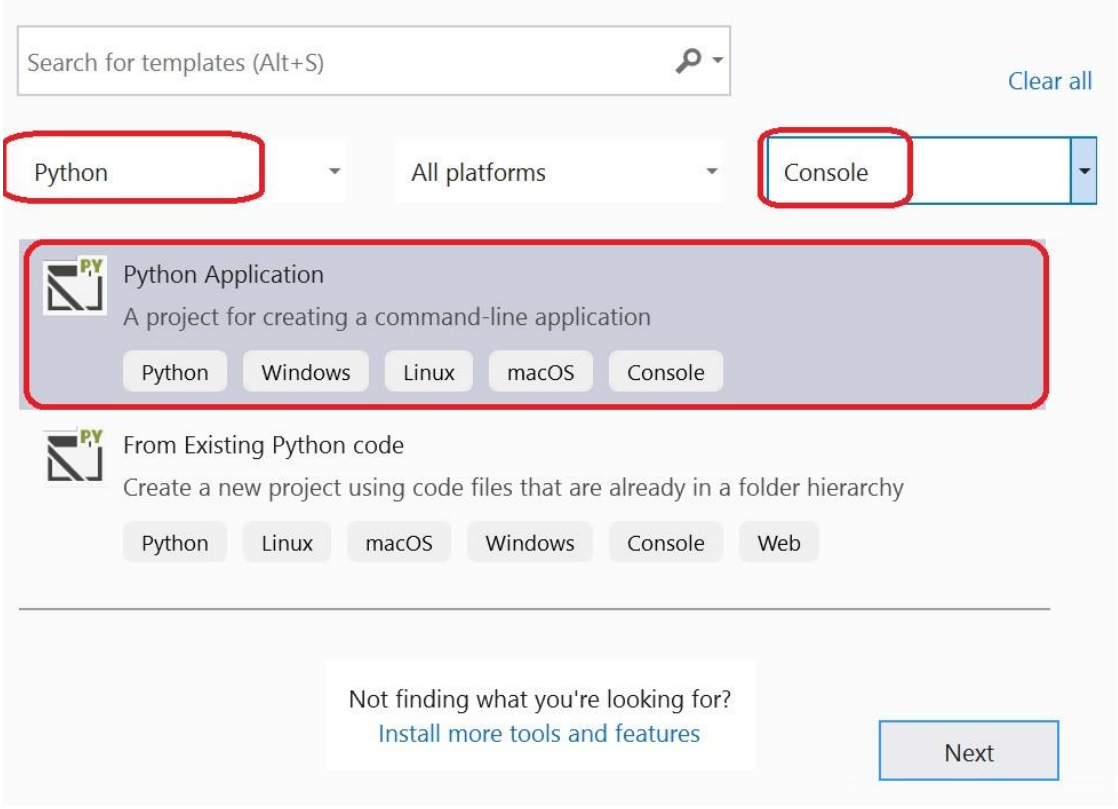
Спочатку розглянемо можливість використання до розробки на ***Python*** інтегрованого середовища ***Microsoft Visual Studio***, рекомендованого вище (див. пункт Програма на C++ до цієї ж задачі) для програмування на C++.

Мова ***Python*** не належить мов платформи ***.Net***, як ***C++***. Тому для виконання програм на ***Python*** у ***Visual Studio*** потрібно додати інтерпретатор цієї мови. Найновіші версії інтерпретатора ***Python*** можна завантажити з офіційного сайту та встановити на ПК окремо, проте поточна версія ***Visual Studio*** може і не підтримувати роботу з найновішою версією ***Python***. Кращий підхід у цьому випадку – встановлення інтерпретатора з ***Visual Studio***.

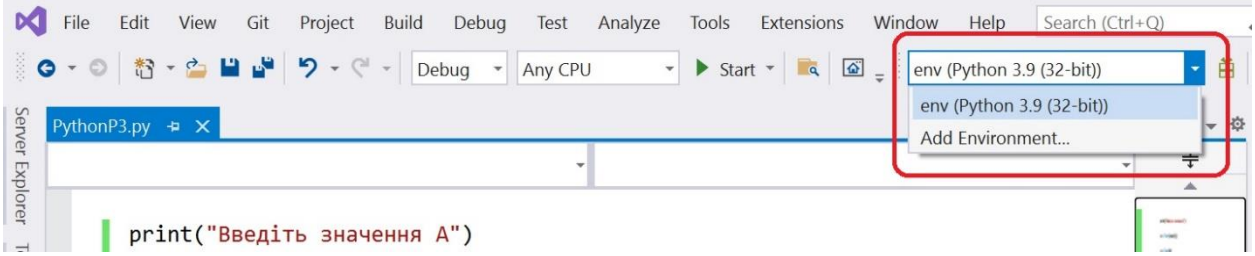
Для того, щоб мати можливість створювати проєкти на Python у Visual Studio потрібно встановити відповідне робоче навантаження за допомогою Visual Studio Installer. Створення проєкту стандартне:



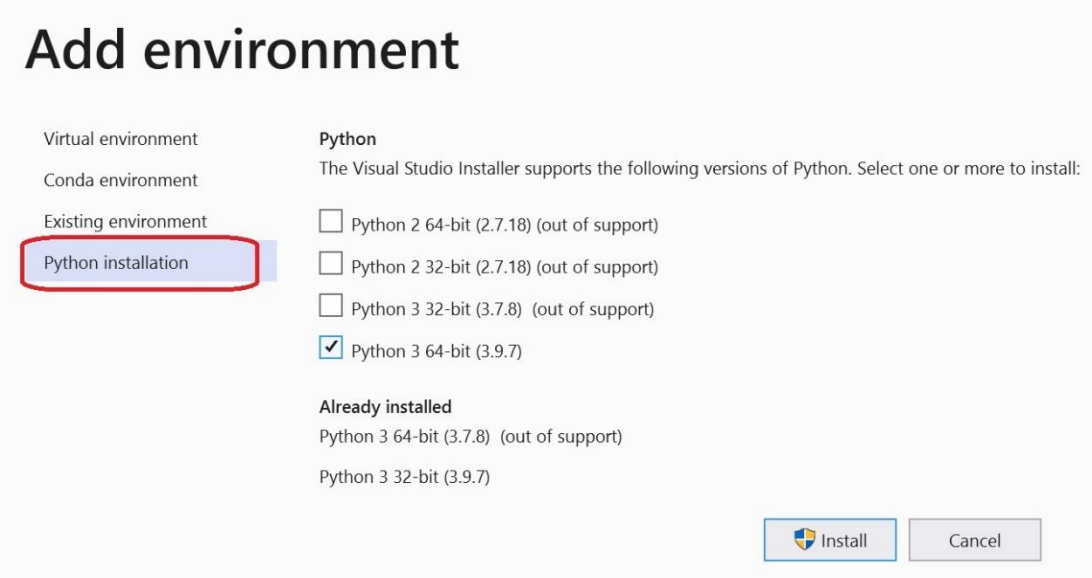
- *запустити Visual Studio та створити проєкт через меню File=> New=> Project, або Create a new project на стартовому екрані програми*
- *задати для пошуку мову Pythom та тип проєкту Console;*



- *обрати тип Python Application та створити проєкт;*
- *після створення нового проекту на Python потрібно знайти на панелі інструментів Visual Studio список середовищ виконання і обрати Add Environment;*



- *для додавання нового інтерпретатора у вікні Add Environment переходимо до пункту Python installation;*
- *тут вже можна обирати останню версію, – у цьому списку присутні лише версії Python, сумісні з нашою версією Visual Studio. Далі залишилося почекати поки Visual Studio Installer завантажить потрібні дистрибутиви і встановить їх.*



Щодо до операторів циклу то Python має доволі просту і лаконічну конструкцію циклу for:

```
For <змінна> in <набір значень> :
    <блок команд>
```

Він виконує блок команд для кожного значення змінної у вказаному наборі. Наборами значень можуть бути різного роду колекції та послідовності.

Код програми для розв'язування задачі 195 на мові ***Python***:

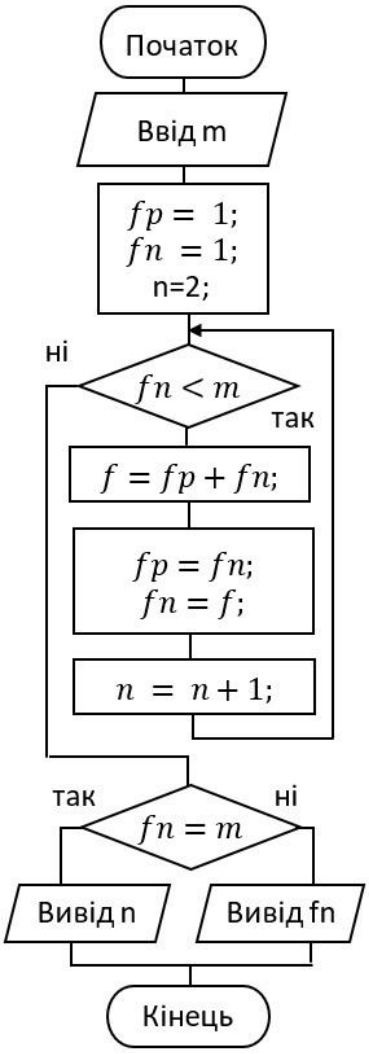
```
a = float(input("Введіть значення A"))
u = float(1)
s = u
n = int(input("Введіть значення n"))
for i in range(n): # функція range(n) генерує набір
    u *= -a        # послідовних натуральних чисел з
    s += u         # з відрізка [0; n]
print("Значення суми s = %10.3f" % s)
```

225. Перевірити чи задане натуральне число m є числом Фібоначчі. Якщо так то вивести його порядковий номер n. Якщо ж ні, то вивести найближче до m число Фібоначчі, більше за m.

Послідовність чисел Фібоначчі задається співвідношеннями:

$$F_1 = 1, F_2 = 1, F_n = F_{n-1} + F_{n-2}, n = 3, 4, \ldots$$

Задача відрізняється від попередньої тим, що тут кількість повторів циклу наперед не відома. На етапі, коли користувач задає число, яке він планує перевірити не можливо визначити скільки саме чисел Фібоначчі потрібно обчислити, щоб досягнути потрібної величини. Тут слід поступово обчислювати члени послідовності і порівнювати їх з числом користувача поки не знайдеться потрібний.



Тобто, в алгоритмі розв'язування цієї задачі будемо використовувати цикл з умовою, а не цикл з параметром (див. пункт з алгоритмом розв'язування задачі 195). Який саме тип циклу, – з умовою на початку циклу, чи з умовою після ітерації, принципового значення не має, ці конструкції є взаємозамінними. Зупинимося на умові на початку циклу.

Фактично процес розв'язування задачі полягає в поступовому знаходженні членів послідовності Фібоначчі і порівнянні їх з заданим користувачем числом:

- *користувач вводить число m, яке він бажає перевірити;*
- *програма бере перші два числа послідовності Фібоначчі;*
- *виконується перевірка того, чи чергове знайдене число Фібоначчі не перевищує число, задане користувачем і якщо ні, то обчислюється наступне число послідовності;*
- *коли програма знаходить перше число Фібоначчі, більше за m, припиняє циклічний пошук і переходить до повідомлення результату;*
- *по завершенню циклу програма перевіряє задане число m співпадає з останнім обчисленим числом Фібоначчі, робить відповідні висновки, повідомляє користувача та.*

Графічне зображення сформульованого алгоритму подано на малюнку.

Програма на Pascal.

Для циклу з умовою на початку ***Pascal*** має оператор:

```
while <умова> do <команда>
```

Тут:

- ✓ *виконання команди повторюється якщо перевірка умови дає істинний результат;*
- ✓ *ітерації циклу продовжують виконуватися до тих пір, поки умова не набуде хибного значення;*
- ✓ *після того, як умова перестає виконуватися, програма переходить до виконання наступного після* **while** *оператора;*
- ✓ *команда в тілі циклу може бути тільки одна, якщо потрібно в тілі циклу виконувати декілька інструкцій, то використовується складений оператор* **begin**…**end.**

Цикл з умовою після ітерації у ***Pascal*** має вигляд:

```
repeat <команда1>;<команда2>;...;<командаN> until <умова>
```

Цікаво, що ця конструкція чи не єдина в ***Pascal*** може містити послідовність операторів без ***begin…end***. Ще одна особливість, – завершення циклу ***repeat*** відбувається, коли умова набуває істинного значення, тому, при заміні ***repeat*** на ***while***, чи навпаки слід інвертувати логічні вирази.

Орієнтовний код реалізації розробленого вище алгоритму на ***Pascal***:

```
program Task225;
var m, f, fp, fn, n: integer;
begin
    fp := 1; fn := 1; {початкові члени послідовності}
    n := 2; {n - номер останнього числа послідовності}
    writeln('Введіть натуральне число:');
    readln(m);
    while fn < m do
    begin
        f := fp + fn;
        {переходимо до пари наступних чисел Фібоначчі}
        fp := fn; fn := f; n := n + 1
    end;
    if fn = m then
        writeln( m, ' є ', n, '-им числом Фібоначчі')
```

```
    else  writeln( 'Найближче до ', m,
        ' справа число Фібоначчі: ', fn);
end.
```

Програма на C++.

В ***C++*** оператор циклу з умовою на початку має вигляд:

```
while(<умова>)
{
  <послідовність команд>;
}
```

а оператор з умовою після ітерації:

```
do {
  <послідовність команд>;
} while(<умова>);
```

В обидвох конструкціях:

- ✓ *виконання команди повторюється до тих пір, поки умова не набуде хибного значення;*
- ✓ *після того, як умова перестає виконуватися, програма переходить до виконання наступного за* **`while`** *оператора;*
- ✓ *в конструкції з умовою на початку фігурних дужок можна не ставити, якщо послідовність команд складається лише з однієї команди.*

Реалізація розв'язку задачі 225 на C++:

```
#include <iostream>
using namespace std;
int main()
{
    int m, f, fp = 1, fn = 1, n = 2;
    cout<<"Enter number m \n>>";
    cin>>m;
    while (fn < m) {
        f = fp + fn;
        fp = fn; fn = f;
        n++;
    }
```

```
    if (fn == m) {
        cout << m << " is "<< n
            <<" Fibonacci number " << endl;
    }
    else {
        cout << "First Fibonacci number after "
            << m <<" is " << fn << endl;
    }
}
```

Програма на Java.

Реалізація цього ж алгоритму на ***Java***, після зроблених щодо програми ***C++*** зауважень, зайвих коментарів не потребує в силу використання цими мовами ідентичних синтаксичних конструкцій циклу:

```
import java.util.Scanner;
public class Task225 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int m, f, fp = 1, fn = 1, n = 2;
        System.out.println("Введіть число m");
        m = scanner.nextInt();
        while(fn<m) {
            f = fp + fn;
            fp = fn; fn = f;
            n++;
        }
        if(fn == m) {
            System.out.println("Число " + m +" є "
                    + n +"-им числом Фібоначчі");
        }else {
            System.out.println("Найближче до " + m +
                     " справа число Фібоначчі: " + fn);
```

```
        }
    }
}
```

Програма на Python.

В ***Python*** оператор циклу з умовою на початку має вигляд:

```
while <умова> :
    <блок команд>
```

а оператор з умовою після ітерації, як синтаксична конструкція, відсутній взагалі. Правила виконання стандартні:

- ✓ *виконання блоку команд повторюється до тих пір, поки умова не набуде хибного значення;*
- ✓ *після того, як умова перестає виконуватися, програма переходить до виконання наступного за* **`while`** *оператора;*
- ✓ *вкладений блок команд може містити довільну кількість команд відділену від основного блоку однаковим горизонтальним відступом зліва.*

Код програми для розв'язування задачі 225 на мові ***Python***:

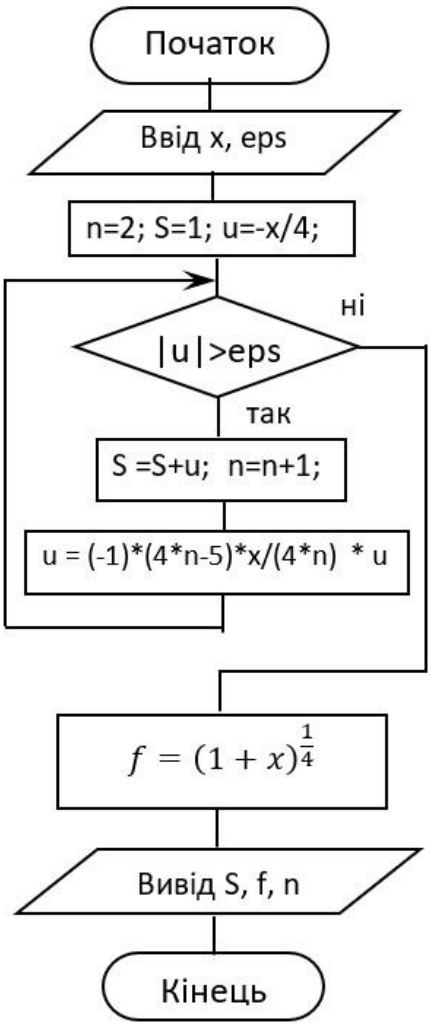
```
fp = 1; fn = 1
n = 2 #номер останнього числа послідовності}
print("Введіть натуральне число:")
m = int(input())
while fn < m :
     f = fp + fn
     fp = fn; fn = f
     n = n + 1
if fn == m :
    print( m, ' є ', n, '-им числом Фібоначчі')
else:
    print( 'Найближче до ', m, \
        ' справа число Фібоначчі: ', fn)
```

240. Скласти програму для обчислення значення елементарної функції в заданій точці за допомогою степеневого ряду з точністю до члена, меншого за 0.00001. Визначити кількість доданків. Порівняти отримане значення суми зі значенням вказаної функції, обчисленим за допомогою бібліотечних функцій.

$$(1+x)^{\frac{1}{4}} = 1 + \frac{1}{4}x - \frac{1*3}{4*8}x^2 + \frac{1*3*7}{4*8*12}x^3 - \frac{1*3*7*11}{4*8*12*16}x^4 + \cdots .$$

За структурою алгоритму розв'язування задача 240 аналогічна до задачі 225:

- *користувач вводить значення x, для якого обчислюється функція за допомогою суми ряду;*
- *програма обчислює початковий доданок суми (або кілька доданків);*
- *на кожній ітерації виконується перевірка того, чи знайдений черговий доданок достатньо малий за абсолютною величиною, щоб забезпечити відповідну точність обчислення значення функції;*
- *якщо абсолютна величина чергового доданка не менша від заданої точності, програма додає його значення до суми, обчислює наступний доданок та повторює перевірку;*
- *при досягненні потрібної точності циклічний процес завершується і програма виводить отриманий результат разом і значенням функції, отриманим за допомогою бібліотечних функцій.*



За складністю ця задача перевищує попередню лише в способі отримання наступного значення члена степеневого ряду. Обчислення загального члена ряду безпосередньо за формулою під знаком суми в умові не завжди можливе через певні обмеження математичного апарату обраної мови програмування. Наприклад, функція піднесення до степеня не працює з від'ємною основою степеня, а потрібно обчислити $(-1)^n$, при великих n значення n! виходить цілочисельного типу, і т.п. В таких випадках для знаходження значення наступного члена ряду A_n зручно використовувати рекурентне співвідношення вигляду $\mathrm{A}_n = M_n(x)\mathrm{A}_{n-1}$, що пов'язує його із значенням попереднього. Множник $M_n(x)$ завди можна знайти, формально розділивши вираз для A_n на вираз для A_{n-1}: $M_n(x) = \frac{\mathrm{A}_n}{\mathrm{A}_{n-1}}$. Деколи, як у випадку цієї задачі, $M_n(x)$ можна побудувати просто проаналізувавши формулу загального члена ряду.

Ряд в умові нашої задачі

$$1+\frac{1}{4}x-\frac{1*3}{4*8}x^2+\frac{1*3*7}{4*8*12}x^3-\frac{1*3*7*11}{4*8*12*16}x^4+\cdots$$

має певну специфіку, порівняно з більшістю задач цієї групи (задачі 226-240), – тут перші два доданки не зовсім відповідають загальній формулі члена ряду, тому перший доданок включаємо безпосередньо в суму, і обчислення суми починаємо з $S=1$, а не з $S=0$. У змінну поточного доданка записуватимемо члени ряду, починаючи з другого ($u=\frac{1}{4}x$). Наступні доданки вже вкладаються в загальну схему і дозволяють побудувати рекурентну формулу.

Для побудови згаданого вище коефіцієнта $M_n(x)$ зауважимо, що нашому знакопочерговому ряді кожен наступний член відрізняється від попереднього протилежним знаком, та на одиницю більшим степенем змінної x. У знаменнику доданків маємо добуток послідовних натуральних чисел, кратних 4, котрі можна записати як $4n$, а в чисельнику, починаючи з другого множника бачимо подібні множники: 3 = 4 – 1, 7=8 – 1 = 4*2 – 1, 11 = 12-1 =4*3 – 1, і т.д. Враховуючи порядок, множник в чисельнику можна задати відповідно виразом $4(n-1)-1=4n-5$. Тобто, $M_n(x)=-\frac{4n-5}{4n}x$ і, починаючи з третього, члени ряду можна шукати за рекурентною формулою

$$A_n=-\frac{4n-5}{4n}x\cdot A_{n-1}.$$

Враховуючи, що синтаксичні конструкції, потрібні для реалізації описаного алгоритму, вже розглянуті в у відповідних пунктах розв’язування задачі 225, далі подаємо програмний код без зайвих пояснень.

Зупинимося лише на не зовсім прозорому моменті щодо підрахунку кількості доданків. Нумеруючи доданки з 0 та записавши значення $A_0=1$ безпосередньо в початкове значення суми ми використовуємо для початку ітераційного процесу член ряду $A_1=-\frac{1}{4}x$.

Значення $n=2$, яким ініціалізується змінна для визначення кількості доданків вказує, що після проходження першої ітерації в сумі буде враховано два доданки A_0 та A_1. Але в самій ітерації значення n збільшується на 1 і по завершенні усього циклу змінна n міститиме кількість доданків, на одиницю більшу за справжню кількість доданків у сумі. Насправді ж $n-$ий доданок буде обчислено в змінній u, але не додано до суми S. Тому для вказання кількості доданків потрібно виводити значення $n-1$. Змінити ініціалізацію $n=2$ не можна, бо ця змінна бере участь в обчисленні наступного члена ряду.

Програма на Pascal.

```
program Task240;
const eps = 0.00001;
var x, s, u, f : double;
n: integer;
begin
    writeln('Введіть значення змінної x');
    readln(x);
    s := 1; n := 2; u := x / 4;
    while abs(u) > eps do
    begin
        s := s + u;
        u := -x * (4 * n - 5) / (4 * n) * u;
        n := n + 1
    end;
    f :=  sqrt(sqrt(1 + x));
    {корінь 4-степеня можна записати і так}
    writeln('Сума ряду s = ', s:12:5);
    writeln('Значення функції f = ', f:12:5);
    writeln('Кількість доданків : ', n - 1 )
end.
```

Програма на C++.

```
#include <iostream>
using namespace std;
int main()
{
    double x, u, s = 1, eps = 0.00001;
    cout<<"Enter variable x value: ";
    cin >> x;
    u = x / 4;
```

```
    int n = 2;
    while (abs(u) > eps)
    {
        s += u;
        u *= -x * (4 * n - 5) / (4 * n);
        n++;
    }
    printf("Summ value s = %.6f\n", s);
    printf("Function value : %.6f\n", pow(1 + x, 0.25));
    cout<<"Count of additions n =  " << n - 1;
}
```

Програма на Java.

```
import java.util.Scanner;
public class Task240 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double x, u, s = 1, eps = 0.00001;
        System.out.println("Введіть значення x");
        x = scanner.nextDouble();
        u = x/4;  int n = 2;
        while(Math.abs(u)>eps) {
            s += u;  u *= -x*(4*n-5)/(4*n);
            n++;
        }
        System.out.printf("Значення суми s = %.6f\n", s);
        System.out.printf("Значення функції: %.6f\n",
            Math.pow(1+x,0.25));
      System.out.println("Кількість доданків " + (n - 1));
    }
}
```

Програма на Python.

```
eps = 0.00001
print('Вееділь значення змінної x')
x = float(input())
s = 1; n = 2; u = x / 4
while abs(u) > eps:
    s += u
    u *= -x * (4 * n - 5) / (4 * n)
    n += 1
f = (1 + x)**0.25
print('Сума ряду s = %12.5f' % s)
print('Значення функції f = %12.5f' % f)
print('Кількість доданків : %5d'% (n - 1))
```

255. Побудувати таблицю значень заданої функції двох змінних у заданому користувачем прямокутнику $[a; b] \times [c; d]$. Крок зміни значень змінних $x \in [a; b]$ та $y \in [c; d]$ вибрати таким чином, щоб на кожному з відрізків розміщувалося по 8 точок. Результат обчислень вивести на екран у вигляді прямокутної таблиці, в якій по горизонталі вказані різні значення змінної x, по вертикалі – значення змінної y, а на перехресті стовпця зі значенням змінної x та рядка зі значенням змінної y – відповідне значення функції.

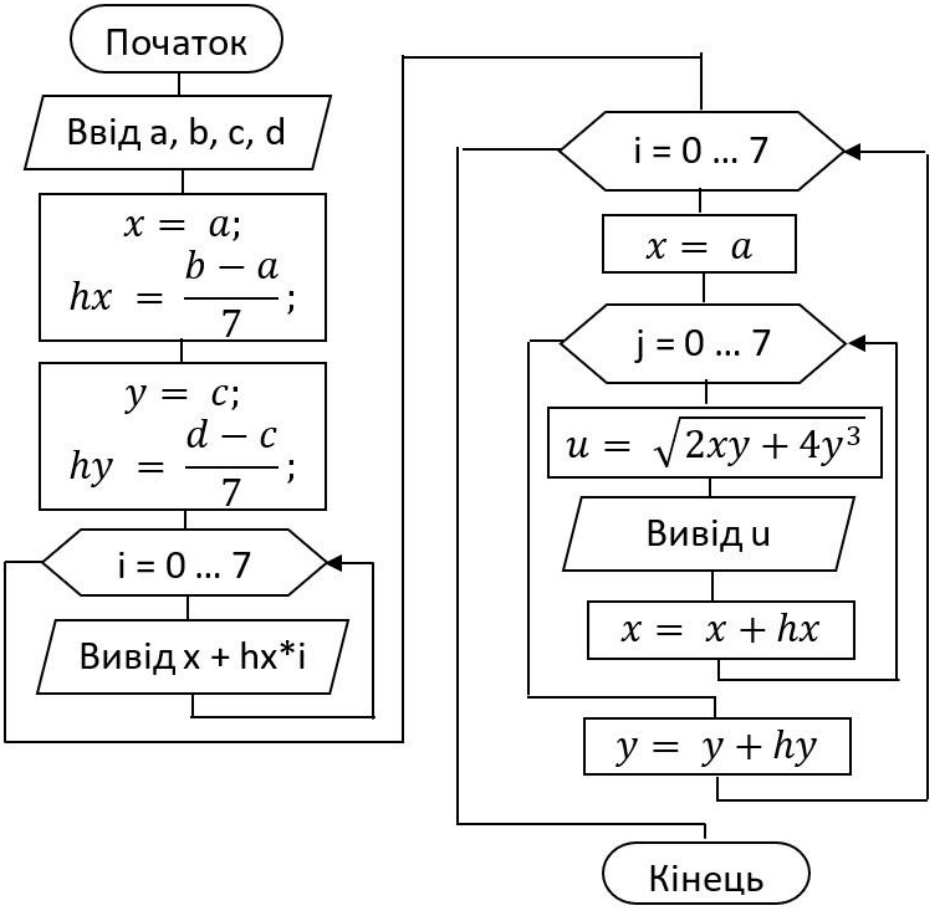
$$u = \sqrt{2xy + 4y^3}.$$

Задачі 241–270 відрізняються від попередніх груп задач цього розділу тим, що потребують виконання «циклу в циклі». Тобто, кожна ітерація одного циклу передбачає в ході свого виконання повний цикл ітерацій ще одного циклу. Така ситуація в програмуванні отримала назву ***вкладення циклів***. Її реалізація в коді програм не потребує якихось специфічних конструкцій, – просто в блоці команд (тілі) ***зовнішнього циклу*** записують ще один оператор циклу – ***внутрішній цикл***. Вкладення циклів може продовжуватися «вглиб», – кожен внутрішній цикл може містити ще один або декілька внутрішніх циклів. Глибина вкладення циклів, взагалі кажучи, нічим не обмежується, окрім, обчислювальних потужностей техніки, бо при кожному вкладенні циклів потреба в ресурсах зростає на цілий порядок.

Для розв'язування нашої задачі достатньо пари вкладених циклів, – зовнішній цикл повинен переглянути усі значення одного аргумента, наприклад, x, а внутрішній, – до поточного значення цієї змінної поставити кожне із значень другого аргумента, та для кожної, утвореної в такий спосіб точки (x, y) обчислити та вивести значення функції $u(x, y)$.

Щодо типу циклів, то в задачах про табулювання функції за відомим відрізком, та кроком нескладно визначити кількість точок (значень аргумента) ще до початку циклу. Тому будемо використовувати конструкцію циклу з лічильником, який часто називають просто ***цикл for*** за назвою відповідного оператора. Тим більше, що умовою задачі кількість точок на відрізку задана наперед, а крок зміни аргументів ми визначаємо відповідно до неї.

Щоб сформулювати алгоритм тут скористаємося блок-схемою (див. малюнок), а словесне формулювання подавати не будемо. Для виводу значень функції у вигляді прямокутної таблиці доведеться використати перед основним циклом додатковий цикл, щоб вивести у заголовок стовпців відповідні значення одного з аргументів.



Програма на Pascal.

Програмування вкладенихциклів на ***Pascal*** потребує уважного використання складеного оператора ***begin…end***. Якщо за узгодженням кількостей ***begin*** та ***end*** слідкує компілятор і не дозволить виконати програму з незакритим складеним оператором, чи зайвим ***end***, то коректне об’єднання операторів повністю належить до сфери відповідальності розробника. Випадково перенесений у невідповідне місце ***end*** часто-густо може виявитися причиною несподіваної поведінки програми, зокрема, зациклень чи інших «сюрпризів».

Ще один важливий момент наступної програми, – належний вибір параметрів форматування виводу дійсних чисел для вирівнювання даних в стовпцях таблиці.

```
program Task255;
var a, b, c, d, u, x, hx, y, hy : double;
    i, j : integer;
begin
    writeln('Задайте відрізок для змінної x');
    readln( a, b);{з консолі можна читати кілька значень}
    x := a; hx := (b - a) / 7;
    writeln('Задайте відрізок для змінної y');
    readln( c, d);
    y := c; hy := (d - c) / 7;
    write(' y\x ');{подальший вивід буде в тому ж рядку}
    {виводимо заголовок таблиці зі значеннями x}
    for i := 0 to 7 do write((x + hx * i):8:3);
    writeln();{перехід на новий рядок}
    {Зовнішній цикл за аргументом y}
    for i := 1 to 8 do
    begin
        write(y:5:2);
        x := a; {вкладений цикл за аргументом x}
        for j := 1 to 8 do
        begin
            u := sqrt(2 * x * y + 4 * y * y * y);
            write(u:8:2);
            x := x + hx;
        end;
        writeln();
        y := y + hy;
    end
end.
```

Програма на C++.

Програмування вкладених циклів на ***C***/***C***++ теж потребує уважного ставлення до блоків коду в тілі циклів, взятих у фігурні дужки. Компілятор завжди контролює кількість відкритих-закритих дужок і, якщо кількість символів “{” у коді не співпадає з кількістю символів “}”, повідомляє про синтаксичну помилку та не компілює такий код. Сучасні інтегровані системи розробки, зокрема, Microsoft Visual Studio повідомляють про подібні ситуації вже на етапі написання коду і навіть підказують як виправити помилку. Попри те, якщо ви охопите фігурними дужками групу команд, невідповідно до логіки алгоритму розв’язування задачі, жоден компілятор чи IDE не врятує вас від помилкових результатів роботи програми.

Для форматування таблиці значень функції використовуємо вивід дійсних чисел за допомогою функції ***printf***.

```
#include <iostream>

using namespace std;

int main()

{

    double a, b, c, d, u;

    cout << "Enter variable x bounds"<<endl;

    cin >> a >> b;

    double x = a, hx = (b - a) / 7;

    cout << "Enter variable x bounds" << endl;

    cin >> c >> d;

    double y = c, hy = (d - c) / 7;

    cout<<"y\\x  ";

    for (int i = 0; i < 8; i++)

    {

       printf("%8.2f", x + hx * i);

    }

    cout << endl;

    for (int i = 0; i < 8; i++) {

        printf("%5.2f", y);

        x = a;

        for (int j = 0; j < 8; j++) {
```

```
            u = sqrt(2 * x * y + 4 * y * y * y);
            printf("%8.2f", u);
            x += hx;
        }
        cout<<endl;
        y += hy;
    }
}
```

Програма на Java.

Сказане вище стосовно програмування вкладених циклів на ***C/C++*** стосується програм на ***Java*** ще в більшій мірі. Тут фігурних дужок в коді ще більше (принаймні, на пару, що обмежує тіло класу). В ***Java*** ООП є ще одне цікаве застосування блоків коду у фігурних дужках, – фрагменти робочого коду у фігурних дужках на рівні класу (їх називають ***ініціалізаторами***) компілятор збирає і компілює в програму, що виконується під час створення об'єкта класу.

Для форматного виводу дійсних чисел в Java також використовується ***printf*** з такими ж форматними вказівниками, але в цьому випадку йдеться не про функцію, а про метод класу PrintStream, що викликається від об'єкта вихідного потоку консолі System.in.

```
import java.util.Scanner;
public class Task255 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double a, b, c, d, u;
        System.out.println("Введіть межі для x");
        a = scanner.nextDouble();
        b = scanner.nextDouble();
        double x = a, hx = (b - a)/7 ;
        System.out.println("Введіть межі для y");
        c = scanner.nextDouble();
        d = scanner.nextDouble();
```

```
        double y = c, hy = (d - c)/7 ;
        System.out.print("y\\x  ");
        for (int i = 0; i < 8; i++)
            System.out.printf("%8.2f",x + hx*i);
        System.out.println();
        for (int i = 0; i < 8; i++){
            System.out.printf("%5.2f",y);
            x = a;
            for (int j = 0; j < 8; j++) {
                u = Math.sqrt(2*x*y + 4*y*y*y);
                System.out.printf("%8.2f", u);
                x += hx;
            }
            System.out.println();
            y += hy;
        }
    }
}
```

Програма на Python.

Програма на ***Python*** в цьому розділі потребує використання форматованого виводу даних в одну стрічку. Проблема в тому, що функція ***print*** в ***Python,*** за замовчуванням, після виводу усього списку своїх даних, завершує стрічку, і наступний виклик цієї функції починає вивід на консоль з нової стрічки. Для відміни переходу виводу на нову стрічку у ***Python*** версій 2.х використовували кому після виклику функції ***print*** в ***Python 3.x*** для цього використовується така синтаксична конструкція:

```
print ("Початок стрічки", end=" ")
print("продовження стрічки")
```

Стосовно вкладених циклів та блоків коду ***Python***, то тут вкладення задається наступним рівнем горизонтального відступу блоку коду. Багато вкладених конструкцій надають коду програми «сходинчатого» вигляду.

Код програми для розв’язування задачі 255 на мові ***Python***:

```
import math #для обчислення кореня квадратного
print('Задайте відрізок для змінної x')
a = float(input())
b = float(input())
x = a; hx = (b - a) / 7
print('Задайте відрізок для змінної y')
c = float(input())
d = float(input())
y = c; hy = (d - c) / 7
print(" y\\x ", end ="")
for i in range(0, 8):
    print("%8.2f" % (x + i*hx), end ="")
print() # перехід виводу на нову стрічку
#Зовнішній цикл за аргументом y
for i in range(0,8):
    print("%5.2f" % y, end ="")
    x = a
    #вкладений цикл за аргументом x
    for j in range(0,8):
        u = math.sqrt(2 * x * y + 4 * y * y * y)
        print("%8.2f" % u, end ="")
        x += hx
    print()
    y += hy
```

Розділ 4. Опрацювання одновимірних масивів.

Скінченні послідовності даних. Масиви і списки.

271. Задано масив, що містить 17 цілих чисел. Вивести парні елементи масиву, що більші за задане користувачем число M та знайти їх кількість.

272. У заданому масиві з 20 дійсних чисел визначити добуток суми додатних елементів та суми від'ємних елементів.

273. Заданий масив містить значення мінімальної добової температури повітря у Львові протягом жовтня. Визначити, скільки днів у жовтні траплялися заморозки (від'ємна температура) та середнє значення мінімальної добової температури за місяць.

274. Для заданого масиву з 19 цілих чисел визначити суму парних елементів масиву, суму непарних елементів та різницю між цими сумами.

275. Задано масив, що містить дійсні числа, які задають суму чистого прибутку компанії протягом 12 місяців минулого року. Визначити скільки місяців компанія мала збитки (від'ємне значення прибутку) та за який місяць компанія отримала найбільший прибуток.

276. Задано масив, утворений з 25 дійсних чисел. Обчислити добуток мінімального та максимального елементів масиву.

277. Задано масив, утворений з 25 дійсних чисел. Обчислити добуток елементів масиву, значення яких потрапляють на відрізок $[a; b]$. Значення a та $b > a$ користувач вводить з клавіатури під час роботи програми.

278. Задано масив, що містить 24 цілі числа, які задають кількість відвідувачів сайту, зафіксовану протягом кожної години доби. Визначити загальну кількість відвідувачів сайту протягом доби. Вивести години доби, коли сайт відвідали найменше та найбільше користувачів.

279. У заданому масиві з 21 дійсного числа визначити суму добутку додатних елементів та добутку від'ємних елементів.

280. Задано масив, що містить 19 цілих чисел. Вивести непарні елементи масиву, менші за задане користувачем число M, та їх суму.

281. Задано масив, що містить 24 дійсні числа, що відображають кількість кіловат електроенергії, спожиту підприємством протягом кожної години доби. Визначити загальну кількість електроенергії, спожитої підприємством за добу. Вивести години доби, коли споживання електроенергії становило від a до b. Значення a та $b > a$ користувач задає з клавіатури під час роботи програми.

282. Задано масив з 27 дійсних чисел. Обчислити відношення максимального елемента масиву до середнього значення усіх додатних елементів масиву.

283. Задано масив, що містить 11 цілих чисел, що задають кількість учнів у кожному з 11-класів школи. Визначити загальну кількість учнів в школі та середню кількість учнів у класі. Вивести клас, у якому учнів найменше.

284. Задано масив, що містить 24 значення температури повітря на вулиці, виміряної кожної години протягом доби. Визначити середньодобову температуру, мінімальне і максимальне значення температури та вивести години доби, коли ці значення досягалися.

285. Задано масив, утворений з 22 цілих чисел. Обчислити суму елементів масиву, значення яких не потрапляють на відрізок $[a; b]$. Значення a та $b > a$ користувач вводить з клавіатури під час роботи програми.

286. Задано масив, що містить 19 цілих чисел. Визначити суму парних елементи масиву, що менші за задане користувачем число M та суму непарних елементів, більших за M.

287. У заданому масиві з 25 дійсних чисел визначити відношення найменшого додатного елемента до найбільшого від'ємного.

288. Для заданого масиву з 20 цілих чисел визначити відношення суми парних елементів масиву до суми непарних.

289. Задано масив, що містить дійсні числа, які задають суму чистого прибутку компанії протягом 12 місяців минулого року. Вивести номери місяців, в яких компанія мала збитки (від'ємне значення прибутку) та обчислити середньомісячний прибуток компанії.

290. Задано масив, утворений з 25 дійсних чисел. Обчислити відношення суми елементів, що більші за задане користувачем число M до суми елементів, що менші за M.

291. Задано масив, що містить 30 дійсних чисел, що відображають кількість кубометрів газу, використану підприємством протягом кожного дня квітня. Визначити загальну кількість газу, спожитого підприємством за місяць, найменшу та найбільшу витрату газу за день та вказати дні місяця, коли вони досягалися.

292. Задано масив з 20 дійсних чисел. Обчислити відношення мінімального елемента масиву до середнього значення усіх від'ємних елементів масиву.

293. Заданий масив з 30 дійсних чисел містить значення добової кількості опадів у Львові протягом червня. Визначити, скільки дощових днів було у Львові протягом червня та встановити середньодобову кількість опадів протягом місяця.

294. Для заданого масиву з 22 цілих чисел визначити відношення суми парних елементів масиву до суми непарних елементів масиву.

295. Задано масив, що містить цілі числа, які задають суму чистого прибутку компанії протягом 12 місяців минулого року. Визначити скільки місяців компанія не мала збитків (додатне значення прибутку або 0) та середнє значення місячного прибутку компанії протягом року.

296. Задано масив з 20 дійсних чисел. Вивести усі додатні елементи масиву у порядку зростання їх номерів та вказати їх кількість, після чого вивести усі від'ємні елементи у порядку спадання їх номерів і їх кількість.

297. Задано масив, що містить 31 дійсне число, що відображає кількість кубометрів газу, використану підприємством протягом кожного дня січня. Визначити середню кількість газу, спожитого підприємством за день. Вивести дні місяця, коли витрати газу підприємством перевищували середнє значення не більше, ніж на $p\%$.

298. У заданому масиві з 20 дійсних чисел визначити відношення суми елементів з парними номерами до суми елементів з непарними номерами.

299. Задано масив, що містить значення максимальної добової температури повітря у Львові протягом листопада. Визначити, скільки днів у цьому місяці повітря у Львові прогрівалося вище за 10°C та день, у який температура була найвищою протягом місяця.

300. Задано масив, що містить 24 цілі числа, які задають кількість відвідувачів сайту, зафіксовану протягом кожної години доби. Визначити середню кількість відвідувачів сайту за годину. Вивести години доби, в які кількість відвідувачів відхилялася від середньої не більше, ніж на $p\%$.

301. Утворити три масиви дійсних чисел A, B та C, кожен містить по 10 елементів. Масив A заповнює користувач, вводячи його елементи з клавіатури. Масив B утворюється за правилом $B_k = \frac{(-1)^k(2k^2+1)}{k}$ $k = 1,2,\ldots,10$. Масив C утворити з масиву B, замінивши в ньому всі від'ємні елементи відповідними елементами масиву A (з тими ж порядковими номерами). Вивести масиви на екран, в кожному масиві знайти середнє арифметичне додатних елементів.

302. Три масиви дійсних чисел U, V та W містять по 12 елементів кожен. Елементи масиву U користувач вводить з клавіатури. Масив V заповнити випадковими числами з відрізка [-2; 3]. Масив W утворити з масиву V, замінивши в ньому всі додатні елементи максимальним елементом масиву U. Вивести масиви на екран, в кожному масиві знайти кількість елементів, більших за 1.

303. Створити і заповнити масиви дійсних чисел M, T та S по 15 елементів кожен. Масив M заповнює користувач, вводячи його елементи з клавіатури. Масив T утворити за правилом $T_k = \frac{99\,sin\,k}{(k+1)^2}$, $k = 1,2,\ldots,15$. Масив S утворити з масиву T, замінивши в ньому всі від'ємні елементи мінімальним елементом масиву M. Вивести масиви на екран, в кожному масиві знайти кількість елементів, абсолютна величина яких більша за 5.

304. Три масиви дійсних чисел A, B та C містять по 10 елементів кожен. Масив A заповнює користувач, вводячи його елементи з клавіатури. Масив B утворити за правилом $B_k = (-1)^k\sqrt{k!}$, $k = 1,2,\ldots,10$. Масив C утворити за правилом $C_k = 2A_k - 3B_k$. Вивести масиви на екран, в кожному масиві знайти суму додатних елементів.

305. Утворити три масиви дійсних чисел U, V та W, кожен містить по 12 елементів. Елементи масиву U користувач вводить з клавіатури. Масив V утворити за правилом $V_k = 11\,sin\left(\sqrt{(k+3)^3}\right)$, $k = 1,2,\ldots,12$. Масив W утворити за правилом $W_k = max\left\{\frac{U_k}{v_k}; \frac{V_k}{U_k}\right\}$. Вивести масиви на екран, в кожному масиві знайти суму елементів, абсолютна величина яких менша за 1.

306. Створити три масиви дійсних чисел A, B та C по 10 елементів кожен. Масив A заповнює користувач, вводячи його елементи з клавіатури. Масив B заповнити випадковими числами з відрізка $[-4; 2]$. Масив C утворити з масиву B, замінивши в ньому всі від'ємні елементи середнім значенням елементів масиву A. Вивести масиви на екран, в кожному масиві знайти елемент, значення якого найближче до числа 1.

307. Утворити три масиви дійсних чисел M, P та Q, кожен містить по 15 елементів. Елементи масиву M користувач вводить з клавіатури. Масив P утворити за правилом $P_k = k\,sin(2k) + 2k\,sin\,k$, $k = 1,2,\ldots,15$. Масив Q утворити за правилом $Q_k = max\{M_k; P_k\}$. Вивести масиви на екран, в кожному масиві знайти різницю максимального та мінімального елемента.

308. Три масиви дійсних чисел A, B та C складаються з 10 елементів кожен. Масив A заповнює користувач, вводячи його елементи з клавіатури. Масив B утворити за правилом $B_k = 20\,cos\,k - k,\ k = 1,2,\ldots,10$. Масив C утворити з масиву B, віднявши від кожного його елемента відповідний елемент масиву A. Вивести масиви на екран, в кожному масиві знайти суму квадратів мінімального та максимального елементів.

309. Утворити три масиви дійсних чисел U, V та W, кожен по 11 елементів. Елементи масиву U задає користувач, вводячи їх з клавіатури. Масив V утворити за правилом $V_k = (-1)^k\sqrt{(k+2)(k+1)},\ k = 1,2,\ldots,11$. Масив W утворити за правилом $W_k = min\{U_k; V_k\}$. Вивести масиви на екран, в кожному масиві знайти різницю між максимальним елементом та середнім арифметичним усіх елементів.

310. Утворити три масиви дійсних чисел P, Q та S, по 17 елементів кожен. Масив P заповнює користувач, вводячи його елементи з клавіатури. Масив Q заповнити випадковими числами з відрізка [-5; 5]. Масив S утворити з елементів масиву P, додавши до кожного середнє арифметичне від'ємних елементів масиву Q. Вивести масиви на екран, в кожному масиві знайти відношення максимального елемента до мінімального.

311. Три масиви дійсних чисел A, B та C утворюються 17 елементів кожен. Елементи масиву A користувач вводить з клавіатури. Масив B утворити за правилом $B_k = 15\,cos\,k - 12\,sin(10-k), k = 1,2,\ldots,17$. Масив C утворити за правилом $C_k = min\{3A_k; B_k\}$. Вивести масиви на екран, в кожному масиві знайти відношення абсолютної величини суми від'ємних елементів до суми додатних елементів.

312. Масиви дійсних чисел A, B та C утворені з 12 елементів кожен. Масив A заповнює користувач, вводячи його елементи з клавіатури. Масив B утворити за правилом $B_k = \frac{k\,sin(2k)}{(k+1)!},\ k = 1,2,\ldots,12$. Масив C утворити за правилом $C_k = 2A_k + B_k$. Вивести масиви на екран, в кожному масиві знайти суму від'ємних елементів.

313. Три масиви дійсних чисел U, V та W містять по 10 елементів кожен. Елементи масиву U користувач вводить з клавіатури. Масив V утворюється за правилом $V_k = 12\,cos\left(\sqrt[k+1]{k^2}\right), k = 1,2,\ldots,10$. Масив W утворити за правилом $W_k = max\{2U_k; 5V_k\}$. Вивести масиви на екран, в кожному масиві знайти номер елемента, найближчого за величиною до числа 5.

314. Створити три масиви дійсних чисел A, B та C по 10 елементів кожен. Масив A заповнює користувач, вводячи його елементи з клавіатури. Масив B заповнити випадковими числами з відрізка [-2.5; 1.5]. Масив C утворити за правилом $C_k = min\{A_k; B_{9-k}\}$. Вивести масиви на екран, в кожному масиві знайти найбільший серед від'ємних елементів.

315. Утворити масиви дійсних чисел U, V та W по 14 елементів кожен. Масив U заповнює користувач, вводячи його елементи з клавіатури. Масив V утворити за правилом $V_k = \frac{cos(12k)}{k!}$, $k = 1,2,\ldots,14$. Масив W утворити з масиву V, замінивши в ньому всі додатні елементи мінімальним елементом масиву U. Вивести масиви на екран, в кожному масиві знайти номер елемента, найближчого за величиною до 0.

316. Створити три масиви дійсних чисел M, P та Q по 10 елементів кожен. Масив M заповнює користувач, вводячи його елементи з клавіатури. Масив P утворити за правилом $P_k = 2\,sin\,k + cos\,k$, $k = 1,2,\ldots,10$. Масив Q утворити додаванням оберненого масиву M до масиву P, тобто $Q = M_{10} + P_1, M_2 = P_9 + Q_2, M_3 = P_8 + Q_3$, і т.д. Вивести масиви на екран, в кожному масиві знайти кількість елементів, значення яких належать інтервалу (-1; 1).

317. Утворити три масиви дійсних чисел A, B та C, кожен містить по 14 елементів. Масив A заповнює користувач, вводячи його елементи з клавіатури. Масив B утворити за правилом $B_k = \sqrt[k+1]{k^3}$, $k = 1,2,\ldots,14$. Масив C утворити за правилом $C_k = max\{|A_k|; 2B_k\}$. Вивести масиви на екран, в кожному масиві знайти номер елемента, найближчого за величиною до 1.

318. Створити і заповнити масиви дійсних чисел U, V та X, кожен по 16 елементів. Масив U заповнює користувач, вводячи його елементи з клавіатури. Масив V заповнити випадковими числами з відрізка [-5.5; 3.5]. Масив X утворити за правилом $X_k = min\{U_k; 3V_k\}$. Вивести масиви на екран, в кожному масиві знайти мінімальний додатний елемент.

319. Утворити три масиви дійсних чисел M, P та S, кожен містить по 12 елементів. Масив M заповнює користувач, вводячи його елементи з клавіатури. Масив P утворити за правилом $P_k = \frac{sin(7k)}{k^2}$, $k = 1,2,\ldots,12$. Масив S утворити з масиву P, замінивши в ньому всі додатні елементи мінімальним елементом масиву M. Вивести масиви на екран, в кожному масиві знайти номер максимального елемента.

320. Три масиви дійсних чисел A, B та C містять по 15 елементів кожен. Масив A заповнює користувач, вводячи його елементи з клавіатури. Масив B утворити за правилом $B_k = sin^3 k + cos\, 2k$, $k = 1,2,\ldots,15$. Масив C утворити додаванням оберненого масиву A до масиву B, тобто $C_1 = A_{15} + B_1$, $C_2 = A_{14} + B_2$, $C_3 = A_{13} + B_3$, і т.д. Вивести масиви на екран, в кожному масиві знайти суму елементів, значення яких належать інтервалу (-2; 0).

321. Створити три масиви дійсних чисел K, M та P, по 12 елементів кожен. Масив K заповнює користувач, вводячи його елементи з клавіатури. Масив M заповнюється випадковими числами з відрізка $[-2;\ 5]$. Масив P утворити з масиву M, замінивши в ньому всі від'ємні елементи середнім значенням від'ємних елементів масиву K. Вивести масиви на екран, в кожному масиві знайти кількість елементів з відрізка $[0;\ 3]$.

322. Утворити три масиви дійсних чисел M, P та Q, кожен містить по 11 елементів. Елементи масиву M користувач вводить з клавіатури. Масив P утворити за правилом $P_k = sin(k^3) + 3k\, sin\, k$, $k = 1,2,\ldots,11$. Масив Q утворити за правилом $Q = max\{3M_k; P_k\}$. Вивести масиви на екран, в кожному масиві знайти півсуму максимального та мінімального елемента.

323. Три масиви дійсних чисел X, Y та W складаються з 15 елементів кожен. Елементи масиву X задає користувач, вводячи їх з клавіатури. Масив Y утворюється за правилом $Y_k = 3\, cos\, k - k^2$, $k = 1,2,\ldots,15$. Масив W утворити з масиву Y, додавши до кожного його елемента середнє значення елементів масиву X. Вивести масиви на екран, в кожному масиві знайти різницю квадратів максимального та мінімального елементів.

324. Утворити три масиви дійсних чисел M, P та Q, кожен містить по 14 елементів. Елементи масиву M користувач задає з клавіатури. Масив P утворюється за правилом $P_k = \frac{(-1)^k(k^3+1)}{k^2}$ $k = 1,2,\ldots,14$. Масив Q створити на основі масиву P, замінивши при цьому всі від'ємні елементи відповідними елементами масиву M (з тими ж порядковими номерами). Вивести масиви на екран, в кожному масиві знайти суму додатних елементів.

325. Три масиви дійсних чисел K, M та W містять по 10 елементів кожен. Елементи масиву K користувач вводить з клавіатури. Масив M заповнити випадковими числами з відрізка [-10; 10]. Масив W утворити з масиву M, замінивши в ньому всі додатні елементи максимальним елементом масиву K. Вивести масиви на екран, в кожному масиві знайти суму від'ємних елементів.

326. Три масиви дійсних чисел A, B та C містять по 12 елементів кожен. Масив A заповнює користувач, вводячи його елементи з клавіатури. Масив B утворити за правилом $B_k = \frac{(-1)^k}{\sqrt{k}}$, $k = 1,2,\ldots,12$. Масив C утворити за правилом $C_k = A_k + 3B_k$. Вивести масиви на екран, в кожному масиві знайти добуток від'ємних елементів.

327. Створити і заповнити масиви дійсних чисел M, T та S, по 15 елементів кожен. Масив M заповнює користувач, вводячи його елементи з клавіатури. Масив T утворити за правилом $T_k = \frac{99\,sin\,k}{(k-7)(k-8)}$, $k = 1,2,\ldots,15$. Масив S утворити з масиву T, замінивши в ньому всі від'ємні елементи мінімальним елементом масиву M. Вивести масиви на екран, в кожному масиві знайти кількість елементів, абсолютна величина яких менша за 1.

328. Утворити три масиви дійсних чисел U, V та W, кожен містить по 11 елементів. Елементи масиву U користувач вводить з клавіатури. Масив V утворити за правилом $V_k = 7\,sin\left(\sqrt{(k-5)(k-6)}\right)$, $k = 1,2,\ldots,11$. Масив W утворити за правилом $W_k = max\left\{\frac{U_k}{v_k}; \frac{V_k}{U_k}\right\}$. Вивести масиви на екран, в кожному масиві знайти добуток елементів, абсолютна величина яких більша за 1.

329. Утворити три масиви дійсних чисел A, B та C, кожен по 14 елементів. Елементи масиву A користувач вводить з клавіатури. Масив B утворюється за правилом $B_k = (-1)^k\sqrt{(k-3)(k-4)}$, $k = 1,2,\ldots,14$. Масив C утворити за правилом $C_k = min\{A_k; B_k\}$. Вивести масиви на екран, в кожному масиві знайти середнє арифметичне додатних елементів.

330. Утворити три масиви дійсних чисел P, Q та S по 17 елементів кожен. Масив P заповнює користувач, вводячи його елементи з клавіатури. Масив Q заповнити випадковими числами з відрізка [-7; 7]. Масив S утворити з елементів масиву P, додавши до кожного середнє арифметичне від'ємних елементів масиву Q. Вивести масиви на екран, в кожному масиві знайти номер елемента, найближчого за величиною до числа 5.

331. У заданому масиві $A(12)$ знайти кількість нульових елементів та вивести їх індекси. Серед чисел, що не дорівнюють нулю, знайти мінімальний елемент та вивести його.

332. У заданому масиві $A(15)$ знайти найбільше значення, що зустрічається в масиві після відкидання з нього всіх елементів, які дорівнюють максимальному. Вивести вихідний масив, максимальне значення та найбільше значення із залишених.

333. Задано масив $A(20)$. Помножити всі додатні елементи масиву на квадрат найменшого елемента, а всі від'ємні - на квадрат найбільшого елемента. Вивести вихідний та сформований масиви а також знайдені мінімум та максимум.

334. У заданому масиві $A(15)$ обчислити середнє арифметичне значення кожної трійки послідовних елементів і записати їх у новий масив $B(5)$. Знайти в цьому масиві найбільший за модулем елемент. Вивести створений масив та його максимальний елемент.

335. Задано масив $A(19)$. Переставити елементи масиву таким чином, щоб на початку розміщувались додатні елементи, а потім - всі від'ємні. Вивести сформований масив і кількість додатних та від'ємних елементів.

336. Задано масив $C(10)$. Переставити елементи масиву таким чином, щоб на початку розміщувалась група елементів, більших за перший елемент вихідного масиву, потім - перший елемент вихідного масиву та група елементів, менших або рівних йому. Вивести вихідний масив та сформований.

337. У заданому масиві $A(14)$ знайти суму квадратів елементів з діапазону $[-2; 3]$ та кількість елементів, значення яких не попадає в наведений діапазон. Обчислену кількість записати на першу позицію вихідного масиву.

338. У заданому масиві $A(17)$ знайти найменше значення, що зустрічається в масиві після відкидання з нього всіх елементів, які дорівнюють мінімальному. Вивести вихідний масив, мінімальне значення та найменше значення із залишених.

339. Поміняти місцями максимальний та передостанній елементи в масиві $A(17)$. Вивести вихідний та перетворений масив.

340. У заданому масиві $A(15)$ поміняти місцями максимальний від'ємний та мінімальний додатний елементи. Вивести результуючий масив.

341. У заданому масиві $A(14)$ визначити різницю між середнім арифметичним перших п'яти елементів та сумою квадратів решти елементів масиву і записати її замість мінімального елемента. Вивести вихідний та отриманий масив.

342. У заданому масиві $A(12)$ знайти максимуми з кожної четвірки послідовних елементів і записати їх у новий масив $X(3)$. Знайти в масиві $X(3)$ мінімальний елемент. Вивести створений масив та його мінімальний елемент.

343. Задано масив $A(20)$. Помножити всі додатні елементи масиву на куб найменшого елемента, а всі від'ємні - на куб найбільшого елемента. Вивести вихідний та сформований масиви а також знайдені мінімум та максимум.

344. У заданому масиві $A(20)$ знайти суму кубів елементів з діапазону $[-5; 5]$ та кількість елементів, значення яких не попадає в наведений діапазон.

345. У заданому масиві $A(15)$ знайти середнє арифметичне додатних елементів та добуток від'ємних. Замінити знайденими значеннями максимальний та мінімальний елементи відповідно. Вивести результуючий масив.

346. Задано два масиви цілих чисел A та B. Утворити з їх елементів масив C у вигляді перетину множин елементів A та B (з елементів що є одночасно і в масиві A і в масиві B), елементи розташувати у порядку їх слідування елементів в масиві A.

347. Задано два масиви цілих чисел A та B. Утворити масив C, що містить лише ті елементи масиву B, яких немає в масиві A.

348. Задано два масиви цілих чисел A та B. Створити з їх елементів масив C, що містить лише додатні елементи масивів A та B, записавши спочатку елементи масиву A, а потім елементи масиву B у тому ж порядку, в якому вони слідували в початкових масивах.

349. Задано два масиви цілих чисел A та B. Утворити масив C, помістивши в нього тільки елементи масиву A, яких немає в масиві B. Порядок слідування елементів зберегти таким, яким він був в початковому масиві.

350. Задано два масиви цілих чисел A та B. Утворити новий масив C лише з додатних елементів масиву A та лише з від'ємних елементів B, записавши спочатку елементи масиву A, а потім елементи масиву B в тому ж порядку, в якому вони слідували в початкових масивах.

351. Задано два масиви цілих чисел A та B. Створити з їх елементів масив C, вибравши з масивів A та B лише елементи з інтервалу $(-10; 10)$. В новому масиві записати спочатку елементи з масиву A, а потім елементи з масиву B у тому ж порядку, в якому вони слідували в початкових масивах.

352. Задано два масиви цілих чисел A та B. Утворити масив C, що містить лише парні елементи масивів A та B, записавши спочатку елементи масиву B, а потім елементи масиву A в тому ж порядку, в якому вони слідували в початкових масивах.

353. Задано два масиви цілих чисел A та B. Утворити масив C лише з тих елементів масиву A, які більші за відповідні елементи масиву B з тими самими індексами.

354. Задано два масиви цілих чисел A та B. Створити масив C з усіх елементів масиву B, які більші за мінімальний елемент масиву A.

355. Задано два масиви цілих чисел A та B. Утворити масив C з усіх елементів масиву A, які більші за середнє значення елементів масиву B.

356. Задано два масиви цілих чисел A та B. Утворити новий масив C лише з парних елементів масиву A та лише з непарних елементів B, записавши спочатку елементи масиву B, а потім елементи масиву A в тому ж порядку, в якому вони слідували в початкових масивах.

357. Утворити цілочисельний масив C лише з непарних елементів заданого цілочисельного масиву A та лише з від'ємних елементів заданого цілочисельного масиву B, записавши елементи обидвох масивів «вперемішку», але у порядку зростання їхніх індексів в початкових масивах.

358. Утворити цілочисельний масив C лише з парних елементів заданого цілочисельного масиву A та лише парних елементів заданого цілочисельного масиву B, записавши спочатку елементи масиву B, а потім елементи масиву A у порядку, зворотному до того, в якому вони слідували в початкових масивах.

359. Створити масив C лише з додатних елементів масиву B та лише з від'ємних A, записавши спочатку елементи масиву A, а потім елементи масиву B у тому ж порядку, в якому вони слідували в початкових масивах. В кожному масиві знайти відношення максимального елемента до мінімального.

360. Задано два масиви цілих чисел A та B. Створити з їх елементів масив C за правилом об'єднання двох множин: кожен елемент масиву A і кожен елемент масиву B повинен входити в результуючий масив, причому лише один раз.

Практикум до розділу 4. Опрацювання одновимірних масивів

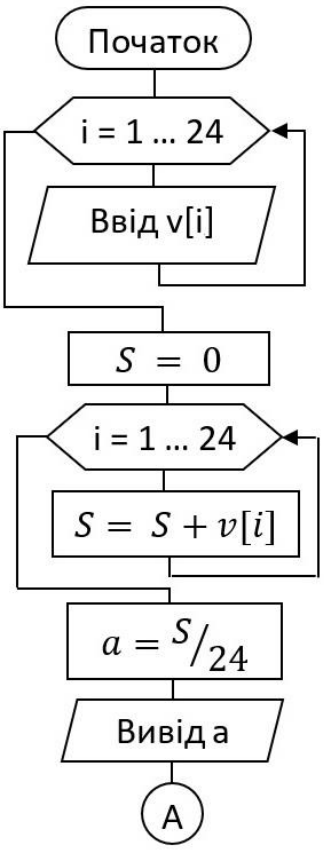
300. Задано масив, що містить 24 цілі числа, які задають кількість відвідувачів сайту, зафіксовану протягом кожної години доби. Визначити середню кількість відвідувачів сайту за годину. Вивести години доби, в які кількість відвідувачів відхилялася від середньої не більше, ніж на $p\%$.

Умова задачі не вказує на застосування масиву в якості структури для зберігання даних в програмі прямо. За умовою задачі ми можемо оголосити 24 змінні для зберігання 24-х цілочисельних даних, 24 рази виконати дії, потрібні для вводу цих даних та для їх опрацювання і. т. п. Зрозуміло, що, з використанням окремих змінних, програмного коду доведеться написати багато. Для автоматизації опрацювання подібних наборів даних в програмуванні використовують спеціальну структуру даних – ***масив***.

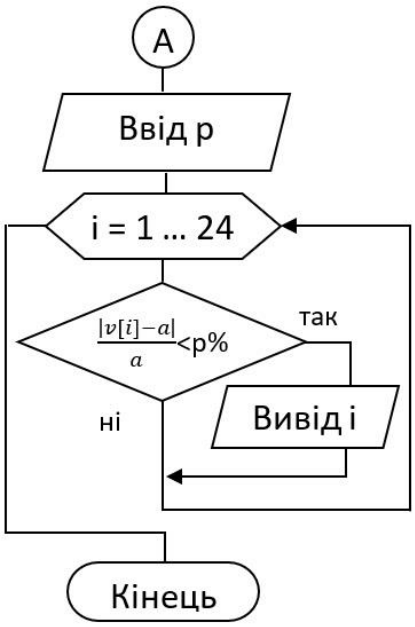
Масив є індексованою сукупністю однотипних даних, що позначаються спільним іменем змінної. Доступ до кожного окремого значення у цих даних здійснюється за іменем масиву та номером елемента в ньому. За допомогою циклу можна перебирати усі елементи масиву з метою виконання з ними вводу, виводу даних, виконання обчислень, тощо.

В нашому випадку потрібно:

- *утворити* **цілочисельний масив** *з 24-х елементів для зберігання даних про відвідування сайту протягом кожної години за добу;*
- *організувати цикл з лічильником, який отримає від користувача дані за кожну годину та запише їх у відповідний елемент масиву;*
- *підрахувати суму відвідувань протягом усієї доби, для чого також потрібно організувати цикл, в якому по черзі отримати значення кожного елемента масиву і додати до шуканої суми;*
- *обчислити середню відвідуваність на годину, розділивши загальну суму на 24 години;*
- *отримати від користувача значення відхилення для розв'язування другого завдання задачі;*
- *знову організувати цикл для перегляду елементів масиву та виводу тих з них, які потрапляють у вказаний діапазон з використанням умовного оператора.*



Заключна частина алгоритму (див. блок-схему) містить алгоритмічну конструкцію розгалуження в тілі циклу. Таке поєднання в літературі деколи можна зустріти під назвою ***циклічно-розгалуженого алгоритму***. Блок-схема до цієї задачі подана на двох малюнках, буква «А» в кільці вказує місце з'єднання окремих частин в один малюнок.



Реалізації цього алгоритму в різних технологіях програмування відрізнятимуться насамперед синтаксисом оголошення масивів. Але не тільки, – зберігання в пам'яті ПК сукупності даних та доступ до її елементів за порядковим номером впливає на швидкодію програм і є далеко не тривіальною задачею. Кожна технологія програмування вирішує її у свій спосіб.

Програма на Pascal.

В ***Pascal*** масиви оголошують за допомогою конструкції вигляду:

```
<variable-name> : array[<index-type>] of <element-type>;
```

у розділі оголошення змінних var. Тут ***variable-name*** – ім'я змінної-масиву, з яким він використовується в програмі, ***index-type*** – будь-який перелічуваний або інтервальний тип, що визначає допустимі значення індекса елемента, ***element-type*** – тип елементів масиву. Це доволі просторе визначення допускає різноманітні трактування і застосування в коді, на практиці ж в якості ***index-type***, зазвичай, використовують інтервальний тип вигляду ***1..maxid***, отримуючи в масиві послідовність елементів з номерами від ***1*** до ***maxid*** включно. Цього достатньо для розв'язування більшості задач з використанням масивів. В цьому розділі використовуватимемо саме таке трактування. Зауважимо, хіба-що, що підхід до організації масивів, при якому програміст сам визначає спосіб індексування елементів, завдає компілятору додаткових клопотів, що в свою чергу відбивається на швидкодії програм.

Запис алгоритму розв'язання задачі 300 мовою ***Pascal*** може виглядати так:

```
program Task;
var visits:array[1..24] of  integer;{оглошуємо масив}
    s, i: integer; a, p: real;
begin
    write('Ведіть відвідуваність сайту ');
    writeln('кожної години протягом доби');
```

```
    {вводимо елементи масиву в циклі}
    for i:=1 to 24 do
    read(visits[i]);
    {функція read дозволяє також читати числа,
       введені в одну стрічку через пробіл}
    s := 0;
    {цикл по елементах списку}
    for i:=1 to 24 do s:= s + visits[i];
    a := s / 24;
    writeln('Середньодобова відвідуваність', a:10:2);
    writeln('Задайте коефіцієнт відхилення у % :');
    readln(p);
    writeln('Години з менш, ніж ', p:5:2,' -');
    writeln(' відсотковим відхиленням від ');
    writeln('середньої відвідуваності:');
    for i:=1 to 24 do
        if abs(visits[i]-a) < a*p/100 then
            writeln(i)
    {розгалуження в циклі є одним оператором,
    тому операторні дужки "begin" та "end" не потрібні}
end.
```

Програма на C++.

Розробники мови програмування ***C*** Деніз Рітчі та Кен Томпсон акцентували свою увагу на ефективній і швидкій роботі програми з ресурсами комп'ютера. Масиви тут, – це фізична ділянка пам'яті, розділена на «порції» а механізм індексації побудований на зміщенні вказівника зчитування до наступної «порції». Сам же доступ до даних використовує «***покажчики***» (***pointers***), що містять адреси пам'яті, де розташовані дані. Цей же підхід до опрацювання збережено і в ***C++.***

Ділянка пам'яті під масив виділяється вказанням її розміру, тобто кількості елементів та їх типу за допомогою простої інструкції оголошення змінної:

```
<element-type> <variable-name> [<count>];
```

Опис масиву вказує компілятору виділити ділянку пам'яті для зберігання вказаної кількості (***count***) елементів заданого типу (***element-type***). Як і інші інструкції опису в ***C/C++,*** оголошення масиву можна розмістити у будь якому місці робочого коду, але до того, як оголошений масив буде використовуватися програмою.

Звертання до оремих елементів масиву здійснюється задананням (в квадратних дужках біля імені масиву) зміщення на вказану кількість елементів від початку ділянки пам'яті, виділеної під масив. Перший елемент лежить на початку ділянки, тому для доступу до нього зміщення не потрібне, тобто дорівнює 0. Щоб працювати з другим елементом масиву, потрібно зміститися на 1 елемент від початку даних, з третім, – на 2 і т.д. Цим пояснюється загальновідоме формулювання: «***елементи масиву в C/C++ індексуються з нуля***».

Контроль за виділенням пам'яті для зберігання масиву покладається на компілятор, тому розмір масиву повинен бути відомий ще до компіляції і не змінюватися під час роботи програми. Тому для задання розмірів масиву в стандартах мов ***C*** та ***C++*** дозволяється використовувати лише цілочисельні константи або цілочисельні літерали.

Підсумовуючи сказане вище, сформулюємо такі правила :

- ✓ *оголошення масиву устворюється з типу елементів масиву, його імені, за яким в квадратних дужках вказується кількість елементів;*
- ✓ *кількість елементів масиву можна задати лише цілочисельним літералом або іменованою константою цілочисельного типу;*
- ✓ *оголошувати масиви можна у буль-якому місці програми, але до початку їх використання;*
- ✓ *доступ до елемента масиву здійснюється за іменем масиву з вказанням індексу елемента в квадратних дужках після імені масиву;*
- ✓ *індекси елементів утворюються з послідовних цілих чисел, починаючи з 0, тобто, індекс першого елемента масиву дорівнює 0, а індекс останнього елемента на одиницю менший за загальну кількість елементів масиву.*

Застосування цих правил продемонструємо на програмі для розв'язування задачі 300:

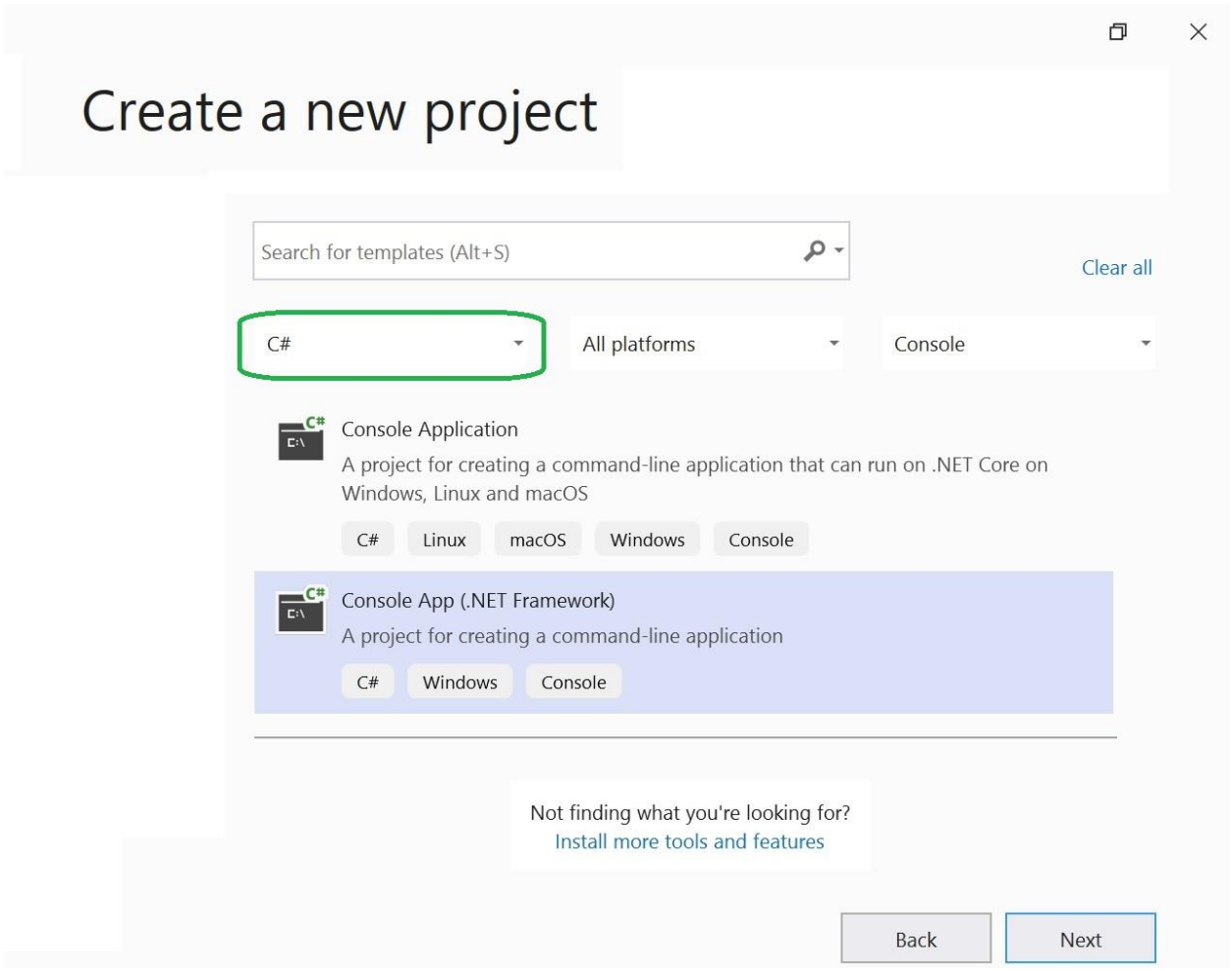
```
#include <iostream>

using namespace std;


int main()

{

    int visits[24]; //оглошуємо масив

    cout<<"Ведіть відвідуваність сайту ";

    cout<<"кожної години протягом доби";
```

```
    //вводимо елементи масиву в циклі
    for (int i = 0; i < 24; i++)
    {
        cin >> visits[i];
    }
    int s = 0;
    //цикл по елементах списку
    for (int i = 0; i < 24; i++)
    {
        s += visits[i];
    }
    double a = double(s) / 24;
    printf("Середньодобова відвідуваність %10.2f", a);
    cout<<"Задайте коефіцієнт відхилення у % :";
    double p;
    cin>>p;
    printf("Години з менш, ніж %5.2f -відсотковим", p);
    printf("відхиленням від середньої відвідуваності:");
    for (int i = 0; i < 24; i++)
    {
        if (abs(visits[i] - a) < a * p / 100)
        {// Години номеруємо 1..24, тому i+1
            cout << i + 1 << endl;
        }
    }
}
```

Програма на C#.

Серед розв'язків цієї задачі розглянемо приклад на ***C#***. Насамперед з огляду на те, що платформа ***Microsoft.net*** внесла у роботу з масивами в ***C#*** принципові особливості, у порівнянні з ***C*** та ***C++***. Окрім цього, для програмування на ***C#*** традиційно використовується розглянуте в попередньому розділі інтегроване середовище ***Microsoft Visual Studio***. Процес створення

програми відрізняється від створення проєктів на ***C++*** чи Python хіба що вибором мови програмування у вікні створення проєкту.



Проєкт типу ***Console Application*** використовує випущену в 2016 компанією Microsoft платформу нового покоління ***.net Core***. Вона є кросплатформною та може працювати також з операційними системами сімейства ***Linux*** та ***MacOS***. Проект Console App (.Net Framework) використовує класичну платформу .net, яка працює тільки на комп'ютерах з операційною системою ***Windows***.

Для наших задач можна обирати будь-який із двох запропонованих ***Visual Studio*** шаблонів консольного проєкту. В особливості їх структури вдаватися не будемо, в якості основної програми використовуватимо метод Main автоматично створеного ***Visual Studio*** класу ***Program***:

```
namespace Task300
{
    class Program
    {
        static void Main(string[] args)
```

```
        {// Тут записуватимемо код нашої програми
        }
    }
}
```

Щодо роботи з масивами в ***C#***, то тут, в цілому, збережено С-синтаксис звертання до елемента масиву за рахунок індекса в квадратних дужках. А також загальні правила індексування елементів:

- ✓ *перший елемент масиву має індекс 0,*
- ✓ *для індексування використовуються послідовні цілі числа,*
- ✓ *індекс останнього елемента масиву на одиницю менший за кількість усіх елементів.*

Проте в плані створення масиву, тобто, виділення пам'яті для зберігання елементів масиву, ***C#*** принципово відрізняється від своїх «старших сестер» ***C*** та ***C++***, – тут всім займається не компілятор, а віртуальна машина (CLR). Вона працює з масивами як з об'єктами спеціального виду і створює їх вже під час виконання програми. Тут оголошення масиву не гарантує його ініціалізації, а для виділення пам'яті слід використовувати спеціальний «конструктор» масиву. Оголошення та створення (ініціалізацію) масиву можна поєднати в такій синтаксичній конструкції вигляду:

```
<array-type> [] <array-name> = new <array-type>[count];
```

Квадратні дужки у лівій частині оператора присвоєння повинні бути порожніми, їх завдання вказати компілятору що змінна з іменем ***array-name*** не є одинарною змінною, а саме масивом. Конструкція зліва від знака рівності вказує компілятору, що в програмі буде використовуватися змінна з іменем ***array-name***, яка є масивом з елементів типу ***array-type***. Інструкція справа від знака рівності, навпаки, виконує конкретну дію: викликає конструктор для створення у пам'яті віртуальної машини об'єкта, що буде утворений як послідовність ***count*** елементів типу ***array-type***.

Обидві частини розглянутої вище синтаксичної конструкції можна використовувати незалежно одна від одної. Але таке поєднання оголошення масиву з його ініціалізацією дозволить нам зберегти в програмі на ***C#*** стиль роботи з масивами, аналогічний до попередніх програм на ***C++*** та ***Pascal***.

```
using System;
namespace Task300
{
    class Program
    {
        static void Main(string[] args)
        {
            //оглошуємо масив та створюємо його
            int [] visits = new int [24];
            Console.WriteLine("Ведіть відвідуваність " +
                "сайту кожної години протягом доби");
            //вводимо елементи масиву в циклі
            for (int i = 0; i < 24; i++)
            {
                visits[i] = Convert.ToInt32(
                                Console.ReadLine());
                //дані з консолі вводяться у формі
                // тексту, тому їх слід конвертувати
                // у відповідний числовий тип
            }
            int s = 0;
            //цикл по елементах списку
            for (int i = 0; i < 24; i++)
            {
                s += visits[i];
            }
            double a = (double)s / 24;
            //приводимо s до дійсного типу
            //інакше буде виконуватися ділення націло
            Console.WriteLine("Добова відвідуваність " +
             a + " відвідувачів на годину.");
            Console.WriteLine("Задайте відхилення у %");
            //оголошену змінну ініціалізуємо даними,
```

```
            // отриманими з консолі
            double p  = Convert.ToDouble(
                                Console.ReadLine());
            Console.WriteLine("Години з менш, ніж "+ p +
                      "-відсотковим відхиленням від " +
                        "середньої відвідуваності:");
            for (int i = 0; i < 24; i++)
            {
                if(Math.Abs(visits[i] - a) < a * p / 100)
                {// години номеруємо 1..24
                    Console.WriteLine(i + 1);
                }
            }
        }
    }
}
```

Програма на Python.

У мові програмування ***Python*** така форми структурування даних, як масив відсутня. Тобто, в класичному розумінні, як у прикладах на інших мовах програмування вище, коли пам'ять виділяється одразу для зберігання усіх елементів масиву. Тут для зберігання індексованих сукупностей використовують ***списки***. Основна відмінність між ***списком*** і ***масивом*** у можливості «дописувати» елементи до списку в ході виконання програми (або вилучати). Проте сам механізм доступу до елемента в списку за його індексом практично нічим не відрізняється від масиву.

В програмах на ***Python*** часто використовують списки, проініціалізовані попередньо деяким набором елементів. Для цього використовують синтасичну конструкцію вигляду:

```
<list-name> = [<value1>, <value2>, …, <valueN>]
```

яка створює список з N елементів і заповнює його значеннями, вказаними в квадратних дужках. У випадку нашої задачі можна б було оголосити змінну для масиву, ініціалізувавши її 24-ма нулями, але ми будемо створювати порожній (без жодного елемента) список і додавати в нього елементи по мірі вводу даних функцією ***append***.

Оголосити порожній список можна, залишивши в розглянутій вище конструкції квадратні дужки порожніми. В коді далі для цього використано функцію ***array*** з модуля ***array***, це дозволяє при оголошення списку

зафіксувати тип елементів. При цьому типи даних можна, зокрема, позначати такими символами: 'b', 'h', 'i', 'l', 'q', якщо елементи масиву – цілі числа, 'f', 'd', якщо елементи масиву дійсні числа, 'u' – символьні дані в Unicode.

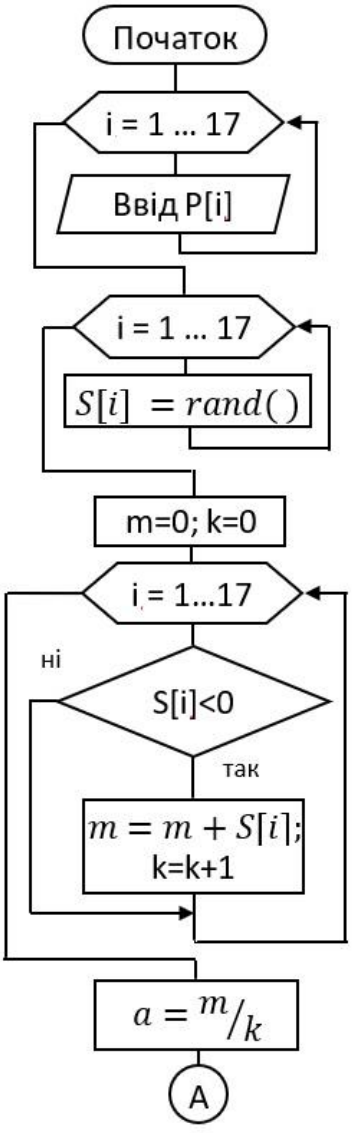
Код програми для розв’язування задачі 300 на мові ***Python***:

```
import array
visits = array.array('i')#створюємо порожній список
print('Ведіть відвідуваність сайту', end=" ")
print('кожної години протягом доби')
for i in range(0,24):
    #дописуємо елементи в список
    visits.append(int(input()))
s = 0
# цикл по елементрах списку
for k in visits :
    s+=k
a = s / 24
print('Середньодобова відвідуваність: %10.2f' % a)
print('Задайте коефіцієнт відхилення у % :')
p = float(input())
print('Години з не більше ніж %5.2f -відсотковим' % p)
print('відхиленням від середньої відвідуваності:')
for i in range(0,24):
    if abs(visits[i]-a)<a*p/100 :
        print(i + 1) # Години номеруємо 1..24
```

330. Утворити три масиви дійсних чисел P, Q та S по 17 елементів кожен. Масив P заповнює користувач, вводячи його елементи з клавіатури. Масив Q заповнити випадковими числами з відрізка [-7; 7]. Масив S утворити з елементів масиву P, додавши до кожного середнє арифметичне від'ємних елементів масиву Q. Вивести масиви на екран, в кожному масиві знайти номер елемента найближчого за величиною до числа 5.

Алгоритм розв'язування цієї задачі буде багатим на ***циклічно-розгалужені конструкції***. Вже для утворення масиву S потрібно знайти середнє арифметичне від'ємних елементів масиву Q, а його обчислення буде містити перевірку від'ємності елемента в тілі циклу. Обчислення в кожному масиві елемента найближчого за величиною до числа 5 реалізується також за допомогою розгалуження в циклі.

Дані задачі представлені у вигляді трьох масисвів, кожен з них потрібно заповнити елементами, а потім в кожному виконати вказаний в умові пошук. Ці дії можна групувати в циклах по різному. Ми зупинимося на варіанті алгоритму, в якому кожен масив опрацьовується в окремому циклі, об'єднаємо, хіба що, вивід масиву на екран з пошуком елемента найближчого за величиною до числа 5.



Алгоритм розв'язування задачі міститиме такі цикли:

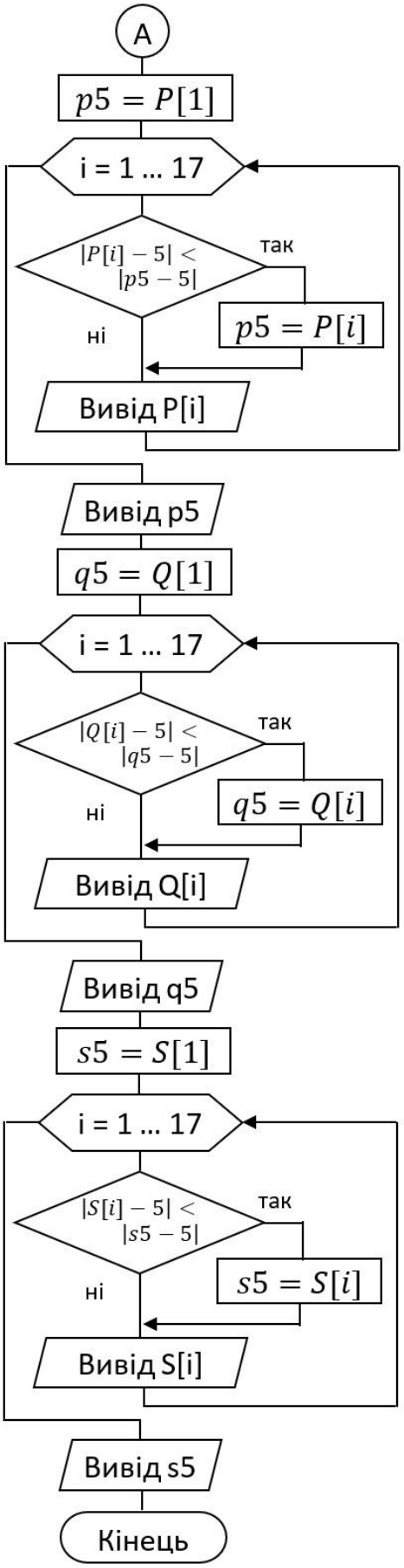
- *цикл для вводу елементів масиву P;*
- *цикл для генерування елементів масиву Q;*
- *цикл для обчислення середнього арифметичного елементів масиву Q;*
- *цикл для заповнення масиву S;*
- *цикл для пошуку елемента масиву P, найближчого за величиною до числа 5, в якому одночасно виведемо усі елементи масиву на екран;*
- *цикл для пошуку елемента масиву Q, найближчого за величиною до числа 5, в якому одночасно виведемо усі елементи масиву на екран;*
- *цикл для пошуку елемента масиву S, найближчого за величиною до числа 5, в якому одночасно виведемо усі елементи масиву на екран.*

Після кожного з трьох останніх циклів виведемо значення знайденого у відповідному масиві елемента. Більш детально дії алгоритму розв'язування задачі подано у блок-схемі на малюнку.

Програма на Pascal.

Базові концепції створення та опрацювання масивів програмування на ***Pascal*** розглянуто вище, при обговоренні розв'язування задачі 300. Нагадаємо лише, що генератор випадкових чисел, представлений в ***Pascal*** функцією ***random(L)*** видає випадкову послідовність цілих чисел з діапазону $[0;L)$, а «дивний» запис в циклі формування елементів масиву Q в коді нижче є спробою спроєктувати цей діапазон на $[-7;7)$ та отримати випадкові числа дійсного типу, з відмінною від нуля дробовою частиною. Решту деталей пропонуємо розглянути безпосередньо в коді програми далі.

```
program Task330;
var P, Q, S: array[1..17] of real;
    i, k: integer;
    m, a, p5, q5, s5: real;
begin
writeln('Введіть масив P');
randomize();{ініціюємо генератор
випадкових чисел}
for i:= 1 to 17 do
    readln(P[i]);
for i:= 1 to 17 do
    Q[i] := random(1400)/100 - 7;
m := 0.0; k := 0;
for i:= 1 to 17 do
    if Q[i] < 0 then
    begin
```



```
        m := m + Q[i];
        k := k + 1
    end;
    a := m / k;
for i:= 1 to 17 do
    S[i] := P[i] + a;
{Вибираємо перший елемент за найближчий до 5}
p5 := P[1];
writeln('Масив P');
for i:= 1 to 17 do
begin
    write(P[i]:8:2);
    if abs(P[i] - 5) < abs(p5 - 5) then p5 := P[i];
end;
writeln(); {Перехід на новий рядок}
writeln('Найближче до 5 число ', p5:8:2);
q5 := Q[1];
writeln('Масив Q');
for i:= 1 to 17 do
begin
    write(Q[i]:8:2);
    if abs(Q[i] - 5) < abs(q5 - 5) then q5 := Q[i];
end;
writeln(); {Перехід на новий рядок}
writeln('Найближче до 5 число ', q5:8:2);
s5 := S[1];
writeln('Масив S');
for i:= 1 to 17 do
begin
    write(S[i]:8:2);
```

```
    if abs(S[i] - 5) < abs(s5 - 5) then s5 := S[i];
end;
writeln(); {Перехід на новий рядок}
writeln('Найближче до 5 число ', s5:8:2)
end.
```

Програма на C++.

Детальний аналіз особливостей використання масивів в ***C++***, викладений в поясненні до прикладу розв'язування задачі 300 дозволяє нам подати тут лише програмний код реалізації сформульованого вище алгоритму.

```
#include <iostream>
#include <math.h>
using namespace std;
void main()
{
    srand(time(0));// ініціюємо генератор випадкових чисел
    double P[17], Q[17], S[17];
    cout << "Enter array P:";
    for (int i = 0; i < 17; i++)
    {
        cin >> P[i];
    }
    for (int i = 0; i < 17; i++)
    {
        Q[i] = double(rand() % 1400) / 100 - 7;
    }
    double s = 0;
    int count = 0;
    for (double d: Q)// цикл для обчислення середнього
        if (d < 0)    // арифметичного від'ємних елементів
        {
            s += d; count++;
```

```
    }
double a = s / count;
for (int i = 0; i < 17; i++)
{
    S[i] = P[i] + a;
}
double p5 = P[0];
cout << "Array P:\n";
for (int i = 0; i < 17; i++)
{
    cout << P[i] << '\t';
    if (abs(P[i] - 5) < abs(p5 - 5))
        p5 = P[i];
}
cout << "\nClosess to 5 is " << p5 << endl;
double q5 = Q[0];
cout << "Array Q:\n";
for (int i = 0; i < 17; i++)
{
    cout << Q[i] << '\t';
    if (abs(Q[i] - 5) < abs(q5 - 5))
        q5 = Q[i];
}
cout << "\nClosess to 5 is " << q5 << endl;
double s5 = S[0];
cout << "Array S:\n";
for (int i = 0; i < 17; i++)
{
    cout << S[i] << '\t';
    if (abs(S[i] - 5) < abs(s5 - 5))
```

```
            s5 = S[i];
    }
    cout << "\nClosess to 5 is " << s5 << endl;
}
```

Програма на Java.

Організація масивів в ***Java*** майже аналогічна до ***C#***, тобто, в цілому, підтримується С-синтаксис звертання до елемента в масиві з допомогою індекса в квадратних дужках, та загальні правила індексування елементів:

- ✓ *перший елемент масиву має індекс 0,*
- ✓ *для індексування використовуються послідовні цілі числа,*
- ✓ *індекс останнього елемента масиву на одиницю менший за кількість усіх елементів.*

Створенням масиву, тобто, виділення пам'яті для зберігання його елементів в ***Java*** (так само, як в ***C#***) займається не компілятор, а віртуальна машина (***Java Virtual Machine - JVM***). Вона працює з масивами як з об'єктами спеціального вигляду і створює їх вже під час виконання програми. Оголошення масиву не гарантує його ініціалізації, тобто виділення пам'яті для зберігання елементів, для цього слід використати «конструктор» масиву зі службовим словом ***new*** та вказати кількість елементів:

```
<array-name> = new <array-type>[count];
```

Оголошення ж масиву, на відміну від ***C#*** можна записувати двояко:

```
<array-type> [] <array-name>;
```

та так, як в ***C#*** та подібно до ***C++*** :

```
<array-type> <array-name> [];
```

квадратні дужки після імені масиву повинні бути порожніми, їх завдання лише вказати компілятору що змінна є масивом, сам же масив буде створено, лише після ініціалізації змінної з допомогою оператора ***new***. Такий підхід дозаоляє, зокрема, перестворювати масисви під час виконання коду, чого не допускаається (стосовно класичних масивів) в ***C*** та ***C++***.

Тут, як і коді на ***C#*** будемо суміщати оголошення та ініціалізацію масиву:

```
<array-type> [] <array-name> = new <array-type>[count];
```

Така конструкція дозволить нам не тільки зберегти стиль роботи з масивами, аналогічний до ***C++*** та ***Pascal***, але й уникнути потенційних помилок при звертанні до елементів не ініціалізованого (не створеного) масиву. Решта особливостей роботи з масивами в ***Java*** пропонуємо розглянути в наступному коді:

```
package net.starbasic.tasks.p4;
import java.util.Scanner;

public class Task330 {
    public static void main(String [] args)
    {
        Scanner scanner = new Scanner(System.in);
        //в оголошенні можна створити декілька масивів
        double P[] = new double[17], Q[] = new double[17];
        double[] S = new double[17];//але зазвичай, окремо
        //оголошення double[] S;  та double S[]; тотожні
        System.out.println("Введіть елементи масиву P");
        for (int i = 0; i < 17; i++)
        {
            P[i] = scanner.nextDouble();
        }
        for (int i = 0; i < 17; i++)
        {
            Q[i] = Math.random() * 1400 / 100 - 7;
        }
        double s = 0;
        int count = 0;
        for (double d: Q) // Цикл «for each» за елементами
            if (d < 0)    // масиву переглядає усі елемен-
            {             // ти масиву Q, поміщаючи їх
                s += d;   // по черзі у змінну d
                count++;
            }
        double a = s / count;
        for (int i = 0; i < 17; i++)
```

```
{
    S[i] = P[i] + a;
}
double p5 = P[0];
System.out.println("Масив P:");
for (int i = 0; i < 17; i++)
{
    System.out.printf(" %8.2f", P[i]);
    if (Math.abs(P[i] - 5) < Math.abs(p5 - 5))
        p5 = P[i];
}
System.out.println();//перехід на нову стрічку
System.out.printf("Найближче до 5: %10.2f \n", p5);
double q5 = Q[0];
System.out.println("Масив Q:");
for (int i = 0; i < 17; i++)
{
    System.out.printf(" %8.2f", Q[i]);
    if (Math.abs(Q[i] - 5) < Math.abs(q5 - 5))
        q5 = Q[i];
}
System.out.println();//перехід на нову стрічку
System.out.printf("Найближче до 5 %10.2f \n", q5);
double s5 = S[0];
System.out.println("Масив S:");
for (double x : S)
{
    System.out.printf(" %8.2f", x);
    if (Math.abs(x - 5) < Math.abs(s5 - 5))
        s5 = x;
```

```
        }
        System.out.println();//перехід на нову стрічку
        System.out.printf("Найближче до 5: %10.2f \n", s5);
    }
}
```

Програма на Python.

В реалізації сформульованого алгоритму на ***Python*** будемо користуватися тими ж прийомами використання списків, що й в прикладі до задачі 300. Код програми для розв'язування задачі 330 на мові ***Python***:

```
import array # для оголошення списків
import random # модуль для випадкових чисел
P = array.array('d')
Q = array.array('d')
S = array.array('d')
print('Введіть елементи масиву P')
for i in range(0,17):
    P.append(float(input()))
for i in range(0,17):
    Q.append(round(random.uniform(-7.0,7.0),2))
m = 0.0
k = 0
# Цикл «for each» на Python виглядає так:
for d in Q: #шукаємо середнє
    if d<0: #лише від'ємних елементів
        m+=d
        k+=1
a = m/k
for i in range(0,17):
    S.append(P[i]+a)
#Вибираємо перший елемент за найближчий до 5
P5 = P[0]
```

```
print('Масив P')
# цикл «for each» переглядає усі елементи масиву P,
# поміщаючи їх по черзі у змінну d
for d in P:
    print(" %8.2f" % d, end=' ')
    if(abs(d - 5) < abs(P5 - 5)):
        P5 = d
print() #Перехід на новий рядок
print('Найближче до 5 число %8.2f' % P5)

Q5 = Q[0]
print('Масив Q')
for d in Q:
    print(" %8.2f" % d, end=' ')
    if(abs(d - 5) < abs(Q5 - 5)):
        Q5 = d
print() #Перехід на новий рядок
print('Найближче до 5 число %8.2f' % Q5)

S5 = S[0]
print('Масив S')
for d in S:
    print(" %8.2f" % d, end=' ')
    if(abs(d - 5) < abs(S5 - 5)):
        S5 = d
print() #Перехід на новий рядок
print('Найближче до 5 число %8.2f' % S5)
```

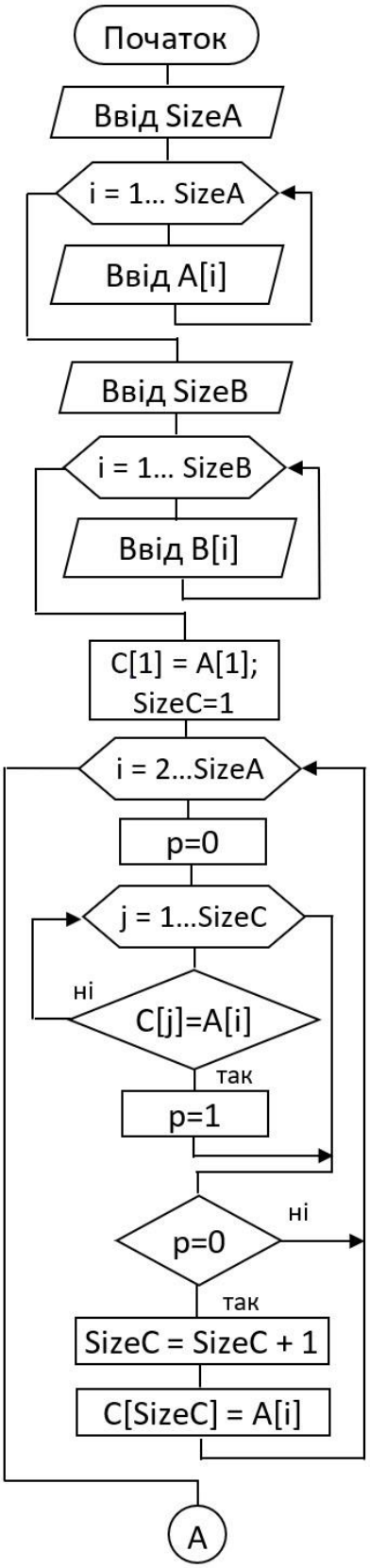
360. Задано два масиви цілих чисел A та B. Створити з їх елементів масив C за правилом об'єднання двох множин: кожен елемент масиву A і кожен елемент масиву B повинен входити в результуючий масив, причому лише один раз.

Задачі 345 – 360 цього розділу мають певну специфіку, яка проявляється в деякій невизначеності в самій умові. Наприклад, тут не вказано який розмір масивів A та B, і навіть, якщо задати деяку константу для вказання кількості елементів в масивах, то виникне закономірне питання: масиви повинні мати однаковий ромір, чи ні?

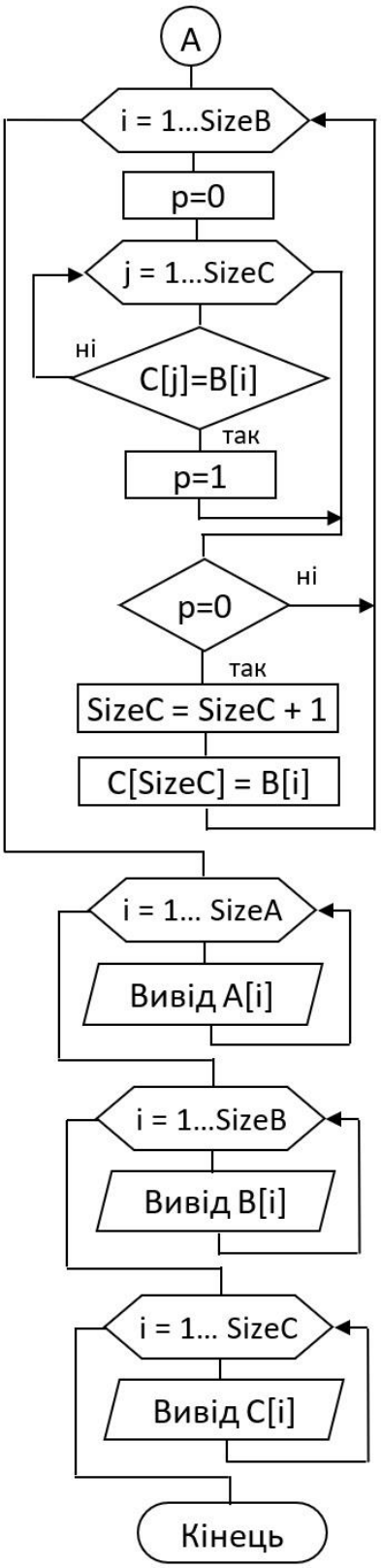
В такій, на першій погляд, невизначеності немає нічого поганого, вона є місцем для креативу розробника програми. В практиці ІТ таких задач переважна більшість, бо розробники отримують від замовника лише побажання до функціоналу програмного продукту а не конкретне технічне завдання. Претендентам на вакансії спеціально дають завдання подібного характеру, з метою оцінити аналітичні здібності кандидата та його здатність до самостійного прийняття рішень.

В нашій задачі приймемо такі домовленості:

- ✓ *оскільки не вказано як саме задані масиви, вважатимемо, що їх елементи вводитиме користувач з клавіатури;*
- ✓ *оскільки не вказано розмір масивів* A *та* B*, вважатимемо, що масиви різних розмірів, – однаковий розмір масивів тоді буде часковим випадком задачі;*
- ✓ *оскільки* A *та* B *– масиви, то можуть містити однакові елементи, чого, за умовою задачі, не повинно бути в результуючому масиві;*
- ✓ *оскільки розмір результуючого масиву стане відомий лише у ході виконання програми, то розміри масивів також будемо дізнаватися від користувача програми під час її виконання.*



Останнє уточнення до умови задачі змушує нас замислитися над засобами реалізації потрібного функціоналу. Справа в тому, що «класичний» масив повинен «знати» кількість своїх елементів



вже під час створення, щоб тут зарезервувати їм місце в пам'яті. Створюються «класичні» масиви на етапі запуску програми, як, наприклад в програмах на Pascal, чи C/C++. Для вирішення цієї проблеми, зазвичай, створюють масив «з запасом», тобто значно більшого розміру, а використовують лише початкові його елементи у потрібній кількості. Інший підхід, – використання спеціальних структур даних, – ***списків***, які допускають додавання елементів під час виконання програми. Функціонал ***списку*** ми вже використовували в прикладах до попередніх задач на мові ***Python***, оскільки в Python відсутні «класичні» масиви.

Наш вибір між двома зазначеними вище підходами залежатиме від можливостей тієї чи іншої технології програмування. А от сам алгоритм розв'язування задачі буде однозначним:

- *програма запитує в користувача розмір масиву A;*
- *за заданим розміром програма організовує цикл, в якому отримує від користувача вказану кількість елементів масиву A;*
- *програма запитує в користувача розмір масиву B;*
- *за заданим розміром програма організовує цикл, в якому отримує від користувача вказану кількість елементів масиву B;*
- *програма записує в результуючий масив перший елемент масиву A;*
- *для кожного наступного елемента масиву A програма організовує цикл порівняння з доданими раніше в результуючий масив C елементами;*
- *якщо такий елемент в масиві C знаходиться, то порівняння припиняється і програма переходить до наступного елемента масиву A;*
- *якщо такого елемента ще нема в C, то дописує його в результуючий масив;*
- *кожен елемента масиву B програма в циклі порівнянює з доданими раніше в результуючий масив C елементами;*
- *якщо такий елемент в масиві C знаходиться, то порівняння припиняється і програма переходить до наступного елемента масиву B;*
- *якщо такого елемента ще нема в масиві C, то дописує його;*
- *програма виводить усі елементи масиву A;*

- *програма виводить усі елементи масиву B;*
- *програма виводить усі елементи результуючого масиву C.*

До сформульованого вище словесного опису алгоритму традиційно додаємо графічну інтерпретацію в формі блок-схеми.

Програма на Pascal.

В програмі на ***Pascal*** скористаємося традиційними масивами та першим підходом, тобто оголосимо масиви достатньо великих розмірів, наприклад по 1000 елементів, сподіваючись, що користувач не буде вводити надто багато чисел. Будемо використовувати додаткові змінні ***sizeA*** та ***sizeB***, що зберігатимуть фактичний розмір масивів. Їх значення вказуватиме користувач, а програма буде використовувати як максимальне значення індекса елемента відповідного масиву.

Природньо вважати, що максимально можлива кількість елементів масиву C буде рівна сумі розмірів обидвох масивів. В коді нижче розмір результуючого масиву оголошено як 2000. Справжню ж кількість записаних в результуючий масив елементів підраховуємо по мірі додавання елементів в масив, зберігаючи її в змінній ***sizeC***. Після завершення процесу наповнення масиву C ця змінна буде містити фактичний розмір цього масиву.

В якості «прапорця» для позначення того, чи знайшовся елемемент в результуючому масиві використовуємо цілочисельну змінну ***present***, якій присвоюємо нульове значення на початку порівняння і змінюємо його на одиницю, якщо потрібний елемент знайдено. Таким чином, якщо після циклу змінна ***present*** зберегла нульове значення, це означає, що такого елемента в результуючому масиві ще немає і його потрібно додати.

Для дострокового переривання ітерацій циклу в ***Pascal*** використовується оператор ***break***. З рештою подробиць ***Pascal***-реалізації розглянутого вище алгоритму пропонуємо ознайомитися в поданому далі програмному коді:

```
program TASK360;

var sizeA, sizeB, sizeC, i, j: integer;
    A, B : array [1..1000] of integer;
    C : array [1..2000] of integer;
    present : integer;
begin
    writeln('Введіть розмір масиву A');
    readln(sizeA);
    writeln('Введіть елементи масиву A');
    for i := 1 to sizeA do readln(A[i]);
```

```
writeln('Введіть розмір масиву B');
readln(sizeB);
writeln('Введіть елементи масиву B');
for i := 1 to sizeB do readln(B[i]);
sizeC := 0;{в sizeC будемо підраховувати кількість
    елементів доданих в масив C   }
C[1] := A[1];
sizeC := sizeC + 1; {збільшуємо значення лічильника}
for i := 1 to sizeA do
begin
    present := 0;{ встановлюємо значення 0 для
    чергового елемента, якщо після циклу це значення
    залишиться рівним 0, це буде означати, що такого
    елемента в результуючому масиві ще немає}
    for j := 1 to sizeC do
        if A[i]=C[j] then
        begin
            present := 1;
            break
        end;
    {внутрішній цикл по j містить в тілі лише один
    оператор розгалуження, begin-end не використовуємо,
    тут закінчується  тіло цього циклу}
    if present = 0 then
    begin
        sizeC := sizeC + 1;
        C[sizeC] := A[i]
    end
end;
{Перевіряємо котрі з елементів масиву B
```

```
    потрібно записати в масив C}
    for i := 1 to sizeB do
    begin
        present := 0;
        for j := 1 to sizeC do
            if B[i]=C[j] then
            begin
                present := 1;
                break
            end; {тут закінчується тіло циклу по j}
        if present = 0 then
        begin
            sizeC := sizeC + 1;
            C[sizeC] := B[i]
        end
    end;
    writeln('Масив A:');
    for i:=1 to sizeA do write(A[i]:8);
    writeln();
    writeln('Масив B:');
    for i:=1 to sizeB do write(B[i]:8);
    writeln();
    writeln('Масив C:');
    for i:=1 to sizeC do write(C[i]:8);
    writeln();
end.
```

Програма на C++.

Технології програмування на ***C*** та ***C++*** мають можливість використовувати динамічний розподіл пам'яті за допомогою механізму, що дозволяє створювати масиви під час виконання програми. Цей підхід доволі складний, тому, залишимо його реалізацію для більш обізнаного читача. Реалізація алгоритму з використанням звичайних масивів оголошених з «запасом»

розміру буде аналогічною до поданої вище програми на ***Pascal***, з відмінностями хіба що в синтаксичних конструкціях відповідних команд, тому її ми також не подаємо. Спробуємо знайти потрібний функціонал у реалізованому в ***C++*** об'єктно-орієнтованому піхдоді.

Як кожна об'єктно-орієнтована технологія, ***C++*** має можливість використовувати бібліотеки стандартних класів. Для зберігання та опрацювання різноманітних наборів даних в ***C++*** розроблена спеціальна бібліотека, – ***Standard Template Librarry*** (***STL***). Вона містить узагальнені шаблонні класи-контейнери для роботи зі списками, множинами, відображеннями, тощо.

Скористаємося в нашій програмі класом ***vector***, що є однією з найпростіших реалізацій індексованого списку елементів. Цей клас, як і інші є представники ***STL*** є шаблонним, тобто реалізований в термінах узагальненого програмування і може створюватися з елементів будь-якого типу. Обраний тип елементів вектора вказується при оголошенні змінної у кутових дужках після назви класу, наприклад ***vector<int>*** – задає список, елемнетами якого є цілі числа.

Клас ***vector*** має можливість звертатися до своїх елементів за їх індексами в квадратних дужках, як до елементів масиву, функцію ***push_back*** для дописування елемента в кінець списку, а також функцію ***size*** для визначення кількості доданих елементів. Цього функціоналу нам цілком достатньо для реалізації описаного вище алгоритму.

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector<int> A, B, C;
    int sizeA, sizeB;
    cout << "Enter size of array A:" << endl;
    cin >> sizeA;
    cout << "Enter array A elements:" << endl;
    for (int i = 0; i < sizeA; i++) {
        int ai;//змінна для вводу елемента масиву A
        cin >> ai;//зчитування значення
        A.push_back(ai);//додавання в список
    }
```

```
cout << "Enter size of array B:" << endl;
cin >> sizeB;
cout << "Enter array B elements:" << endl;
for (int i = 0; i < sizeB; i++) {
    int bi;//змінна для вводу елемента масиву B
    cin >> bi;//зчитування значення
    B.push_back(bi);//додавання в список
}
// записуємо в список перший елемент A
C.push_back(A[0]);/* vector допускає
доступ до елементів за індексом в [ ] */
/* Переглядаємо решту елементів масиву A
метод size() визначає кількість елементів */
for (int i = 1; i < A.size(); i++) {
    bool present = false;
    for (int el : C) {// цикл for each дозволяє пере-
    // глянути усі елементи контейнера в т.ч. масиву
        if (el == A[i]) {
            present = true;
            break;
        }//завершуємо перевірку достроково
        //якщо знаходимо такий елемент
    }
    if (!present) C.push_back(A[i]);
}
//Перевіряємо котрі з елементів масиву B
//потрібно дописати до списку C
for (int i = 0; i < B.size(); i++) {
    bool present = false;
    for (int el : C) {
```

```
            if (el == B[i]) {
                present = true;
                break;
            }
        }
        if (!present) {
            C.push_back(B[i]);
        }
    }
    cout << "Array A:" << endl;
    for (int el : A) cout << el << '\t';
    cout << endl;
    cout << "Array B:" << endl;
    for (int el : B) cout << el << '\t';
    cout << endl;
    cout << "Array C:" << endl;
    for (int el : C) cout << el << '\t';
    cout << endl;
}
```

Програма на C#.

Щодо технологій програмування, які використовують проміжний код та віртуальні машини, зокрема, ***Java*** та ***.Net (C#),*** то за створення «класичних» масивів тут якраз відповідає віртуальна машина, а не компілятор. Віртуальна машина виділяє місце для зберігання елементів масиву саме під час виконання програми, тому послідовність: запитати в користувача розмір масиву, в потім створити в пам'яті масив відповідного розміру, тут є цілком природною. Тобто, масиви в ***C#*** є динамічними, їх можна створювати (перестворювати) за потребою. Для вхідних масивів A та B будемо використовувати цей підхід.

Для результуючого масиву, з огляду на постійне його доповнення новими елементами, зручніше скористатися списком. В ***C#*** для роботи зі списками можна використовувати узагальнений клас ***List<T>***, що за функціональністю дуже схожий на клас ***vector<T>*** з ***STL*** в ***C++***. Він дозволяє дописувати

елементи в кінець списку, звертатися до них за індексом в квадратних дужках і. т.п.

З особливостями обраного способу реалізації алгоритму пропонуємо ознайомитися в поданому нижче коді.

```
using System;
using System.Collections.Generic;
namespace Task360
{
  class Program
  {
    static void Main(string[] args)
    {
      int []A, B;
      int sizeA, sizeB;
      Console.WriteLine("Введіть розмір масиву A");
      sizeA = Convert.ToInt32(Console.ReadLine());
      A = new int[sizeA];// CLR дозволяє створювати
            // масиви в ході виконання програми
      Console.WriteLine("Введіть елементи масиву A");
      for (int i = 0; i < sizeA; i++)
      {
        A[i] = Convert.ToInt32(Console.ReadLine());
      }
      Console.WriteLine("Введіть розмір масиву B");
      sizeB = Convert.ToInt32(Console.ReadLine());
      B = new int[sizeB];
      Console.WriteLine("Введіть елементи масиву B");
      for (int i = 0; i < sizeB; i++)
      {
        B[i] = Convert.ToInt32(Console.ReadLine());
      }
```

```
//Для масиву C використовуємо список
List<int> listC = new List<int>();
listC.Add(A[0]);// записуємо в список перший
//елемент A, переглядаємо решту його елементів
for (int i = 1; i < sizeA; i++)
{
  bool present = false;
  foreach(int el in listC)
  {
    if (el == A[i])
    {
      present = true;
      break;  //завершуємо перевірку достроково
    } //якщо такий елемент знайдено
  }
  if (!present)
  {
    listC.Add(A[i]);
  }
}
//Перевіряємо котрі з елементів масиву B
//потрібно дописати до списку C
for (int i = 0; i < sizeB; i++)
{
  bool present = false;
  foreach (int el in listC)
  {
    if (el == B[i])
    {
      present = true;
```

```
            break;
          }
        }
        if (!present)
        {
          listC.Add(B[i]);
        }
      }
      Console.WriteLine("Масив A:");
      for(int i = 0; i < A.Length; i++)
      {
        Console.Write($"{A[i]} ");
      }
      Console.WriteLine();
      Console.WriteLine("Масив B:");
      for(int i = 0; i <B.Length; i++)
      {
        Console.Write($"{B[i]} ");
      }
      Console.WriteLine();
      Console.WriteLine("Масив C:");
      for(int i = 0; i < listC.Count; i++)
      {
        Console.Write($"{listC[i]} ");
      }
      Console.WriteLine();
    }
  }
}
```

Програма на Java.

Подамо «переклад» реалізованого вище C#-коду на мову ***Java***. Код далі майже ідентичний, з точністю до назв класів та методів консольного вводу-виводу, оголошення масивів та домовленостей, щодо іменування методів та полів класів.

Більш глибокі розбіжності між ***Java*** та ***C#*** можна розгледіти, при уважному погляді на опрацювання списку. Параметром шаблону в ***.Net*** може бути будь-який тип даних, в т.ч. ***int***, – в ***Java*** цей тип даних належить до так званих ***примітивних типів*** і замість нього в якості типу елементів списку доводиться використовувати спеціальний клас-обгортку ***Integer***. Ще одна цікава особливість пов'язана з перевантаженням стандартних операторів, – в ***C#*** воно передбачене, тому звертання до окремого елемента списку можна виконати з допомогою індекса в квадратних дужках (***list[i]***). В ***Java*** перевантаження операоторів відсутнє, – отримувати елемент списку за індексом доводиться за допомогою методу (***list.get(i)***).

```
import java.util.*;
public class Task360 {
    public static void main(String[] args) {
        int A[], B[];    int sizeA, sizeB;
        Scanner scanner = new Scanner(System.in);
        System.out.println("Введіть розмір масиву A");
        sizeA = scanner.nextInt();
        A = new int[sizeA];// Java Virtual Machine
        //створює масиви  в ході виконання програми
        System.out.println("Введіть елементи масиву A");
        for (int i = 0; i < sizeA; i++) {
            A[i] = scanner.nextInt();
        }
        System.out.println("Введіть розмір масиву B");
        sizeB = scanner.nextInt();
        B = new int[sizeB];
        System.out.println("Введіть елементи масиву B");
        for (int i = 0; i < sizeB; i++) {
            B[i] = scanner.nextInt();
```

```
}
//Для масиву C використовуємо список
ArrayList<Integer> listC = new ArrayList<>();
listC.add(A[0]);// записуємо в список перший
//елемент A та переглядаємо решту елементів
for (int i = 1; i <sizeA; i++) {
    boolean present = false;
    for (Integer el: listC){
        if(el==A[i]) {
            present = true;
            break; //завершуємо перевірку
        } //якщо такий елемент вже знайдено
    }
    if(!present){
        listC.add(A[i]);
    }
}
for (int i = 0; i <sizeB; i++) {
    boolean present = false;
    for (Integer el: listC){
        if(el==B[i]) {
            present=true;
            break;
        }
    }
    if(!present){
        listC.add(B[i]);
    }
}
System.out.println("Масив A:");
```

```
        for (int i = 0; i < A.length; i++){
            System.out.printf("%5d",A[i]);
        }
        System.out.println();
        System.out.println("Масив B:");
        for (int i = 0; i < B.length; i++){
            System.out.printf("%5d",B[i]);
        }
        System.out.println();
        System.out.println("Масив C:");
        for (int i = 0; i < listC.size(); i++){
            System.out.printf("%5d",listC.get(i));
        }
        System.out.println();
    }
}
```

Програма на Python.

Нагадаємо, що для роботи з масивами в ***Python*** ми використовуємо списки, які при оголошенні можна створювати порожніми, а далі наповнювати їх елементами, дописуючи дані в кінець списку. Тобто, так само як і в реалізації на ***C++*** з використанням класу ***vector<T>***, програма на ***Python*** повина запитувати в користувача розмір кожного з вхідних масивів, організовуючи цикли для зчитування та додавання в список вказаної кількості чисел. Далі реалізовуються описані в алгоритмі цикли для наповнення результуючого масиву унікальними елементами масиву A, а потім елементами масиву B, яких ще немає в результуючому масиві.

За лаконічністю коду ***Python***–програма знову перемагає інші подані тут програмні реалізації:

```
import array # модуль для оголошення списків
#оголошуємо масив (список) A та заповнюємо
#його введеними користувачем числами
A = array.array('d')
print('Введіть розмір масиву A')
size_a = int(input())
```

```
print('Введіть елементи масиву A')
for i in range(0,size_a):
    A.append(float(input()))
#оголошуємо масив (список) B та заповнюємо
#його введеними користувачем числами
B = array.array('d')
print('Введіть розмір масиву B')
size_b = int(input())
print('Введіть елементи масиву B')
for i in range(0,size_b):
    B.append(float(input()))
#оголошуємо результуючий масив (список)
#та організовуємо його заповнення
C = array.array('d')
C.append(A[0])
for i in range(1,size_a):
    present = False
    for el in C:
        if el == A[i]:
            present = True
            break
    if not present:
        C.append(A[i])
for i in range(0,size_b):
    present = False
    for el in C:
        if el == B[i]:
            present = True
            break
    if not present :
```

```
        C.append(B[i])
#Виводимо масиви
print('Масив A')
for el in A:
    print(" %8i" % el, end=' ')
print() #Перехід на новий рядок
print('Масив B')
for el in B:
    print(" %8i" % el, end=' ')
print() #Перехід на новий рядок
print('Масив C')
for el in C:
    print(" %8i" % el, end=' ')
print() #Перехід на новий рядок
```

Розділ 5. Опрацювання двовимірних масивів

Двовимірні табличні дані та матриці. Застосування вкладених циклів.

361. Елементи матиці A(7, 5) є дійсними числами, що обчислюються за правилом $A(i,j) = \frac{15*\sin(j)}{i+j}$, $i = 1..7, j = 1..5$. Знайти рядок з мінімальною сумою елементів та рядок з максимальною сумою елементів. Вивести утворену матрицю, максимальну та мінімальну суми та номери відповідних рядків.

362. У квадратній матриці $A(6,6)$, що заповнена за правилом $A(i,j) = \frac{3i-2j}{3*cos(j)}$, $i,j = 1..6$, знайти середнє арифметичне елементів, які знаходяться під головною діагоналлю та добуток діагональних елементів. Вивести на екран утворену матрицю та обчислені значення.

363. Масив цілих чисел M розміром 5×6 заповнено випадковими числами з відрізка $[-50;\ 50]$. Вивести на екран масив M у вигляді прямокутної таблиці. Сформувати одновимірний масив, кожний елемент якого дорівнює найбільшому за абсолютною величиною елементу відповідного стовпця M. Отриманий масив вивести на екран.

364. Обчислити добуток та кількість додатних елементів кожного стовпця матриці $A(6,6)$, заповненої за правилом $A(i,j) = 3j * cos(2i * j),\ i,j = 1..6$. Результати вивести у вигляді трьох рядків: в першому – номери стовпців матриці, в другому – відповідні добутки, в третьому – відповідні кількості. Під результатами обчислень вивести матрицю A.

365. Задано матрицю $M(7,8)$, заповнену випадковими числами з відрізка $[-7;\ 7]$. Вивести матрицю M на екран. Під матрицею окремо вивести усі її рядки, в яких кількість від'ємних елементів дорівнює кількості додатних.

366. Вивести на екран суму від'ємних елементів кожного рядка матриці $A(7,5)$, заповненої дійсними числами за правилом $A(i,j) = i * sin(i * 5j),\ i = 1..7,\ j = 1..5$. На екран вивести елементи матриці A, а в кінці кожного рядка після відповідного пояснення вказати обсчислену для нього суму.

367. Задано квадратну матрицю розміром 7×7, заповнену випадковими цілими числами з відрізка $[1;\ 9]$. Вивести матрицю на екран. Сформувати одновимірний масив, елементи якого дорівнюють значенням другого за порядком парного елемента рядка (за відсутності такого елемента відповідному елементу одновимірного масиву присвоїти значення 8).

368. Елементи матиці A(5, 8) – дійсні числа, що обчислюються за правилом $A(i,j) = \frac{i*\sin(j)}{3i+j}$, $i = 1..5, j = 1..8$. Сформувати одновимірний масив, кожний елемент якого дорівнює кількості елементів відповідного рядка, більших за задане число p. Вивести на екран матрицю та сформований масивом.

369. Задано матрицю розміру 9×5, заповнену випадковими цілими числами з відрізка [1; 20]. Вивести матрицю на екран. Під матрицею окремо вивести усі її рядки, в яких парних елементів більше, аніж непарних.

370. Масив цілих чисел розміром 7×5 заповнено випадковими числами з відрізка $[-8; 9]$. Вивести на екран заповнений масив. Сформувати одновимірний масив, кожний елемент якого дорівнює добутку парних додатних елементів відповідного рядка. Результат вивести на екран.

371. Задано квадратну матрицю P розміром 7×7, заповнену дійсними числами за правилом $P(i,j) = 4j * \sin(5i)$, $i,j = 1..7$. Сформувати одновимірний масив, елементи якого дорівнюють значенням другого за порядком додатного елемента стовпця (за відсутності такого елемента відповідному елементу одновимірного масиву присвоїти значення 0.77). Вивести матрицю та сформований масив на екран.

372. У квадратній матриці $A(8,8)$, що заповнена за правилом $A(i,j) = \frac{i+2j}{12cos(5i)}$, $i,j = 1..8$, знайти різницю середнього арифметичного елементів, які розташовані під головною діагоналлю та середнього арифметичного елементів, розташованих над нею. Вивести на екран утворену матрицю та обчислене значення.

373. Елементи матиці M(7, 6) є дійсними числами, що обчислюються за правилом $A(i,j) = \frac{17*\sin(3j)}{2i}$, $i = 1..7, j = 1..6$. Знайти номер стовпця з мінімальною сумою елементів та стовпця з максимальною сумою елементів. Вивести утворену матрицю, та усі елементи стовпця з максимальною та стовпця з мінімальною сумами елементів.

374. Задано матрицю $A(8,7)$, заповнену за правилом $A(i,j) = \frac{i+2j}{7sin(3j)}$, $i = 1..8$, $j = 1..7$. Вивести матрицю на екран. Під матрицею окремо вивести усі її рядки, в яких кількість від'ємних елементів більша за кількість невід'ємних.

375. Задано квадратну матрицю P розміром 7×7 заповнену дійсними числами за правилом $P(i,j) = \frac{10\sin(3i)}{i+j}$, $i,j = 1..7$. Сформувати одновимірний масив $Q(7)$, кожен елемент $Q(k)$ якого дорівнює різниці максимального елемента k-го рядка та мінімального елемента k-го стовпця матриці. Вивести матрицю та сформований масив на екран.

376. Масив $M(8,5)$ містить дані про кількість зібраних приладів кожним з 8-и складальників бригади протягом 5-ти робочих днів тижня. Скласти програму, яка за заданою вартістю складання одного приладу підрахує суму тижневої оплати праці кожного працівника. В програмі передбачити заповнення масиву даними, отриманими від користувача. Визначити максимальну кількість приладів, зібраних одним працівником за день.

377. Масив $M(10,3)$ містить дані про зараховану вагу, підняту кожним з 10-и спортсменів у 3-х залікових дисциплінах змагань з важкої атлетики. Скласти програму, яка визначить пісумкову вагу (суму) за усіма вправами для кожного важкоатлета та визначить переможця змагань. В програмі передбачити заповнення масиву даними, отриманими від користувача.

378. В деякому регіоні працюють 5 метеостанцій, які фіксують добову кількість опадів на квадратний метр площі. Підсумки замірів за тиждень формують у вигляді двовимірного масиву $R(5,7)$. Скласти програму для заповнення масиву даними, отриманими від користувача. Визначити середню кількість опадів у регіоні протягом тижня(поденно та сумарну).

379. Масив $M(4,9)$ містить дані про вартість кожного з 4-х видів робіт на 9-ти об'єктах компанії. Скласти програму для заповнення масиву даними, отриманими від користувача. Визначити загальну вартість виконання робіт для кожного з об'єктів та сумарні вартості кожного виду робіт на усіх об'єктах компанії.

380. Масив $A(10,4)$ містить дані про оцінки у 100-бальній системі групи з 10-ти студентів з кожного з 4-х іспитів. Скласти програму для заповнення масиву даними, отриманими від користувача. Визначити номер студента з найвищим рейтингом (сумарною оцінкою з усіх предметів).

381. Дані про оптову ціну 8-и видів продукції у кожного з 5-и оптових постачальників задається у вигляді двовимірного масиву $P(8,5)$. Скласти програму для визначення номера постачальника з найнижчою ціною на вказаний користувачем вид продукції (за номером). В програмі передбачити ввід користувачем даних для заповнення масиву та вивід середньої (за всіма постачальниками) ціни кожного виду продукції.

382. Масив $M(7,6)$ містить дані про кількість кілометрів дорожнього покриття, відремонтованого кожною з 6-и ремонтних бригад протягом 7-и днів тижня. Скласти програму, яка за заданою вартістю ремонту одного кілометра дороги підрахує суми, які вкладали в ремонт доріг кожного дня окремо та загальну вартість робіт за тиждень. В програмі передбачити заповнення масиву даними, отриманими від користувача.

383. Масив $M(5,8)$ містить дані про кількість балів, виставлену кожним з 5-и суддів кожній з 8-и учасниць у обов'язковій програмі змагань з художньої гімнастики. Скласти програму, яка визначить рейтингову таблицю (середній бал) для усіх гімнасток та визначить переможицю змагань. В програмі передбачити заповнення масиву даними та вивід результатів.

384. Дані про вартість 6-и вузлів, з яких збирають прилад, від кожного з 5-и виробників задається у вигляді двовимірного масиву $P(6,5)$. Скласти програму для визначення мінімальної собівартості приладу та формування масиву номерів постачальників, в яких слід замовити кожен з вузлів для її досягнення. В програмі передбачити ввід користувачем даних для заповнення масиву.

385. В масив $P(8,5)$ записують ціни на 8 різних товарів у 5-ти інтернет магазинах. Скласти програму для заповнення масиву даними, отриманими від користувача. Вивести для кожного товару номер магазину, в якому він коштує найдешевше.

386. Дані розподіленої системи датацентру доступні для користувачів мережі на 6-ти окремих серверах. Добова кількість запитів фіксується окремо для кожного сервера, а підсумки за тиждень формуються у вигляді двовимірного масиву $R(6,7)$. Скласти програму для визначення дня тижня з максимальним та дня тижня з мінімальним навантаженням на сервери датацентру. В програмі передбачити ввід користувачем даних для заповнення масиву.

387. Скласти програму для визначення загальної вартості замовлення вказаних користувачем кількостей кожного з 7-и видів продукції у кожного з 6-и постачальників. Дані про ціну кожного виду продукції у кожного з постачальників оформити у вигляді двовимірного масиву $P(6,7)$. В програмі передбачити ввід користувачем даних для заповнення масиву, а також вивід номера постачальника, у якого вказане замовлення коштуватиме найдешевше.

388. Масив $M(5,6)$ містить дані про кількість кубометрів деревини, заготованої кожною з 6-и бригад лісорубів протягом 5-и робочих днів тижня. Скласти програму, яка за заданою вартістю заготівлі одного кубометра деревини підрахує суму оплати праці кожної з бригад за тиждень. В програмі передбачити заповнення масиву отриманими від користувача даними. Визначити номер найпродуктивнішої бригади.

389. Масив $M(3,10)$ містить дані про висоту, взяту кожним з 10-и учасників змагань у стрибах з жердиною у кожній із 3-х спроб. Скласти програму, яка визначить рейтингову таблицю (найкращий результат) для усіх стрибунів та визначить переможця змагань. В програмі передбачити заповнення масиву даними та вивід результатів.

390. Масив $M(6,6)$ містить дані про курс продажу деякої валюти у 6-ти різних комерційних банках протягом 6-ти робочих днів тижня. Скласти програму, яка формує «графік» (масив зі значеннями) середнього курсу продажу валюти протягом тижня, а також за заданим номером дня визначає номер банку, в якому курс найнижчий. В програмі передбачити заповнення масиву отриманими від користувача даними.

391. Заповнити матрицю $A(5,6)$ випадковими числами з відрізка $[-10;\ 10]$. В кожному рядку матриці поміняти місцями останній та мінімальний елементи відповідного рядка. Вивести на екран вихідну та перетворену матрицю.

392. Дано масив розмірності 5×5, елементи якого натуральні числа. Вивести на дисплей вихідний масив. Дзеркально відобразити його елементи відносно бічної діагоналі. Вивести початковий масив та результат його перетворення на дисплей.

393. У заданій квадратній матриці $M(7,7)$ замінити середній рядок масивом, утвореним з мінімальних елементів кожного стовпця матриці. Вивести задану та перетворену матрицю на екран.

394. Задано квадратну матрицю $Q(6,6)$, заповнену випадковими числами з відрізка $[-5;\ 5]$. Замінити усі елементи матриці, розташовані під головною діагоналлю на максимальний елемент відповідного їм рядка. Вивести на екран вихідну та змінену матрицю.

395. Дано масив розмірності 7×6, елементи якого натуральні числа. Вивести на дисплей вихідний масив. Переставити рядки так, щоб в першому стовпці елементи були впорядковані за спаданням. Результат вивести на дисплей.

396. Задано матрицю $P(6,7)$, заповнену випадковими числами з відрізка $[-9;\ 18]$. Поміняти місцями перший стовпець матриці, зі стовпцем, що містить максимальний її елемент. Вивести на екран вихідну та змінену матрицю.

397. Заповнити матрицю $M(4,8)$ випадковими числами з відрізка $[0;\ 20]$. В кожному стовпці матриці поміняти місцями перший та максимальний елементи відповідного рядка. Вивести на екран матрицю M та утворену з неї, після перестановки вказаних елементів, матрицю.

398. Дано масив розмірності 6×7, елементи якого натуральні числа. Вивести на дисплей вихідний масив. Переставити рядки так, щоб в останньому стовпці елементи були впорядковані за зростанням. Результат вивести на дисплей.

399. У заданій квадратній матриці $B(7,7)$ замінити середній стовпець масивом, утвореним з максимальних елементів кожного рядка матриці. Вивести задану та перетворену матрицю на екран.

400. Задано матрицю $M(7,5)$, заповнену випадковими числами з відрізка $[-7;\ 7]$. Поміняти місцями перший рядок матриці з рядком, що містить максимальний її елемент. Вивести на екран вихідну та змінену матрицю.

401. Дано масив розмірності 6×6, елементи якого натуральні числа. Вивести на дисплей вихідний масив. Повернути його на 90^0 за годинниковою стрілкою. Результат вивести на дисплей.

402. Заповнити матрицю $A(6,6)$ випадковими числами з відрізка $[-9;\ 9]$. В кожному рядку матриці поміняти місцями перший та максимальний елементи відповідного рядка. Вивести на екран матрицю A та утворену з неї, після перестановки вказаних елементів, матрицю.

403. Задано квадратну матрицю $Q(7,7)$, заповнену випадковими числами з відрізка $[-7;\ 7]$. Замінити усі елементи матриці, розташовані над головною діагоналлю на максимальний елемент відповідного їм стовпця. Вивести на екран вихідну та змінену матрицю.

404. Задану квадратну матрицю $P(5,5)$ перетворити за правилом: рядок з номером n зробити стовпчиком з номером n і навпаки, де n – задане число, введене користувачем з клавіатури. Вивести задану та перетворену матрицю на екран.

405. У заданій квадратній матриці $M(6,6)$ замінити головну діагональ масивом, утвореним з максимальних елементів кожного рядка матриці. Вивести задану та перетворену матрицю на екран.

406. Скласти програму для заповнення квадратної матриці розміру 6×6 цілими числами так, як показано на малюнку. Заповнену матрицю відобразити симетрично відносно головної діагоналі. Вивести на екран утворену матрицю, виконати її перетворення і відобразити на екрані його результат.

0	1	2	3	4	5
2	3	4	5	6	7
4	5	6	7	8	9
6	7	8	9	10	11
8	9	10	11	12	13
10	11	12	13	14	15

407. Розробити програму для утворення квадратної матриці $M(7,7)$ та заповнення її цілими числами за правилом, що проілюстроване поданим тут малюнком. Заповнену матрицю повернути на 90° за годинниковою стрілкою. Вивести на екран утворену матрицю, виконати вказане перетворення і відобразити на екрані його результат.

1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1

408. Запрограмувати утворення квадратної матриці розміру 6×6 та заповнення її цілими числами так, як показано на малюнку. Заповнену матрицю відобразити симетрично відносно бічної діагоналі. Вивести на екран утворену матрицю, виконати її перетворення і відобразити на екрані його результат.

```
 1  1  2  2  3  3
 4  4  5  5  6  6
 7  7  8  8  9  9
10 10 11 11 12 12
13 13 14 14 15 15
16 16 17 17 18 18
```

409. Створити програму для побудови квадратної матриці $A(7,7)$ та заповнення її цілими числами за правилом, яке ілюструє поданий тут малюнок. Заповнену матрицю відобразити симетрично відносно середнього рядка. Вивести на екран утворену матрицю та результат її перетворення.

```
1 2 3 4 5 6 7
0 1 2 3 4 5 6
0 0 1 2 3 4 5
0 0 0 1 2 3 4
0 0 0 0 1 2 3
0 0 0 0 0 1 2
0 0 0 0 0 0 1
```

410. Скласти програму для заповнення квадратної матриці розміру 6×6 цілими числами так, як показано на малюнку. Заповнену матрицю повернути на 90° за годинниковою стрілкою. Вивести на екран утворену матрицю, виконати її перетворення і відобразити на екрані його результат.

```
0 1 0 3  0  5
2 0 4 0  6  0
0 3 0 5  0  7
4 0 6 0  8  0
0 5 0 7  0  9
6 0 8 0 10  0
```

411. Створити програму для побудови квадратної матриці $A(7,7)$ та заповнення її цілими числами за правилом, яке ілюструє поданий тут малюнок. Заповнену матрицю повернути на 90° проти годинникової стрілки. Вивести на екран утворену матрицю та результат її перетворення.

```
1 0 0 0 0 0 7
0 2 0 0 0 6 0
0 0 3 0 5 0 0
0 0 0 4 0 0 0
0 0 3 0 5 0 0
0 2 0 0 0 6 0
1 0 0 0 0 0 7
```

412. Запрограмувати утворення квадратної матриці розміру 6×6 та заповнення її цілими числами так, як показано на малюнку. Заповнену матрицю повернути на 90° проти годинникової стрілки. Вивести на екран утворену матрицю, виконати її перетворення і відобразити на екрані його результат.

```
0 1 2 3 4  5
1 2 3 4 5  6
2 3 4 5 6  7
3 4 5 6 7  8
4 5 6 7 8  9
5 6 7 8 9 10
```

413. Створити програму для побудови квадратної матриці $B(7,7)$ та заповнення її цілими числами за правилом, яке ілюструє поданий тут малюнок. Заповнену матрицю відобразити симетрично відносно середнього стовпця. Вивести на екран утворену матрицю та результат її перетворення.

```
1 2 3 4 5 6 7
7 1 2 3 4 5 6
6 7 1 2 3 4 5
5 6 7 1 2 3 4
4 5 6 7 1 2 3
3 4 5 6 7 1 2
2 3 4 5 6 7 1
```

414. Скласти програму для заповнення квадратної матриці розміру 6×6 цілими числами так, як показано на малюнку. Транспонувати утворену матрицю та вивести на екран спочатку матрицю у первинному вигляді, а потім у вигляді, якого вона набуде внаслідок транспонування.

```
 0  6 12 18 24 30
 1  7 13 19 25 31
 2  8 14 20 26 32
 3  9 15 21 27 33
 4 10 16 22 28 34
 5 11 17 23 29 35
```

415. Створити програму для побудови квадратної матриці $C(7,7)$ та заповнення її цілими числами за правилом, яке ілюструє поданий тут малюнок. Заповнену матрицю відобразити симетрично відносно бічної діагоналі. Вивести на екран утворену матрицю та результат її перетворення.

```
1  1  1  1   1   1   1
1  2  3  4   5   6   7
1  3  6  10  15  21  28
1  4  10 20  35  56  84
1  5  15 35  70  126 210
1  6  21 56  126 252 462
1  7  28 84  210 462 924
```

416. Запрограмувати утворення квадратної матриці розміру 6×6 та заповнення її цілими числами так, як показано на малюнку. Заповнену матрицю відобразити симетрично відносно бічної діагоналі. Вивести на екран утворену матрицю, виконати її перетворення і відобразити на екрані його результат.

```
0 1 2 3 4 5
6 5 4 3 2 1
0 1 2 3 4 5
6 5 4 3 2 1
0 1 2 3 4 5
6 5 4 3 2 1
```

417. Розробити програму для побудови квадратної матриці $M(7,7)$ та заповнення її цілими числами за правилом, яке показане на малюнку. Заповнену матрицю відобразити симетрично відносно середнього рядка. Вивести на екран утворену матрицю та результат її перетворення.

```
1 2 3 4 5 6 7
2 3 4 5 6 7 0
3 4 5 6 7 0 0
4 5 6 7 0 0 0
5 6 7 0 0 0 0
6 7 0 0 0 0 0
7 0 0 0 0 0 0
```

418. Скласти програму для заповнення квадратної матриці розміру 6×6 цілими числами так, як показано на малюнку. Заповнену матрицю повернути на 90° за годинниковою стрілкою. Вивести на екран утворену матрицю, виконати її перетворення і відобразити на екрані його результат.

```
0 1 0 1 0 1
1 2 1 2 1 2
0 1 0 1 0 1
1 2 1 2 1 2
0 1 0 1 0 1
1 2 1 2 1 2
```

419. Створити програму для побудови квадратної матриці $M(7,7)$ та заповнення її цілими числами за правилом так, як показано на малюнку. Заповнену матрицю відобразити симетрично відносно бічної діагоналі. Вивести на екран утворену матрицю та результат її перетворення.

```
1 1 1 1 1 1 1
1 2 3 4 5 6 7
1 3 6 0 5 1 8
1 4 0 0 5 6 4
1 5 5 5 0 6 0
1 6 1 6 6 2 2
1 7 8 4 0 2 4
```

420. Запрограмувати утворення квадратної матриці розміру 6×6 та заповнення її цілими числами так, як показано на малюнку. Заповнену матрицю повернути на 90° проти годинникової стрілки. Вивести на екран утворену матрицю, виконати її перетворення і відобразити на екрані його результат.

```
5 4 3 2 1 0
1 2 3 4 5 6
5 4 3 2 1 0
1 2 3 4 5 6
5 4 3 2 1 0
1 2 3 4 5 6
```

Практикум до § 5. Опрацювання двовимірних масивів

Під двовимірним масивом розуміють сукупність елементів одного типу, яка індексується за двома напрямами двома цілими числами та має спільну назву. Така сукупність даних має зручне візуальне подання у вигляді прямокутної таблиці, що відобразилося у вживанні термінів «рядки» чи «стовпці» до елементів з однаковим значенням одного з індексів (див. малюнок).

Стовпці 1…k

Рядки 1…n				
	елемент(1,1)	елемент(1,2)	…	елемент$(1,k)$
	елемент(2,1)	елемент(2,2)	…	елемент$(2,k)$
	…	…	…	…
	елемент$(n,1)$	елемент$(n,2)$	…	елемент(n,k)

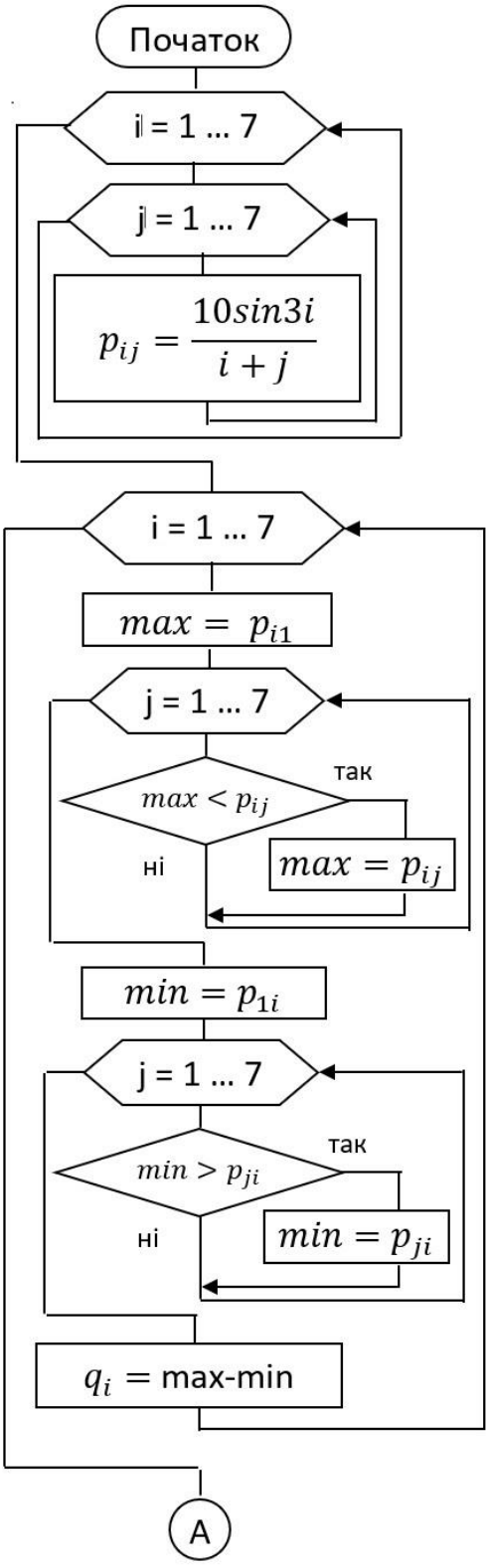
Доступ до конкретного елемента двовимірного масиву здійснюється вказанням двох чисел, – номера «рядка», та номера «стовпця». За значенням лише одного з індексів тут можна встановити лише сукупність елементів, яка сама по собі також є масивом, але вже одновимірним.

З точки зору збереження даних пам’яті комп’ютера така структура є дещо «штучною», бо адресація комірок пам’яті послідовною. Тому з двовимірним масивом працють послідовно як із сукупністю одновимірних масивів, а в літературі з програмування часто можна зустріти визначення двовимірного масиву як масиву масивів або списку масивів.

Необхідність зберігання та програмного опрацювання двовимірних масивів обумовлюється, насамперед, їх прикладним значенням. В різних задачах фізики та техніки, економіки, нейромережевих технологіях систем штучного інтелекту і т. п. використовується математитичний апарат матриць, які за своєю конструкцією і є двовимірними числовими масивами.

В задачах цього розділу, як і загалом в літературі з програмування, подекуди терміни матриця та двовимірний масив ототожнюються. Насправді, поняття масиву є ширшим, оскільки позначає сукупність елементів довільного типу, а матриця утворюється лише з чисел.

375. Задано квадратну матрицю P розміром 7×7 заповнену дійсними числами за правилом $P(i,j) = \frac{10\sin(3i)}{i+j}$, $i,j = 1..7$. Сформувати одновимірний масив $Q(7)$, кожен елемент $Q(k)$ якого дорівнює різниці максимального елемента k-го рядка та мінімального елемента k-го стовпця матриці. Вивести матрицю та сформований масив на екран.

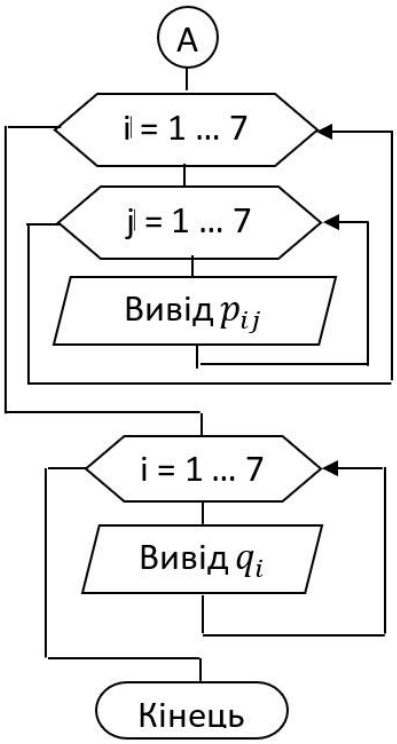


Умова задачі, на перший погляд, дещо порушує одну з важливих властивостей алгоритмів – ***масовість***, яка вимагає, щоб алгоритм був застосовний до розв'язування задачі певного типу при довільних значеннях вхідних даних. Тут дані, за умовою задачі, заповнюються за заданим правилом і будуть одними і тими ж при кожному запуску прогами, а, отже, результат її виконання кожного разу буде однаковим. Насправді ж ***масовість*** розглянутого далі алгоритму полягає в тому, що правильний результат його виконання не повинен залежити від правила заповнення вхідного масиву даними.

Алгоритми опрацювання двовимірних масивів, зазвичай, утворюються з декількох пар вкладених циклів. Якщо трактувати двовимірний масив як прямокутну таблицю, утворену з рядків та стовпців, то завдання зовнішнього циклу поступово один за одним переглядути кожен рядок, в якому внутрішній цикл повинен переглянути послідовно кожен елемент та виконати з ним потрібну дію. Така конструкція використана на початку нашого алгориму для обчислення елементів матриці, а пізніше, для виводу їх на екран.

Конструкція утворення шуканого вектора дещо складніша. Тут зовнішній цикл перераховує номери рядків для першого внутрішнього, які одночасно є номерами стовпців для другого.

У підсумку послідовність дій алгоритму виглядатиме так:



- *спочатку програма для кожного рядка, для кожного елемента в поточному рядку, обчислює значення за заданою формулою;*
- *організовується цикл, що перераховує від 1 до 7 номери і рядка/стовця для внутрішніх циклів;*
- *для і-го рядка перший внутрішній цикл шукає максимальний елемент, починаючи з першого елемента, поступово порівнючи з усіма натупними елементами цього рядка;*
- *другий внутрішній цикл для і-го стовпця шукає мінімальний елемент, починаючи з першого елемента та поступово порівнюючи з усіма натупними;*
- *від знайденого максимуму і-го рядка віднімаємо знайдений мінімум і-го стовпця, отриману величину записуємо в і-й елемент шуканого вектора;*
- *парою вкладених циклів виводимо по рядках елементи початкової матриці P;*
- *у підсумку за допомогою циклу виводимо усі елементи отриманого у результаті обчислень вектра.*

Програма на Pascal.

Перекладаючи сформульований алгоритм на ***Pascal*** врахуємо, що двовимірні масиви на цій мові оголошуються за допомогою інструкції вигляду:

```
<змінна>: array[1..<maxRow>,1..<maxCol>] of <тип елемента>;
```

де розміри ***maxRow*** та ***maxCol*** вказують кількість рядків та стовпців масиву, відповідно. Таке оголошення записують в розділі опису змінних ***var***, а також в розділі ***const***, можна також створити новий тип на основі масиву в розділі ***type***.

Основні конструкції поданої далі ***Pascal***-програми уважному читачеві мають бути зрозумілими на основі матеріалу попередніх розділів. Звернемо увагу лише на те що в конструкції першої пари вкладених циклів не використовується складений оператор begin…end, тілом внутрішнього циклу тут є оператор присвоєння, а тілом зовнішнього, – сам внутрішній цикл.

```
program Task375;
var i, j: integer; max, min : real;
p: array[1..7,1..7] of real; q: array[1..7] of real;
begin
{Заповнення матриці P}
    for i:=1 to 7 do
        for j:=1 to 7 do
            p[i,j] := 10*sin(3*i)/(i+j);
{Обчислення елементів масиву Q}
    for i:=1 to 7 do
    begin
        max := p[i, 1];
        for j:=1 to 7 do
            if p[i, j] > max then max:= p[i, j];
        min := p[1, i];
        for j:=1 to 7 do
            if p[j, i] < min then min := p[j, i];
        q[i] := max - min;
    end;
    writeln('Матриця P:');
    for i:=1 to 7 do {Вивід  матриці P}
    begin
        for j:=1 to 7 do
            write(p[i,j]:10:4);
        writeln();{перехід на наступну стрічку}
    end;
{Вивід  масиву Q}
    writeln('Масив Q:');
    for i:=1 to 7 do write(q[i]:10:3);
end.
```

Програма на C++.

В ***C++***, як і у інших нащадків ***C***, у реалізації «багатовимірності» масивів проглядається концепція масиву масивів. Причому, не тільки в способі оголошення, який у випадку двовимірного масиву виглядає так:

```
<тип> <назва змінної>[<к-сть рядків>][<к-сть стовпців>];
```

Проявляється цей підхід також у способі розміщення елементів масиву, – елементи двовимірного масиву в пам'яті комп'ютера записуються послідовно, – рядок за рядком. Звертання до окремого елемента масиву здійснюється вказанням двох індексів, – номера масиву (рядка) в перших квадратних дужках та номера елемента окремому масиві (рядку) в других. Варта також не забувати про індексацію елементів масивів, яка в ***C++*** починається з 0. Решта особливостей розв'язування задачі 375 мовою ***C++***, – в коді далі:

```
#include <iostream>

#include <math.h>

using namespace std;


int main()

{ //Оголошення та створення двовимірного

    double p[7][7];// масиву дійсних чисел

    // Заповнення матриці p. Використовуємо пару

    // вкладених циклів за індексами елементів,

    // індекси елементів пробігають значення 0..6

    for (int i = 0; i < 7; i++)

    {

        for (int j = 0; j < 7; j++)

        {//у виразі враховуємо, що індекси i, j

            //набувають значень 0..6 а не 1..7

            p[i][j] = 10 * sin(3 * (i+1) )

                / (i + 1 + j + 1);

        }

    }

    //Обчислення елементів масиву Q

    double q[7];

    for(int i = 0; i < 7; i++)
```

```
    {
        double max = p[i][0];
        for (int j = 0; j < 7; j++)
        {
            if (p[i][j] > max) max = p[i][j];
        }
        double min = p[0][i];
        for (int j = 0; j < 7; j++)
        {
            if (p[j][i] < min) min = p[j][i];
        }
        q[i] = max - min;
    }
    // Вивід  матриці P
    cout<<"Матриця P:"<<endl;
    for (int i = 0; i < 7; i++)
    {
        for (int j = 0; j < 7; j++)
            printf("%10.3f", p[i][j]);
        cout << endl;
    }
    cout<<"Масив Q:";
    for (int i = 0; i < 7; i++)   //Вивід  масиву Q
    {
        printf("%10.3f", q[i]);
    }
}
```

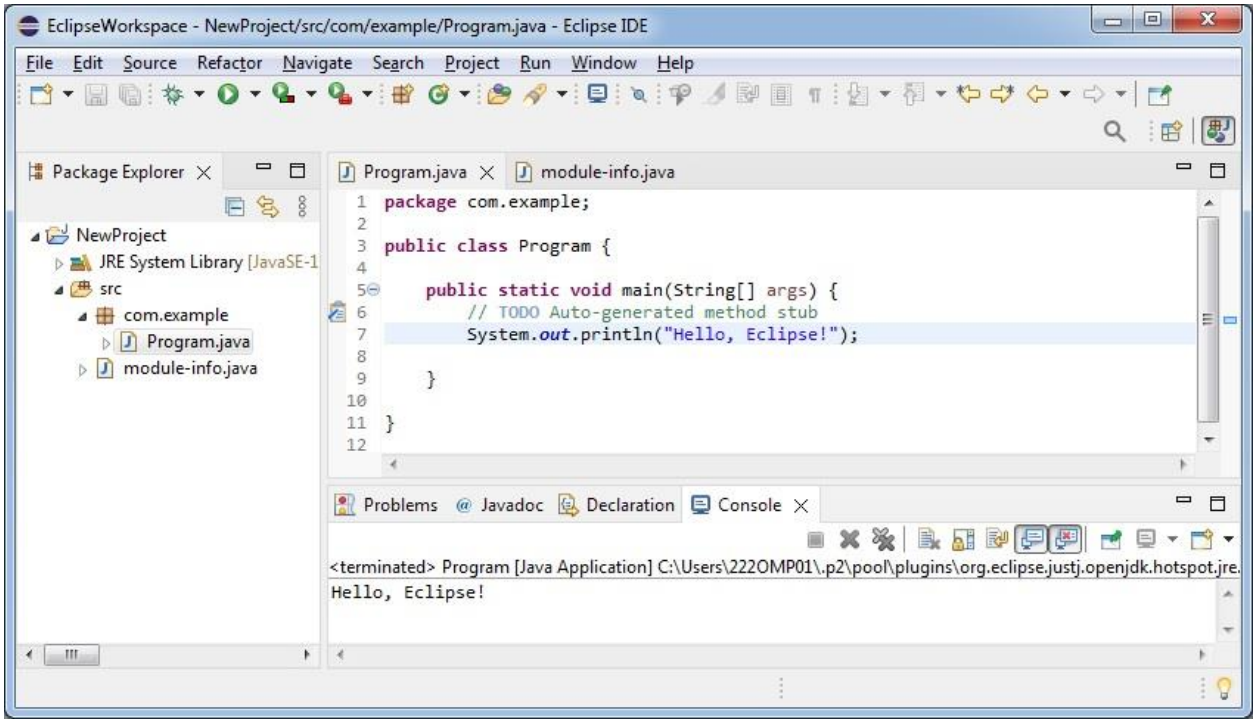
В рамках цього розділу приділимо увагу засобам розробки на Java. Уся історія розвитку цієї мови програмування відбувалася в руслі вільного програмного забезпечення (***Open source***). Велика кількість розробників та команд, софтверних компаній приносить в ***Java*** власні розробки бібліотек, інструментів та допоміжного програмного забезпечення власної розробки. Наприклад, компанія ***Oracle***, що є правонаступницею засновниці мови Sun Microsystems, випускає інтегроване середовище ***Oracle JDeveloper***, хоча найбільшою популярністю серед кількох десятків програм цього сегмента користуються безкоштовні IDE ***Eclipse***, ***Apache Netbeans*** та ***Intellij Idea***.

Усі середовища розробки використовують спеціальний пакет – ***Java Development Kit*** (***JDK***), що містить компілятор мови Java, бібліотеки стандартних класів та віртуальну машину ***JVM*** (***Java Virtual Machine***), на якій виконується програмний код. Тобто, щоб підготувати власний ПК до розробки на ***Java***, спочатку слід встановити пакет ***JDK***. Його можна завантажити з офіційного сайту Oracle за алресою https://www.oracle.com/java/technologies/downloads/, попередньо зареєструвавши власний профіль в системі. Популярністю серед java-розробників користується також і альтернативний пакет – ***OpenJDK***, актуальну збірку якого ***AdoptOpenJDK*** можна завантажити за адресою https://adoptopenjdk.net/. Встановлений JDK вже дозволяє розробляти Java-програми, компілювати та запускати їх в режимі командної стрічки. Для набору та редагування тексту програми при цьому можна скористатися будь-яким доступним текстовим редактором (Блокнот, Notepad++, GEdit і т.п.), проте, набагато зручніше це робити, встановивши відповідне інтегроване середовище розробки.

Вільне модульне інтегроване середовище розробки програмного забезпечення ***Eclipse*** підтримується некомерційною організацією ***Eclipse Foundation***. Складається з самої платформа ***Eclipse***, та набору інструментів для програмістів на ***Java***, систем контролю версій, конструкторів графічного інтерфейсу, тощо. Може бути використане для розробки застосунків на інших мовах програмування, зокрема, ***Ada***, ***C***, ***C++***, ***C#***, ***Fortran***, ***Perl***, ***PHP***, ***Python***, ***Ruby***, ***Scala*** завдяки встановленню різних додаткових плагінів, що розробляються паралельно з платформою.

Проєкт розпочато в компанії ***IBM***, він був розрахований на розробників ***Java*** і складався з ***Java Development Tools*** (***JDT***). Користувачі інструменту могли розширювати його можливості, встановлюючи готові плагіни, такі як інструменти розробки під інші мови програмування, та мали можливість створювати та використовувати власні плагіни і модулі. У 2004 році для супроводу та вдосконалення проєкту було створено ***Eclipse Foundation***, під

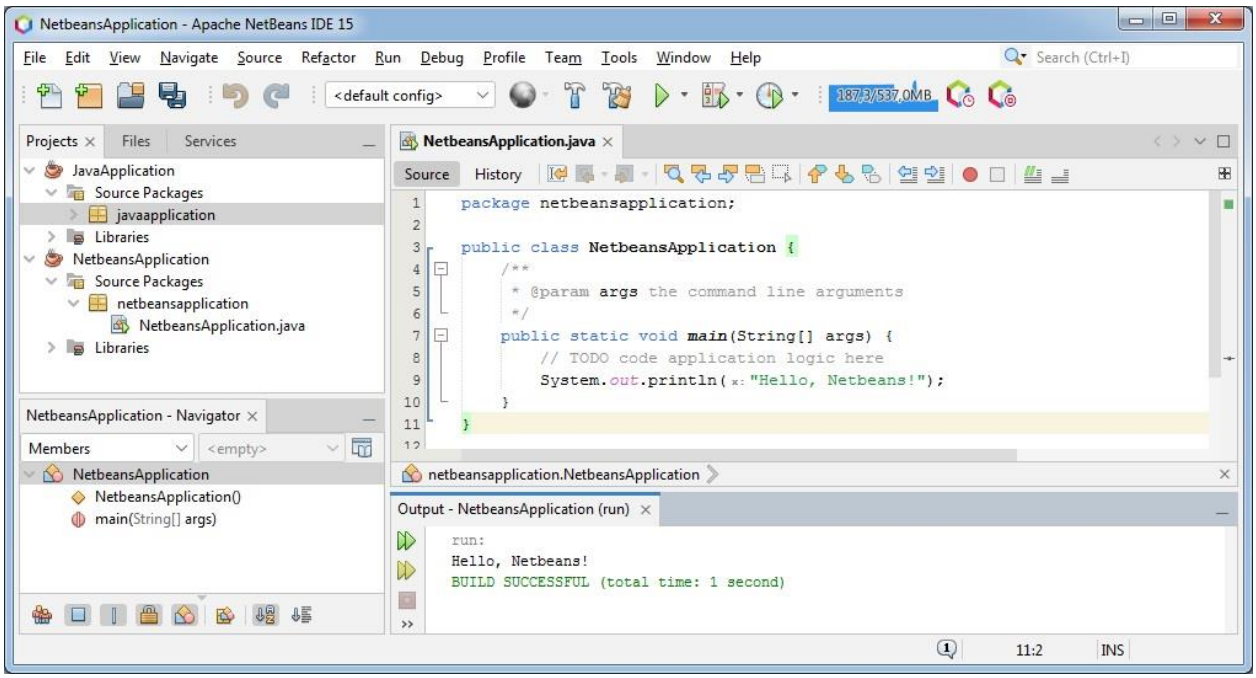
егідою якої сьогодні виходять нові версії ***IDE Eclipse*** та багато суміжних бібліотек та інструментів для java-розробки.



З офіційного сайту фундації https://www.eclipse.org/downloads/ можна безкоштовно завантажити актуальну версію ***Eclipse*** встановлення на власний ПК. ***Eclipse*** більш популярний на Заході, за версією західних інтернет-видань він вже багато років поспіль посідає перше місце в рейтингу популярності Java-IDE.

Популярне середовище для java-розробки ***NetBeans*** розпочалосвою історію в стінах Карлова Університету в Празі, як розробка студентів, згодом, було передане компанії ***Sun Microsystems***, яка перевела його в площину ***Open Source***, відкривши його програмний код для розробників. Тривалий час ***IDE Netbeans*** розвивалася як спільна розробка різних колективів та іт-компаній і пропонувалася в якості середовища розробки на сторінці завантаження пакету ***JDK***. За цей час програму укомплектували найрізноманітнішими шаблонами java-проєктів, інструментами для роботи з базами даних та серверами, а також інструментами розробки на інших мовах програмування, зокрема, ***C++*** та ***PHP***.

Починаючи з версії 9 (2016 р.) програмний продукт було передано в в проєкт ***Apache Incubator*** компанії ***Apache***. Його почали випускати під назвою ***Apache NetBeans*** і спочатку позбавили багатьох функціональних можливостей, оновлюючи і додаючи їх в наступних версіях. У релізі ***Apache NetBeans 15*** оновлено майже увесь функціонал, що був доступний в ***NetBeans*** 8.x та додано низку нових можливостей.



Завантажити актуальну версію ***Apache NetBeans*** для встановлення на свій ПК можна з офіційного сайту ***Apache*** за адресою:

https://netbeans.apache.org/download/index.html.

Чеська компанія ***JetBrains*** з 2001 постачає на ринок найпопулярнішу на наших теренах ***IDE*** для ***Java - Intellij Idea***. Повна версія програми (***Ultimate***) є платною, проте компанія пропонує безкоштовну версію продукту ***Intellij Idea Community Edition***, а також надає безкоштовні ліцензії на повну версію (для ознайомлення - на 30 днів та на рік для студентів та викладачів навчальних закладів). В компанії розробляютьтакож і популярне IDE для Python – ***PyCharm***.

Завантажити обидві версії продукту можна з офіційного сайту компанії за адресою https://www.jetbrains.com/idea/download. Більшість прикладів на java до задач цього збірника розроблені за допомогою ***Intellij Idea***.

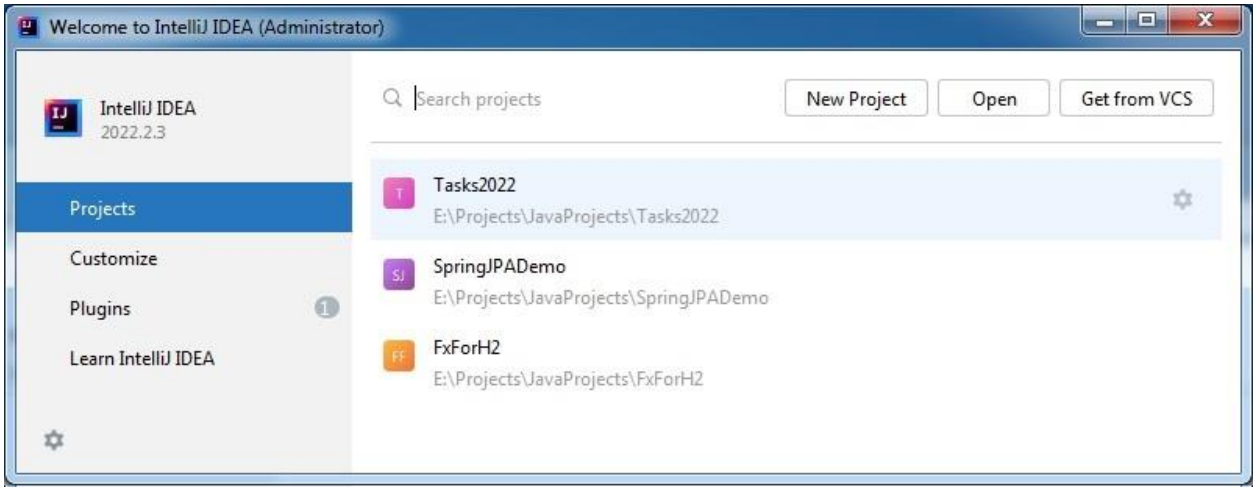
Зупинимося коротко на створенні та виконанні java-коду в цьому середовищі розробки. Щоб створити програму на Java за допомогою Intellij Idea спочатку потрібно:

- *завантажити та встановити пакет JDK;*
- *завантажити та встановити Intellij Idea;*
- *запустити Intellij Idea та створити Java-проект;*
- *до створеного проєкту додати новий Java-клас;*
- *в класі оголосити статичний main-метод.*

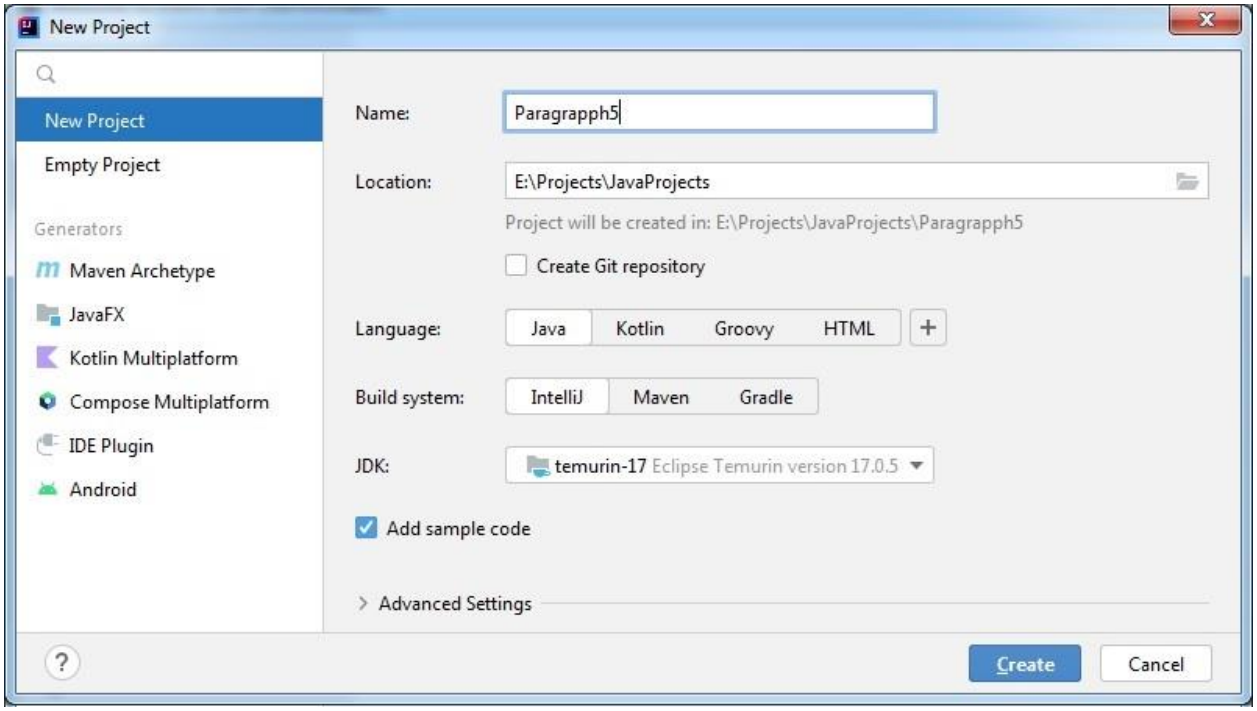
Далі можна переходити до написання коду програми.

Перший запуск щойно встановленої ***Intellij Idea*** буде починатися зі стартового вікна, подібного до показаного на малюку. Наступні запуски, зазвичай, починаються з проєкту, що був відкритий за попереднього сеансу

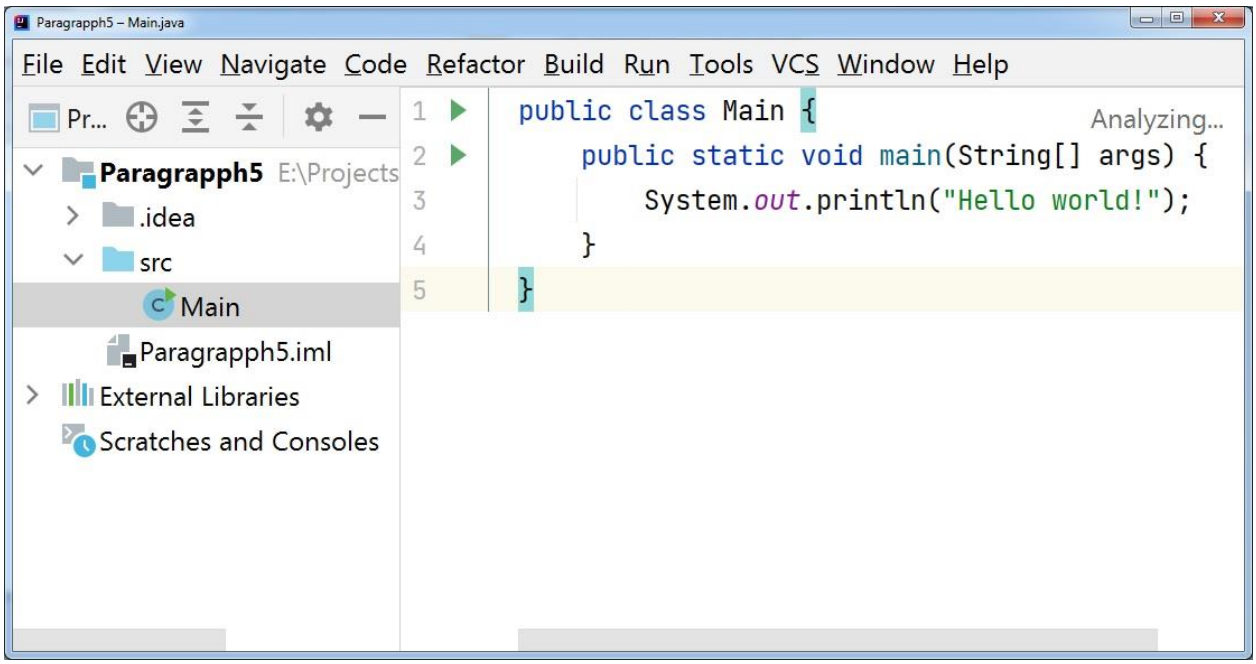
роботи IDE. Якщо закрити поточний проєкт, виконавши команду **File => Close Project**, – ми знову побачимо стартове вікно ***Intellij Idea***.



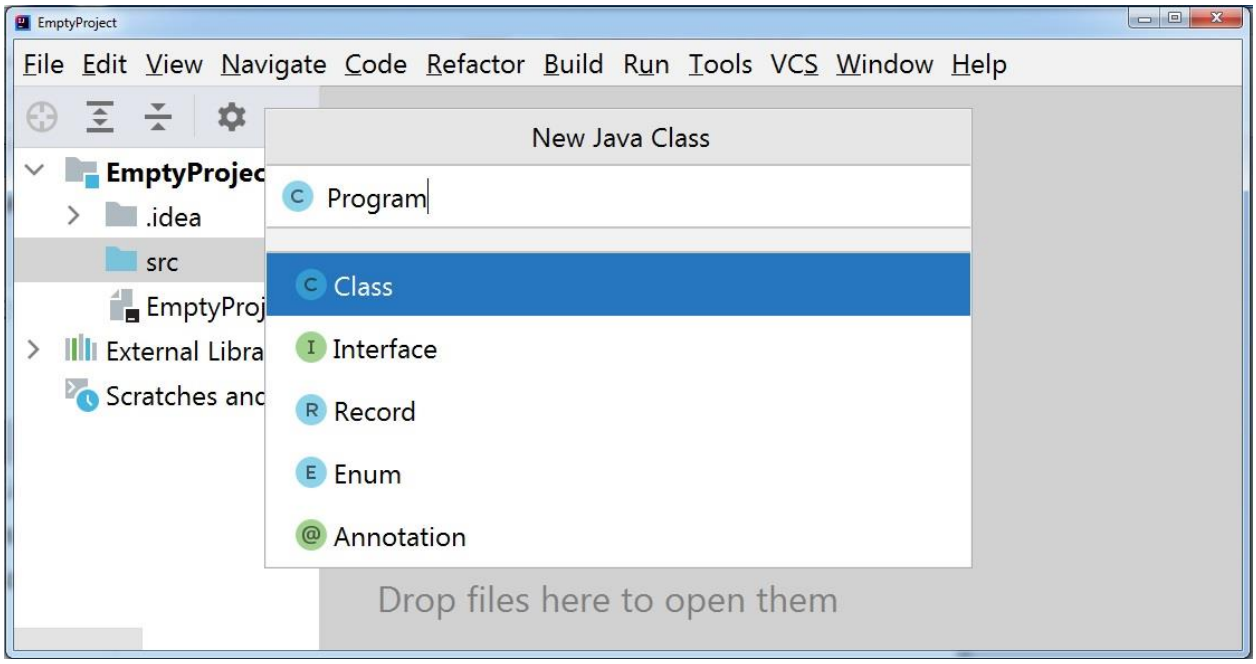
Розпочати створення нового проєкту можна кнопкою **New Project** на стартовому вікні ***Intellij Idea***, або **File => New => Project** з меню основного вікна програми.



На наступному кроці у вікні ***New Project*** слід обрати потрібні параметри проєкту, зокрема, папку для розташування проєкту та його назву. Для створення класичного консольного java-застосунку тип проєту у списку зліва слід залишити на пункті “New Project”. Так само варта залишити вибраними опції “Java” та “Intellij”. У розкривний список з підписом “JDK”, зазвичай, Intellij успішно знаходить і показує поточну версію ***Java Development Kit*** втановлену в системі, хоча інколи доводиться вказувати тут шлях до пакету ***JDK*** вручну. Ще одна опція “Add simple code” дозволяє при створенні проєкту отримати готовий клас ***`Main`*** з готовим методом ***`main`***, в якому слід писати код програми.



Якщо прапорець “Add simple code” не позначити, то ми отримаємо порожній проєкт, а клас для написання коду програми потрібно буде до нього додати. Це можна зробити з контекстного меню, викликаного на вузлі “src” у дереві проєкту зліва.



Далі можемо переходити до написання самого коду. Зазначимо лише, що найпростіший спосіб запустити створену програму – натиснути позначку у формі зеленого трикутника зліва від назви класу, чи таку ж позначку зліва від заголовку метода ***main***.

Програма на Java.

Концепція «масиву масивів» для створення багатовимірних масивів в ***Java***, проявляється ще більше, аніж в C++. Тут масиви є об’єктами в пам’яті віртуальної машини, і кожен рядок двовимірного масиву зберігається окремо,

він доступний за посиланням, як окремий об'єкт пам'яті. Синтаксично оголошення багатовимірного масиву вказується парою квадратних дужок на кожну розмірність, наприклад, тривимірний масив цілих чисел можна оголосити так:

```
int matrix3d[][][];
```

Для зручності роботи з багатовимірними масивами, в синтаксисі ***Java*** передбачено таке ж створення багатовимірного масиву оператором ***new*** (з вказанням в квадратних дужках розміру масиву по кожній розмірності). Як це використовується для двовимірного масиву пропонуємо розглянути у коді прикладу далі:

```
public class Task375 {
  public static void main(String[] args) {
    //Оголошення та створення
    //двовимірного масиву дійсних чисел
    double p[][] = new double [7][7];
    // Заповнення матриці p.
    // Використовуємо пару вкладених циклів
    // за індексами елементів
    // індекси елементів пробігають значення 0..6
    for (int i = 0; i < 7; i++){
      for (int j = 0; j < 7; j++) {
      //у виразі враховуємо, що індекси i, j
      //набувають значень 0..6 а не 1..7
        p[i][j] = 10 * Math.sin(3 * (i+1) )
                          / (i + 1 + j + 1);
      }
    }

    //Обчислення елементів масиву Q
    double q[] = new double[7];
    for(int i = 0; i < 7; i++){
      double max = p[i][0];
      for (int j = 0; j < 7; j++){
```

```
        if (p[i][j] > max) max = p[i][j];
      }
      double min = p[0][i];
      for (int j = 0; j < 7; j++){
        if (p[j][i] < min) min = p[j][i];
      }
      q[i] = max - min;
    }
    // Вивід  матриці P
    System.out.println("Матриця P:");
    for (int i = 0; i < 7; i++){
      for (int j = 0; j < 7; j++){
        System.out.printf("%10.3f", p[i][j]);
      }
      System.out.println();
    }
    //Вивід  масиву Q
    System.out.println("Масив Q:");
    for (int i = 0; i < 7; i++){
      System.out.printf("%10.3f", q[i]);
    }
  }
}
```

Програма на Python.

Використання в ***Python*** списків замість масивів в двовимірному випадку завдає ще більше клопоту. Якщо в ***Java*** чи ***C++*** «масив масивів» – лише концепція, за якою система працює з багатовимірними масивами, то ***список списків*** в ***Python***, – це реальний об'єкт, який потрібно створити в програмі для того, щоб зберігати та опрацьовувати двовимірний масив.

Система оголошення змінних в ***Python***, яка визначає тип змінної за присвоєним їй значенням, хоча і є динамічною, проте строго контролює тип об'єкта записаного в змінну та дозволяє виконувати зі значенням змінної лише дії, які передбачені її типом. Інколи програмісти намагаються оголосити двовимірний список відповідних розмірів інструкцією вигляду:

```
<variable> = [[<value>] * <colCount>] * <rowCount>
```

ініціалізуючи всі його елементи деяким значенням.

Слушність такого підходу частково оправдана тим, що значення ***value*** має певний тип, і цей тип буде задавати тип елементів списку. Хоча динамічна типізація не буде перешкоджати запису в елемент такого списку значень будь-якого іншого типу.

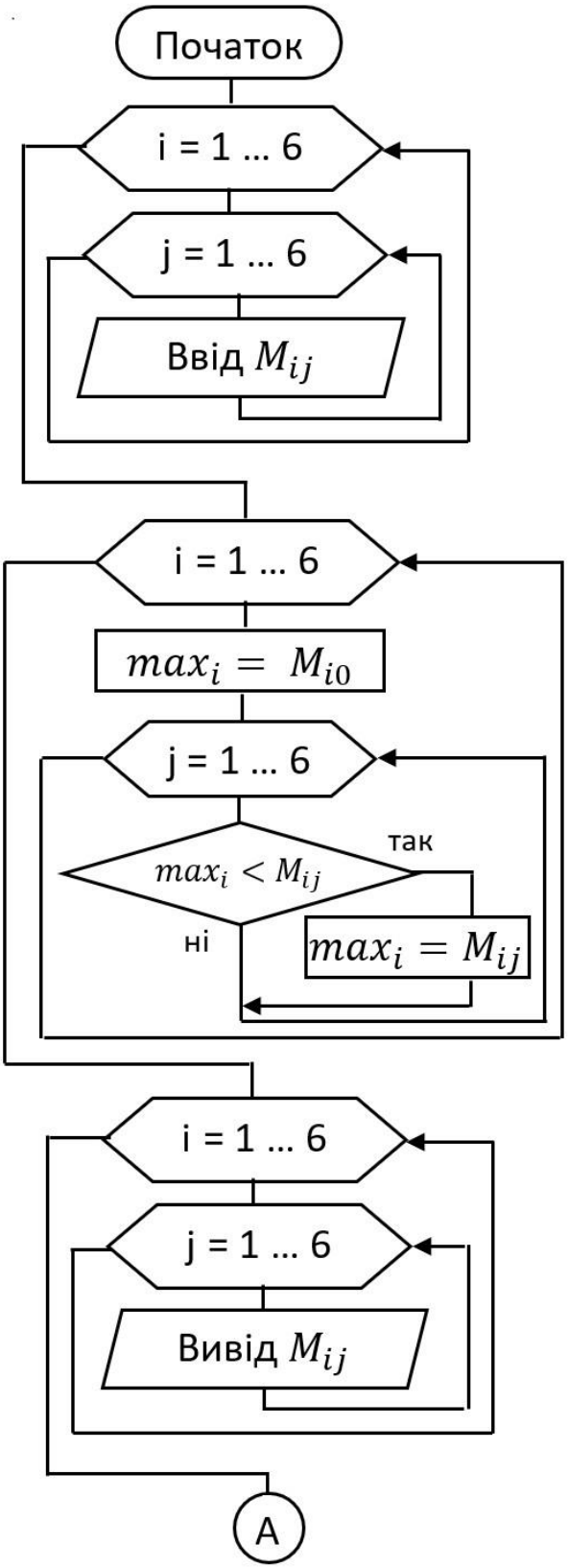
У своїй реалізації ми використали простіший підхід. Оскільки списки в ***Python*** можуть містити об'єкти будь-якого типу, то сама матриця ***p*** в коді нижче оголошена як звичайний (одновимірний) список. Далі в циклі для кожного рядка матриці створється окремий список, заповняється елементами та додається до списку ***p***. У підсумку звертання до елемента матриці за індексами ***p[i][j]*** насправді конвертується у вибір ***j-го*** елемента в ***i-му*** списку-рядку. Код програми для розв'язування задачі 375 на мові ***Python***:

```
import math

# Створюємо порожній список, в який

# записуватимемо рядки матриці

p = []

#зовнішній цикл по рядках від 0 до 6

for i in range(0,7):

    #створюємо новий одновимірний масив для кожного

    #рядка матриці і записуємо в нього елементи

    tmp = []

    for j in range(0,7):

        tmp.append(10*math.sin(3.0*(i+1))/(i + 1 + j + 1))

    p.append(tmp)
```

```
#Обчислення елементів масиву Q
q = []
for k in range(0,7):
    max = p[k][1]
    for j in range(0,7):
        if p[k][j] > max:
           max = p[k][j]
    min = p[1][k]
    for j in range(0,7):
        if p[j][k] < min:
            min = p[j][k]
    q.append( max - min )
print('Матриця P:')
# Для виводу елементів матриці скористаємося циклом
# типу foreach, який переглядає усі елементи списку
# зовнішній цикл записуватиме в змінну s
# кожен рядок матриці по черзі
for s in p:
    # внутрішій цикл проходитиме кожен елемент
    # поточного рядка-списку s
    for t in s:
        print('%10.3f'% t, end='')
    print()
#Вивід  масиву Q
print('Масив Q:')
for i in range(0, 7):
    print('%10.3f'% q[i], end='')
print()
```

405. У заданій квадратній матриці $M(6,6)$ замінити головну діагональ масивом, утвореним з максимальних елементів кожного рядка матриці. Вивести задану та перетворену матрицю на екран.

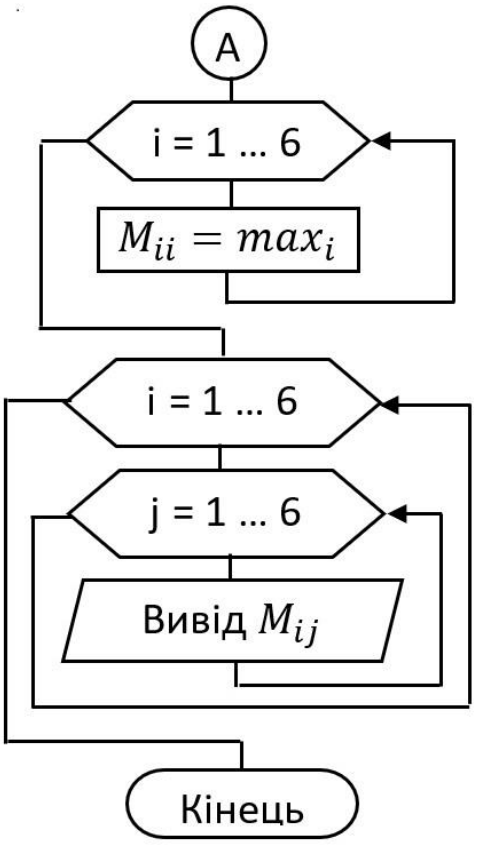
Задачі 391-420 відрізняються від попередньої групи задач тим, що передбачають зміну початкової матриці. Тобто, для демонстрації результату роботи програми слід відобразити на екрані матрицю в початковому стані, та вигляд цієї матриці після виконаних перетворень. Деколи в таких випадках намагаються використовувати дві окремі матриці, – одну для вхідних даних, другу для формування результату. Умовою задачі таке не передбачається, тому користуватимемося двома масивами.



За структурою алгоритм розв'язування задачі (див. блок-схему) мало чим відрізняється від алгоритму в задачі 375:

- *спочатку програма отримує від користувача 36 значень, по одному для кожного елемента кожного рядка,;*
- *організовується цикл, що перераховує номери рядків матриці від 1 до 6;*
- *для i-го рядка внутрішній цикл шукає максимальний елемент, починаючи з першого та поступово порівнюючи з усіма натупними елементами цього рядка, кожного разу обираючи більший з двох;*
- *знайдений максимум i-го рядка записується в i-й елемент допоміжного одновимірного масиву максимумів;*
- *парою вкладених циклів виводимо елементи початкової матриці P по рядках;*

- *за допомогою одного циклу замінюємо усі елементи матриці з однаковими індексами на відповідний елемент з масиву максимумів.*
- *парою вкладених циклів виводимо по рядках елементи матриці після виконаної заміни.*



Ввід усіх елементів двовимного масиву для опрацювання програмою, очевидно, займатиме в користувача багато часу. Якщо це відбувається в процесі експлуатації програми, то це цілком природньо, бо тут користувач вирішує поставлену задачу з допомогою створеної для цього програми.

Але цей процес неодмінно супроводжуватиме розробника програми у ході її налагодження, а це вже додаткові витрати часу. Тому в кодах реалізацій цього алгоритму ми будемо заповнювати (ініціалізувати) матрицю деякими вхідними даними, а під час виконання програми пропонуватимемо користувачеві або ввести елементи матриці, або виконати програму для заданих в програмі даних. Подібний підхід застосовується в ***модульному тестуванні***: для того, щоб перевірити працездатність написаного ним коду, програміст створює спеціальні замінники даних, так звані, ***mock-об'єкти***.

Програма на Pascal.

До сказаного про двовимірні масиви в ***Pascal*** у розв'язуванні задачі 375 залишається додати, що ініціалізація масиву тут здійснюється за допомогою круглих дужок та списку літералів, розділеного комами. Для двовимірних масивів кожен рядок записують також в дужках, наприклад:

```
m : array [1..3,1..3] of integer = ((1, 2, 3), (4, 5, 6),
                                    (7, 8, 9));
```

Ініціалізаця змінної повинна відбуватися при її оголошенні, тобто в розділі опису змінних або констант. Принципової різниці в якому розділі, ***var*** чи ***const,*** ініціалізовано масив немає, – в обидвох випадках значення його елементів будуть доступні для модифікації. Відмінність буде лише в тому, що розділ опису констант вимагає обов'язкової ініціалізації. Кількість елементів в списку ініціалізації повинна чітко відповідати розмірам масиву, – при неспівпадінні компілятор видаватиме повдомлення про помилку.

Код алгоритму розв'язання задачі 405 мовою ***Pascal:***

```
program Task405;
var i, j: integer; p:char;
    m: array[1..6,1..6] of integer =((2,3,4,5,6,7),
        (7,7,7,7,7,9),(1,2,3,4,5,6),
        (4,5,12,2,3,4),(1,1,1,14,5,3),(9,8,5,2,4,3));
        max: array[1..6] of integer;
begin
    writeln('Ввести дані для матриці? y/n');
    readln(p);
    if p = 'y' then
    begin
        {Ввід матриці}
        for i:=1 to 6 do
            for j:=1 to 6 do
                readln(m[i,j]);
    end;
{Формування вектора для заміни в матриці}
    for i:=1 to 6 do
    begin
        max[i] := m[i, 1];
        for j:=1 to 6 do
            if m[i, j] > max[i] then
                    max[i]:= m[i, j];
    end;
{Вивід  матриці M}
    writeln('Початкова матриця M:');
    for i:=1 to 6 do
    begin
        for j:=1 to 6 do
            write(m[i,j]:10);
```

```
        writeln();{перехід на наступну стрічку}
    end;
{Перетворення матриці M}
    for i:=1 to 6 do
        m[i, i] := max[i];
{Вивід  матриці M після перетворення}
    writeln('Перетворена матриця M:');
    for i:=1 to 6 do
    begin
        for j:=1 to 6 do
            write(m[i,j]:10);
        writeln();{перехід на наступну стрічку}
    end;
end.
```

Програма на C++.

В ***C***++ оголошення та ініціалізація змінних, (масивів, в тому числі) можуть виконуватися в будь-якому місці коду. Для ініціалізації масиву йому передають список значень відповідного типу, розділених комами, записаний у фігурних дужках. Наприклад, одновимірний масив з семи цілих чисел можна оголосити та ініціалізувати так:

```
int a[7] = { 1, -2, 3, -4, 5, -6, 7};
```

або так:

```
int s[] = { 1, -2, 3, -4, 5, -6, 7};
```

В другому оголошенні розмір масиву, тобто кількість його елементів, явним чином не вказується, – тут він визначається за кількістю даних у списку ініціалізації.

Перший спосіб «небезпечний» тим, що кількість елементів у списку може не відповідати вказаному розміру масиву. Компілятор видаватиме повідомлення про помилку і програми не буде компілювати такий код. Щоправда, лише за ситуації, коли елементів у списку забагато, якщо ж елементів у списку замало, то компілятор доповнить його нульовими значеннями до кількості, вказаної в оголошенні масиву. Наприклад, наступна інструкція

```
int p[5] = { 1, -2 };
```

створює масив { 1, -2, 0, 0, 0 }. Оскільки оголошення масиву в ***C/C++*** виділяє пам'ять під елементи масиву, але нічим її не заповнює, цю властивість ініціалізації масивів деколи використовують для заповнення новоствореного масиву нулями, наприклад, оголошення:

```
int T[100] = { };
```

створює одновимірний масив зі 100 нулів. Це ж стосується і ініціалізації двовимірних масивів. Ще одну цікавинку в ініціалізацію двовимірних масивів додає послідовний спосіб фізичного розташування їх елементів у пам'яті комп'ютера. Наприклад, дві наступні кострукції створюють однакові матриці 3x3:

```
int M1[3][3] = {{1, 2, 3}, {6, 5, 4}, {7, 8, 9}};

int M2[3][3] = {1, 2, 3, 6, 5, 4, 7, 8, 9};
```

Якщо в списку ініціалізації масиву ***M2*** забракне елементів, то компілятор допише туди нулі і масив все рівно матиме розмірність 3x3, якщо ж список буде задовгим, то при компіляції коду виникне помилка. Конструкція з порожніми квадратними дужками для ініціалізації двовимірного масиву також передбачена, але лише за умови фіксованого розміру за другою розмірністю:

```
int M4[][4] = {1, 2, 3, 6, 5, 4, 7, 8, 9};
```

Якщо невідома кількість елементів у рядку, то компілятор не має критерію для поділу елементів на рядки та стовпці, більше того, а оголошення масиву на кшалт ***int k[][]*** і взагалі позбавлене сенсу. Оголошення масиву M4 може містити як завгодно довгий список елементів, – всі вони будуть послідовно розподілені на рядки по 4 елементи, а брак елементів останнього рядка буде доповнений нулями. В коді програми на ***C++*** до задачі 405 ми ініціалізуємо масив із врахуванням його розмірів:

```
#include <iostream>

#define size 6

using namespace std;

int main()

{

    int M[][size] = { { 1, 2, 3, 4, 5, 6 }, {1,3,5,7,9,1},

            {1,4,4,3,9,1}, {1,2,3,4,5,6}, {2,2,2,3,3,5},

        {9,8,7,6,4,2 }};

    cout<<"Do you want to enter matrix elements?(y\\n)"

             <<endl;

    char choise;
```

```
cin >> choise;
if (choise == 'y')
{
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            cin >> M[i][j];
        }
    }
}
// Шукаємо максимальні елементи рядків
// і зберігаємо їх в окремому масиві
int max[size];
for (int i = 0; i < size; i++)
{
    max[i] = M[i][0];
    for (int j = 0; j < size; j++)
    {
        if (M[i][j] > max[i])
        {
            max[i] = M[i][j];
        }
    }
}
cout<<"Matrix M:"<<endl;
for (int i = 0; i < size; i++)
{
    for (int j = 0; j < size; j++)
    {
```

```
            printf("%5i", M[i][j]);
        }
        cout<<endl;
    }
    //Виконуємо перетворення
    for (int i = 0; i < size; i++)
    {
        M[i][i] = max[i];
    }
    cout<<"Transformation result:"<<endl;
    for (int i = 0; i < size; i++)
    {
        for (int j = 0; j < size; j++)
        {
            printf("%5i", M[i][j]);
        }
        cout << endl;
    }
}
```

Програма на Java.

Концепція «масиву масивів» для багатовимірних масивів в ***Java*** знаходить відображення і у способі ініціалізації масиву. Наприклад, інструкція:

```
int M[][] = {{3,4,5},{3,5,7},{1,2,3}};
```

створює і заповнює вказаними числами цілочисельну матрицю 3x3.

Такий підхід, зокрема, дозволяє створювати так звані «рвані масиви», – двовимірні масиви, в яких рядки мають різну довжину:

```
int M[][] = {{1,2,3,4,5},{37},{1,2,3}, {1,-1,1,-1}};
```

Для обходу елементів такого масиву зчитують значення поле length кожного рядка окремо, або застосовують цикл “for each”.

В коді нижче опрацьовуємо матрицю 6x6, тому важливо, щоб в ініціалізації масиву кожен рядок (група чисел у фігурних дужках) мав рівно по 6 елементів:

```
import java.util.Scanner;

public class Task405 {
  public static void main(String[] args) {
    final int size = 6;
    int[][] M = new int[][]{{1, 2, 3, 4, 5, 6},
         {1, 3, 5, 7, 9, 1}, {1, 4, 4, 3, 9, 1},
        {1,2,3,4,5,6}, {2,2,2,3,3,5}, {9,8,7,6,4,2}};
    System.out.println("Бажаєте ввести значення для" +
                "елементів матриці?(yes\\no)");
    Scanner scanner = new Scanner(System.in);
    String choise = scanner.nextLine();
    if (choise.equalsIgnoreCase("yes")) {
      for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
          M[i][j] = Integer.parseInt(scanner.nextLine());
        }
      }
    }
    // Шукаємо максимальні елементи рядків
    // і зберігаємо їх в окремому масиві
    int[] max = new int[size];
    for (int i = 0; i < size; i++) {
      max[i] = M[i][0];
      for (int j = 0; j < size; j++) {
        if (M[i][j] > max[i]) max[i] = M[i][j];
      }
    }
    System.out.println("Матриця M:");
```

```
    for (int i = 0; i < size; i++) {
      for (int j = 0; j < size; j++) {
        System.out.printf("%9d", M[i][j]);
      }
      System.out.println();
    }
    //Виконуємо перетворення
    for (int i = 0; i < size; i++) {
      M[i][i] = max[i];
    }
    System.out.println("Перетворена матриця M:");
    for (int i = 0; i < size; i++) {
      for (int j = 0; j < size; j++) {
        System.out.printf("%9d", M[i][j]);
      }
      System.out.println();
    }
  }
}
```

Програма на C#.

Логіка організації двовимірних масивів в ***C#*** подібна до ***Java,*** – у формі масиву об'єктів у пам'яті віртуальної машини, кожен з яких є масивом елементів відповідного типу. Як і в ***Java,*** тут можна користуватися «рваними масивами» та ініціалізувати їх елементами:

```
int[][] P = new int[][] {
                new int[] {37, 45},
                new int[] {1, 2, 3, 4},
                new int[7]
            };
```

Щоправда, тут сисntаксис вимагає явного виклику конструктора оператором ***new***, що в свою чергу дозволяє створювати «порожні», тобто заповнені нулями рядки (як в третьому рядку оголошення у прикладі вище).

Ми могли б скористатися таким способом, для ініціалізації нашої матриці в форматі «масиву масивів», аналогічно до коду на ***Java***. Наприклад,

```
int[][] P = new int[][] {
                new int[] {1, 3, 5, 7},
                new int[] {1, 2, 3, 4},
                new int[] {4, 5, 6, 7}
            };
```

Використання такого способу для прямокутного масиву з однаковою кількістю елементів в кожному рядку, як вже зазначалося вище, потребує ретельного контролю цієї кількості під час оголошення масиву. Розробники ***.net*** в цьому напрямку пішли далі, аніж творці ***Java*** і для роботи з матрицями та багатовимірними масивами додали ще одну синтаксичну конструкцію. Двовимірний масив 3x3, наприкла, в ***C#*** можна оголошувати та ініціалізувати так:

```
int[,] M = new int[,] { {1, 2, 3}, {2, 3, 4}, {1, 1, 1} };
```

Такий спосіб контролює «прямокутність» масиву, тобто однакову кількість елементів у кожному рядку масиву. Цим підходом скористаємося в нашому коді:

```
class Program
{
  static void Main(string[] args)
  {
    const int size = 6;
    int[,] M = new int[,]{ {1, 2, 3, 4, 5, 6},
            {1, 3, 5, 7, 9, 1}, {1, 4, 4, 3, 9, 1},
            {1, 2, 3, 4, 5, 6}, {2, 2, 2, 3, 3, 5},
                          {9, 8, 7, 6, 4, 2}};
    Console.WriteLine("Do you want to enter matrix " +
                        "elements?(yes\\no)"); ;
    string choise = Console.ReadLine();
    if (choise == "yes")
    {
      for (int i = 0; i < size; i++)
      {
```

```
    for (int j = 0; j < size; j++)
    {
      M[i,j] = Convert.ToInt32(Console.ReadLine());
    }
  }
}
// Шукаємо максимальні елементи рядків
// і зберігаємо їх в окремому масиві
int[] max = new int[size];
for (int i = 0; i < size; i++)
{
  max[i] = M[i, 0];
  for (int j = 0; j < size; j++)
  {
    if (M[i, j] > max[i]) max[i] = M[i, j];
  }
}
Console.WriteLine("Matrix M:");
for (int i = 0; i < size; i++)
{
  for (int j = 0; j < size; j++)
  {
    Console.Write(string.Format("{0, 5}",M[i, j]));
  }
  Console.WriteLine();
}
//Виконуємо перетворення
for (int i = 0; i < size; i++)
{
  M[i, i] = max[i];
```

```
    }
    Console.WriteLine("Transformation result:");
    for (int i = 0; i < size; i++)
    {
      for (int j = 0; j < size; j++)
      {
        Console.Write(string.Format("{0, 5}",M[i, j]));
      }
      Console.WriteLine();
    }
  }
}
```

Програма на Python.

В коді на ***Python*** ініціалізація змінних відіграє ключову роль, бо вона пов'язана з визначенням типу змінної. Масиви в ***Python***, точніше списки, які ми використовуємо замість масивів, ініціалізують списком значень, розділеним комами, у квадратних дужках. Елементом списку може бути будь-який об'єкт, в тому числі, і інший список. Ініціалізація двовимірного масиву в ***Python*** дуже схожа до аналогічної конструкціх в ***Java***:

```
M = [[3, 4, 5], [3, 5, 7], [1, 2, 3]]
```

В такий спосіб ініціалізують як «прямокутні» так і «рвані» масиви. Код програми для розв'язування задачі 405 на мові ***Python***:

```
import array
size = 6
M = [[1,2,3,4,5,6],[1,3,5,7,9,1],[1,4,4,3,9,1],\
      [1,2,3,4,5,6],[2,2,2,3,3,5],[9,8,7,6,4,2]]
print('Бажаєте ввести матрицю з клавіатури?(так\ні)')
choise = input()
if choise == 'так':
    for k in range(size):
        tmp = array.array('i')
        for i in range(size):
            tmp.append(int(input()))
```

```
        M[k] = tmp
# Шукаємо максимальні елементи рядків
# і зберігаємо їх в окремому масиві
max = array.array('i')
for line in M:
    m = line[0]
    for x in line:
        if x > m: m = x
    max.append(m)
print('Матриця M:')
for line in M:
    for x in line:
        print('%3i'% x, end='')
    print()
#Виконуємо перетворення
for i in range(size):
    M[i].pop(i) # вилучаємо і-ий елемент з рядка, на його
    M[i].insert(i, max[i]) # місце вставляємо максимум
print('Результат перетворення:')
for line in M:
    for x in line:
        print('%3i'% x, end='')
    print()
```

Розділ 6. Створення та використання підпрограм

Елементи процедурного програмування. Використання підпрограм до опрацювання масивів.

В задачах 421 – 435 запрограмувати обчислення заданого виразу, розбивши його на частини однотипної структури. Створити підпрограми для обчислення значень окремих частин виразу. В програмі передбачити ввід вхідних даних та вивід результатів. Вхідні дані задати самостійно.

421. $z = \frac{|v^2-u^2|^{cos(x^2-y^2)}}{|x^2-y^2|^{sin(v^2-u^2)}}$;

422. $p = \frac{\sqrt{|x\cos\alpha t - y\cos\beta t|}}{\sqrt{|y\cos\beta t - x\cos\alpha t|}}$;

423. $y = \frac{\left(\sqrt[3]{v^2-u^2}\right)}{\left(\sqrt[3]{b^2-a^2}\right)} - \frac{\left(\sqrt[3]{b^2-v^2}\right)}{\left(\sqrt[3]{a^2-u^2}\right)}$;

424. $y = \frac{sin(\omega t+\gamma)}{sin(\gamma t+\beta)} - \frac{sin(\omega t+\beta)}{sin(\beta t+\gamma)}$;

425. $y = \frac{v^{\sin p}-v^{\sin q}}{v^{\sin p}+v^{\sin q}}$, $v > 0$;

426. $z = \frac{\sqrt[5]{x\sin\alpha t + y\sin\beta t}}{\sqrt[5]{y\sin\beta t - x\sin\alpha t}}$;

427. $y = \frac{\left(\sqrt{|a+b^3|}-\sqrt{|b+a^3|}\right)}{-\sqrt{|t+(a-b)^3|}}$;

428. $s = \frac{\left(\sqrt[3]{t^2+u^2}\right)}{\left(\sqrt[3]{b^2+a^2}\right)} - \frac{\left(\sqrt[3]{b^2+t^2}\right)}{\left(\sqrt[3]{a^2+u^2}\right)}$;

429. $y = \frac{e^{\sqrt{p+q}}-e^{-\sqrt{p+r}}}{e^{-\sqrt{p+r}}+e^{\sqrt{q+r}}}$, $p, q, r > 0$;

430. $t = \frac{\sqrt[v]{|\sin p|}-\sqrt[v]{|\sin q|}}{\sqrt[u]{|\sin q|}+\sqrt[u]{|\sin p|}}$;

431. $p = \frac{(v^2+u^2)^{ln(x^2+y^2)}}{(x^2+y^2)^{lg(v^2+u^2)}}$;

432. $z = \frac{(v^2+u^2)^{sin(x^2+y^2)}}{(x^2+y^2)^{cos(v^2+u^2)}}$;

433. $y = \frac{a\,sin(\omega t+\gamma)}{b\,sin(\omega t+\beta)} - \frac{b\,sin(\omega t+\beta)}{a\,sin(\omega t+\gamma)}$;

434. $z = (x^2+y^2+2)\,log_{(x^2+y^2+2)}(a^2+b^2+2)$;

435. $p = \left(\sqrt[3]{yb^2} + \sqrt[3]{xa^2}\right) \cdot \left(\sqrt[3]{bx^2} - \sqrt[3]{ay^2}\right)$.

В задачах 436 – 450 запрограмувати обчислення заданого виразу, реалізувавши потрібні для його обчислення функції (мінімум, максимум, і т.п.) у вигляді окремої підпрограми. В програмі передбачити ввід вхідних даних та вивід результатів. Вхідні дані задати самостійно.

436. $p = max\big(min(y^2, yx), min(yx, x^2), min(b^2, x), min(y, a^2)\big)$;

437. $t = min(3 \cdot max(x^2, y^2, 2xy),\ 2 \cdot max(ax, by, a^2+b^2))$;

438. $q = max\big(min\big(\sqrt[3]{yb^2},\ \sqrt[3]{xa^2}\big),\ min\big(\sqrt[3]{bx^2},\ \sqrt[3]{ay^2}\big),\ ab\big)$;

439. $s = min(x^2, y^2, 2xy) + max(ax, ay, xy) - min(a^2, ax^2, ay^2)$;

440. $t = max(min(x^2, y^2, 2xy),\ min(px, py, x^2+y^2),\ \max(x, y, p^2))$;

441. $d = \frac{|\max(a,b,c)|^{min(a+b,a*b)}}{\min(a+c,c+b)}$;

442. $p = \frac{\sqrt{|\max(tx,ty,x^2,y^2)|}}{\sqrt{|\max(t+x,t+y,2x,y-x)|}}$;

443. $y = \frac{min(t,s,x)-\min(st,sx,s^2)}{min(t+x,x+s,3t,5s)}$;

444. $y = \frac{min(a,b,c)}{max(a+b,c)} - \frac{max(ac,bc)}{min(a,b,ab)}$;

445. $f = \frac{max\left(\sqrt{|b+a^3|},\ \sqrt{|a+b^3|},\ t^2\right)}{min\left(\sqrt{|t^3+ab|},\ \sqrt{|(a+b)^3|}\right)}$;

446. $s = \frac{min\left(\sqrt[3]{t^2+u^2}, \sqrt[3]{t^2+v^2}, \sqrt[3]{2uv}\right)}{max\left(\sqrt[3]{u^2+v^2}, \sqrt[3]{tu+tv}\right)}$;

447. $w = \frac{\min(ax+v,\ av+x,\ vx+a)}{max(ax,v)+max(av,x)}$;

448. $f = \frac{\sqrt[3]{|min(s,t)|}-\sqrt[3]{|max(s,t,v)|}}{\sqrt[3]{|max(s,u,v)|}+\sqrt[3]{|min(u,v)|}}$;

449. $p = min\left(\sqrt[3]{yb^2},\ \sqrt[3]{xa^2},\ \sqrt[3]{2xy}\right) - max\left(\sqrt[3]{ax},\ \sqrt[3]{ay},\ \sqrt[3]{bx},\ \sqrt[3]{by}\right)$;

450. $p = \frac{min(t,x,y)}{min(s,t,y)} - \frac{max(t,s)}{min(s,t,x)} + \frac{max(x,y)}{min(s,x,y)} - \frac{max(t+s,x+y)}{max(x+s,t+y)}$.

451. Реалізувати підпрограму для обчислення довжини відрізка AB на координатній площині за координатами точок $A(x_a, y_a)$ та $B(x_b, y_b)$. Використати створену підпрограму в програмі для знаходження периметра трикутника, заданого координатами його вершин.

452. Створити підпрограму для перевірки того, чи є задане натуральне число простим (натуральне число називають простим, якщо воно не має жодного натурального дільника, окрім себе та одиниці). Застосувати цю підпрограму до підрахунку кількості простих чисел в заданому наборі (масиві) натуральних чисел.

453. Розробити підпрограму для перевірки того, чи є задане натуральне число квадратом деякого натурального числа. Використовуючи її вивести усі повні квадрати із заданого набору (масиву) натуральних чисел.

454. Створити підпрограму для перевірки того, чи задане натуральне число є паліндромом (дорівнює числу, записаному тими ж цифрами в зворотному порядку). Застосувати цю підпрограму до виводу усіх паліндромів з заданого відрізка $[A;\ B]$.

455. Створити підпрограму для знаходження найбільшого спільного дільника (НСД) двох натуральних чисел за алгоритмом Евкліда. Написати програму для знаходження НСД чотирьох заданих чисел, враховуючи, що $\text{НСД}(m,n,k) = \text{НСД}(\text{НСД}(m,n),k)$.

456. Задано координати вершин трикутника $A(x_a, y_a)$, $B(x_b, y_b)$ та $C(x_c, y_c)$. Користувач вводить координати x та y деякої точки $M(x,y)$ на площині. Визначити найближчу до M вершину трикутника ABC. В програмі використати підпрограму для обчислення відстані між двома точками на площині.

457. Реалізувати підпрограму для обчислення площі рівнобедреного трикутника за відомими довжинами його основи та бічної сторони. Використати

створену підпрограму в програмі для знаходження площі бічної поверхні правильної n-кутної піраміди зі стороною основи a та бічним ребром l.

458. Створити підпрограму для знаходження суми цифр натурального числа. У заданому масиві з 20-ти цілих додатних чисел знайти число з максимальною сумою цифр, використовуючи створену підпрограму. Якщо чисел з такою сумою цифр декілька, то вивести перше з них.

459. Задано натуральне число n. Вивести усі натуральні числа з відрізка $[n; 2n]$, які можна подати у вигляді суми квадратів двох натуральних чисел. Розробити та використати підпрограму для перевірки того, чи є задане натуральне число квадратом деякого натурального числа.

460. Створити підпрограму для знаходження найбільшого спільного дільника (НСД) двох натуральних чисел за алгоритмом Евкліда. Використовуючи цю підпрограму вивести на екран усі пари взаємно простих чисел (НСД = 1) серед елементів заданого масиву натуральних чисел.

461. Розробити підпрограму для перевірки того, чи є задане натуральне число квадратом деякого натурального числа. Використовуючи її порахувати кількість повних квадратів натуральних чисел, що належать заданому відрізку $[A; B]$

462. Реалізувати підпрограму для обчислення довжини відрізка AB в тривимірних декартових координатах за відомими координатами вершин $A(x_a, y_a, z_a)$ та $B(x_b, y_b, z_b)$. Використати створену підпрограму в програмі для визначення найближчої до заданої точки $P(x, y, z)$ вершини трикутника ABC, заданого координатами його вершин.

463. Реалізувати підпрограму для обчислення довжини відрізка AB на координатній площині за координатами точок $A(x_a, y_a)$ та $B(x_b, y_b)$. Використати створену підпрограму в програмі для знаходження довжини ламаної $A_1A_2 \dots A_{10}$, координати вершин якої задано за допомогою пари масивів $X(10)$ та $Y(10)$.

464. Створити підпрограму для знаходження кількості цифр натурального числа. Заповнити довільно масив з 20-ти цілих додатних чисел. Використовуючи створену підпрограму вивести спочатку одноцифрові елементи масиву, потім двоцифрові, трицифрові і т. д.

465. Перевірити, чи введене користувачем натуральне число підтверджує гіпотезу Гольдбаха, яка полягає в тому, що будь яке парне натуральне число, більше за 2 можна подати у вигляді суми двох простих чисел (натуральне число називають простим, якщо воно не має жодного натурального дільника, окрім себе та одиниці). В програмі використати підпрограму для перевірки того, чи є вказане натуральне число простим.

466. Створити підпрограму для обчислення $n!$ з використанням рекурсії. Використати створену підпрограму для обчислення кількостей сполук без

повторів $C_m^k = \frac{m!}{(m-k)!\cdot k!}$ та кількостей розміщень $A_m^k = \frac{m!}{(m-k)!}$ без повторів для довільних заданих користувачем $m > k$.

467. Реалізувати рекурсивну підпрограму для обчислення n −го числа послідовності Фібоначчі $F_1 = 1, F_2 = 1, F_n = F_{n-1} + F_{n-2}, n = 3,4, \dots$. Використати створену підпрограму для обчислення суми

$$F_{k1} + F_{k1+1} + \cdots + F_{k2},$$

для заданих користувачам номерів $k1 < k2$.

468. Створити підпрограму для обчислення N-го члена послідовності

$$R(N) = \begin{cases} N^2 - N, якщо\ N \le 4, \\ R(N-1) - R(N-2), \ якщо\ N > 4 \end{cases}$$

з використанням рекурсії. Застосувати реалізовану підпрограму для знаходження значення виразу $\frac{R(m+k)-R(m-k)}{R(m)}$ для довільних заданих користувачем натуральних $m > k$.

469. Створити рекурсивну підпрограму для знаходження найбільшого спільного дільника (НСД) двох натуральних чисел за алгоритмом Евкліда. Використовуючи цю підпрограму вивести на екран усі пари взаємно простих чисел (НСД = 1) із заданого користувачем відрізка $[a, b]$.

470. Реалізувати рекурсивну підпрограму, яка для заданого натурального числа утворює число записане тими ж цифрами у зворотному порядку. Використати цю підпрограму для знаходження усіх паліндромів із заданого користувачем відрізка $[a, b]$.

471. Для заданого N обчислити значення функції натурального аргумента

$$Q(N) = \begin{cases} N^2 - 2N, N \le 5, \\ Q(N-1) - Q(N-2) * Q(N-4), N > 5, \end{cases}$$

використовуючи рекурсію. Обчислення значення функції оформити у вигляді окремої підпрограми, яку використати для побудови таблиці значень $G(N)$ на заданому користувачем відрізку $[A; B]$.

472. Обчислити та вивести на екран перші 50 елементів послідовності $A_0 = 1,\ A_1 = 2,\ A_2 = 3,\ A_n = A_{n-3} * A_{n-1}, n > 2$, використовуючи для їх обчислення рекурсивну підпрограму.

473. Розробити рекурсивну підпрограму, для визначення числа сполук, використовуючи рівність Паскаля $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ та співвідношення $C_n^0 = C_n^n = 1,\ n = 0,1, \dots$. Використати створену підпрограму в програмі, яка для заданого степеня n виводить на екран розклад бінома Ньютона $(1 + x)^n = \sum_{k=0}^{n} C_n^k\, x^k$.

474. Створити рекурсивну підпрограму для знаходження суми цифр натурального числа. Використовуючи створену підпрограму визначити число з найбільшою сумою цифр серед трьох заданих користувачем багатоцифрових натуральних чисел.

475. Для заданого N обчислити значення функції натурального аргумента

$$G(N) = \begin{cases} N^2, N < 10, \\ G(N-1) - G(N-2) + G(N-3), N >= 10, \end{cases}$$

використовуючи рекурсію. Обчислення значення функції оформити у вигляді окремої підпрограми, яку використати для обчислення значення виразу значення виразу $\frac{G(m-k)+G(m)}{G(m+k)}$ для довільних заданих користувачем натуральних $\mathrm{m} > k$.

476. Створити рекурсивну підпрограму для знаходження кількості цифр натурального числа. Використовуючи створену підпрограму вивести спочатку одноцифрові елементи заданого користувачем масиву, потім двоцифрові, трицифрові і т. д.

477. Розробити рекурсивну функцію для визначення цілого степеня дійсного числа, використовуючи співвідношення

$$x^n = \begin{cases} 1, & \text{якщо } n = 0, \\ \dfrac{1}{x^{|n|}}, & \text{якщо } n < 0, \\ x \cdot x^n, & \text{якщо } n > 0. \end{cases}$$

Створити програму обчислення значення виразу

$$\frac{x^k - x^m}{x^k + x^m}$$

для заданих користувачем значень x, k та m. В програмі використати створену рекурсивну функцію.

478. Створити рекурсивну підпрограму для обчислення суми n членів арифметичної прогресії за заданими першим членом a_1 та різницею d. Використовуючи створену підпрограму визначити скільки потрібно взяти послідовних членів заданої прогресії (починаючи з першого), щоб їх сума перевищила задане користувачем число M.

479. Створити підпрограму для обчислення N-го члена послідовності

$$F(N) = \begin{cases} N^2 + N, \textit{якщо } N \le 5, \\ F(N-1) - F(N-3), \ \textit{якщо } N > 5, \end{cases}$$

з використанням рекурсії. Застосувати реалізовану підпрограму для знаходження значення виразу $\frac{F(m+k)}{F(m-k)}$ для довільних заданих користувачем натуральних $\mathrm{m} > k$.

480. Реалізувати рекурсивну процедуру виводу цілого числа, записаного в системі числення з основою q $(1 < q < 10)$. Використовуючи створену підпрограму вивести введене користувачем натуральне число на екран в двійковому, трійковому, сімковому та десятковому записах.

481. Реалізувати підпрограму для з клавіатури елементів масиву дійсних чисел та використати її для заповнення двох масивів M, P кожен містить по 14 елементів. Утворити третій масив Q на основі масиву P, замінивши при цьому всі від'ємні елементи відповідними елементами масиву M (з тими ж порядковими номерами). Створити підпрограми для виводу одновимірного масиву дійсних чисел на екран та для знаходження в масиві суми додатних елементів. Використовуючи ці підпрограми вивести на екран масиви M, P, Q та суми додатних елементів у кожному з них.

482. Масиви дійсних чисел K та M, що містять по 10 елементів кожен, заповнюються випадковими числами з відрізка [-10; 10] за допомогою спеціально створеної для цього підпрограми. Масив P утворити з масиву M, замінивши в ньому всі додатні елементи максимальним елементом масиву K. Реалізувати підпрограми для виводу елементів масиву на екран та для знаходження в ньому суми від'ємних елементів і виконати їх із кожним масивом.

483. Три масиви дійсних чисел A, B та C по 12 елементів кожен заповняються елементами за правилом $\frac{(-1)^k x}{\sqrt{k}}$, $k = 1,2,\ldots,12$, де x деяке задане користувачем число (різне для кожного масиву). Реалізувати підпрограму для заповнення масивів елементами за вказаним правилом та використати її в програмі. Вивести масиви на екран, в кожному масиві знайти добуток від'ємних елементів, створивши для цього відповідні підпрограми, та викликаючи їх в основній програмі.

484. Створити підпрограму, яка заповнює масив з 15 дійсних чисел введеними з клавіатури даними та використати її для заповнення заданих в програмі масивів $M(15)$, та $T(15)$. Масив S утворити з масиву T, замінивши в ньому всі від'ємні елементи мінімальним елементом масиву M. За допомогою підпрограм вивести масиви на екран та знайти в кожному масиві кількість елементів, абсолютна величина яких менша за 1.

485. Утворити масиви дійсних чисел U, V та W, кожен містить по 11 елементів та заповнити їх елементами за допомогою підпрограми, яка заповнює масив елементами за правилом $x_k = 7\,sin\left(\sqrt[3]{(k-m)(k-n)}\right)$, $k = 1,2,\ldots,11$, де m та n – деякі задані користувачем числа (різні для кожного з масивів). Вивести масиви на екран, в кожному масиві знайти добуток елементів, абсолютна величина яких більша за 1, створивши для цього відповідні підпрограми.

486. Реалізувати підпрограму для вводу з клавіатури елементів масиву та використати її для заповнення двох масивів дійсних чисел A та B, по 14

елементів кожен. Масив C утворити за правилом $C_k = min\{A_k; B_k\}$. Створити підпрограму для виводу елементів масиву на екран та виконати її для кожного із масивів. У кожному масиві знайти середнє арифметичне додатних елементів за допомогою спеціально розробленої для цього підпрограми.

487. Створити три масиви дійсних чисел K, M та P, по 12 елементів кожен та заповнити їх за допомогою створеної для цього підпрограми випадковими числами з відрізка $[-a;\ a]$ (число a задає користувач, окремо для кожного масиву). Реалізувати підпрограми для виведення масиву на екран та для знаходження в ньому кількості елементів, з відрізка $[0;\ 3]$. Вивести кожен масив разом із шуканою величиною на екран за допомогою створених підпрограм.

488. Реалізувати підпрограму для створення та заповнення масиву з 15 дійсних чисел за правилом $sin(x^3) + 3x\ sin\ k$, $k = 1,2,\ldots,15$, x – деяке задане користувачем число. Використовуючи її утворити три масиви дійсних чисел $M(15)$, $P(15)$ та $Q(15)$. Вивести масиви на екран, в кожному масиві знайти півсуму максимального та мінімального елемента з використанням окремих, розроблених для цього підпрограм.

489. Три масиви дійсних чисел X, Y та W, складаються з 20 елементів кожен. Створити підпрограму для заповнення масиву введеними з клавіатури числами та використати її для вводу елементів заданих масивів. Вивести масиви на екран, в кожному масиві знайти різницю квадратів максимального та мінімального елементів, використовуючи окремі, розроблені для цього підпрограми.

490. Утворити масиви дійсних чисел A та B по 12 елементів кожен та заповнити їх випадковими числами з відрізка $[-50;\ 50]$ за допомогою створеної для цього підпрограми. Масив C утворити за правилом $C_k = max\{|A_k|; 2B_k\}$. Вивести масиви на екран, в кожному масиві знайти номер елемента, найближчого за величиною до 1. Вивід кожного масиву та пошук у ньому вказаної величини оформити у вигляді окремих підпрограм.

491. Створити і заповнити масиви дійсних чисел U, V та X, кожен по 16 елементів за допомогою підпрограми, яка заповняє масив випадковими числами з відрізка $[-2x;\ 3x]$ (число x задає користувач, окремо для кожного масиву). Вивести масиви на екран, в кожному масиві знайти мінімальний додатний елемент. Для виводу масиву та пошуку у ньому вказаної величини реалізувати окремі підпрограми.

492. Утворити масиви дійсних чисел M, P по 12 елементів кожен та заповнити їх введеними користувачем даними за допомогою окремої, розробленої для цього підпрограми. Масив S утворити з масиву P, замінивши в ньому всі додатні елементи мінімальним елементом масиву M. Вивести масиви на екран, в кожному масиві знайти номер максимального елемента, використовуючи окремі, розроблені для цього підпрограми.

493. Три масиви дійсних чисел A, B та C містять по 15 елементів кожен. Заповнити масиви A та B введеними користувачем даними за допомогою окремої, розробленої для цього підпрограми. Масив C утворити додаванням оберненого масиву A до масиву B, тобто $C_1 = A_{15} + B_1$, $C_2 = A_{14} + B_2$, $C_3 = A_{13} + B_3$, і т.д. Вивести масиви на екран, в кожному масиві знайти суму елементів, значення яких належать інтервалу (-2; 0). Вивід кожного масиву та пошук у ньому вказаної величини оформити у вигляді окремих підпрограм.

494. Утворити три масиви дійсних чисел M, P та Q, кожен містить по 15 елементів та заповнити їх за допомогою створеної для цього підпрограми випадковими числами з відрізка $[0;\ p]$ (число p задає користувач, окремо для кожного масиву). Вивести масиви на екран, в кожному масиві знайти різницю максимального та мінімального елемента, використовуючи для цього окремі підпрограми.

495. Створити і заповнити масиви дійсних чисел A та B, кожен по 16 елементів за допомогою підпрограми, яка заповняє масив введеними з клавіатури числами. Масив C утворити з масиву B, віднявши від кожного його елемента відповідний елемент масиву A. Розробити підпрограму для виводу масиву на екран і з її допомогою вивести усі масиви. В кожному масиві знайти суму квадратів мінімального та максимального елементів, створивши для обчислення окрему підпрограму.

496. Утворити три масиви дійсних чисел U, V та W, кожен по 11 елементів та заповнити їх елементами за допомогою підпрограми, яка заповняє масив елементами за правилом $x_k = (-1)^k\sqrt{(k+p)(k-q)^2}$, де p та q – деякі задані користувачем числа (різні для кожного з масивів), $k = 1,2,\ldots,11$. Вивести масиви на екран, створивши для цього окрему підпрограму. За допомогою ще однієї підпрограми в кожному масиві знайти різницю між максимальним елементом та середнім арифметичним усіх елементів.

497. Утворити чотири масиви дійсних чисел P, Q, R та S, по 17 елементів кожен, за допомогою підпрограми, яка заповняє масив введеними з клавіатури числами. Створити підпрограми для виведення масиву на екран та для обчислення в масиві відношення максимального його елемента до мінімального. Використовуючи створені підпрограми вивести на екран усі масиви та відношення максимального елемента до мінімального в кожному з них.

498. Три масиви дійсних чисел A, B та C утворюються з 17 елементів кожен. Розробити підпрограму, яка заповнює масив випадковими числами з відрізка $[-5x;\ 4x]$ (число x задає користувач, окремо для кожного масиву) та скористатися нею для заповнення масивів A, B та C. Реалізувати підпрограми для того, щоб вивести масиви на екран, та знайти в кожному відношення абсолютної величини суми від'ємних елементів до суми додатних елементів. Використовуючи створені підпрограми вивести масиви та вказану величину на екран.

499. Реалізувати підпрограму для створення та заповнення масиву з 12 дійсних чисел елементами за правилом $sin(p) + 3x\,sin\,k$, де $k = 1,2,\ldots,12$, p та x – деякі числа, задані користувачем при створенні масиву. Використовуючи її заповнити три масиви дійсних чисел A, B та C по 12 елементів кожен. Вивести масиви на екран, в кожному масиві знайти суму від'ємних елементів, розробивши для цього окремі підпрограми.

500. Задано три масиви дійсних чисел U, V та W, що містять по 10 елементів кожен. Реалізувати підпрограму для створення та заповнення масиву з 10 дійсних чисел елементами за правилом $2p\,cos\left(\sqrt[k+1]{k^2}\right)$, де $k = 1,2,\ldots,10$, а p – деяке задане користувачем при створенні масиву число, та використати її до заповнення масивів U та V. Масив W утворити за правилом $W_k = max\{2U_k; 5V_k\}$. Вивести масиви на екран, в кожному масиві знайти номер елемента, найближчого за величиною до числа 5, розробивши для цього окремі підпрограми.

501. Створити три масиви дійсних чисел A, B та C по 10 елементів кожен. Для заповнення масивів A та B розробити підпрограму, яка заповнює масив випадковими числами з відрізка $[-3a;\ a]$ (число a задає користувач, окремо для кожного масиву). Масив C утворити за правилом $C_k = min\{A_k; 3B_k\}$. Створити підпрограми для виводу масиву на екран та пошуку найбільшого серед від'ємних елементів масиву. Використовуючи їх вивести масиви та вказану величину на екран.

502. Утворити масиви дійсних чисел U, V та W по 14 елементів кожен. Для заповнення масивів U та V розробити підпрограму, яка заповнює масив введеними з клавіатури числами. Масив W утворити з масиву V, замінивши в ньому всі додатні елементи мінімальним елементом масиву U. Вивести масиви на екран, в кожному масиві знайти номер елемента, найближчого за величиною до 0, розробивши для цього окремі підпрограми.

503. Реалізувати підпрограму для створення та заповнення масиву з 10 дійсних чисел за правилом $2xsin\,k + cos\,k$, $k = 1,2,\ldots,10$, x – деяке задане користувачем число. Використовуючи її утворити масиви дійсних чисел $M(10)$ та $P(10)$. Третій масив Q утворити додаванням оберненого масиву M до масиву P, тобто $Q = M_{10} + P_1, M_2 = P_9 + Q_2, M_3 = P_8 + Q_3$, і т.д. Розробити ще дві підпрограми, за допомогою яких вивести масиви на екран і в кожному з них знайти кількість елементів, значення яких належать інтервалу (-1; 1).

504. Утворити три масиви дійсних чисел A, B та C, розробивши для цього підпрограму, яка заповнює масив з 16 дійсних чисел елементами за правилом $\frac{(-1)^k(2k^2+p)}{p}$, де $k = 1,2,\ldots,16$, а p – деяке задане користувачем для кожного масиву окремо. Розробити ще дві підпрограми, за допомогою яких вивести масиви на екран, в кожному масиві знайти середнє арифметичне додатних елементів.

505. Утворити три масиви дійсних чисел U, V та W, кожен по 12 елементів та заповнити їх елементами за допомогою підпрограми за правилом $x_k = (-1)^k\sqrt{(k^2+p)(k+q^2)}$, де p та q – деякі задані користувачем числа (різні для кожного з масивів), $k = 1,2,\ldots,12$. Вивести масиви на екран, в кожному масиві знайти кількість елементів, більших за 1, розробивши для цього окремі підпрограми.

506. Утворити три масиви дійсних чисел P, Q та S, по 17 елементів кожен, та заповнити їх за допомогою створеної для цього підпрограми випадковими числами з відрізка $[-p;\ 2p]$ (число p задає користувач, окремо для кожного масиву). Вивести масиви на екран, в кожному масиві знайти номер елемента, найближчого за величиною до числа 5. Для виводу масиву та пошуку у ньому вказаної величини реалізувати окремі підпрограми.

507. Створити і заповнити масиви дійсних чисел M, T та S, по 15 елементів кожен та заповнити їх елементами за допомогою підпрограми, яка заповняє масив елементами за правилом $a_k = \frac{p\,sin(qk)}{(k+1)^2}$, де p та q – деякі задані користувачем числа (різні для кожного з масивів) $k = 1,2,\ldots,15$. Вивести масиви на екран, в кожному масиві знайти кількість елементів, абсолютна величина яких більша за 5. Для виводу масиву та пошуку у ньому вказаної величини створити та використати окремі підпрограми.

508. Три масиви дійсних чисел A, B та C містять по 10 елементів кожен. Для заповнення масивів A та B створити підпрограму, що заповняє масив числами, які користувач вводить з клавіатури. Масив C утворити за правилом $C_k = 2A_k - 3B_k$. Вивести масиви на екран, в кожному масиві знайти суму додатних елементів, розробивши для цього окремі підпрограми.

509. Утворити масиви дійсних чисел U та V по 18 елементів кожен та заповнити їх випадковими числами з відрізка $[-9;\ 9]$ за допомогою створеної для цього підпрограми. Масив W утворити за правилом $W_k = max\left\{\frac{U_k}{v_k};\frac{V_k}{U_k}\right\}$. Вивести масиви на екран, в кожному масиві знайти суму елементів, абсолютна величина яких менша за 1, розробивши для цього окремі підпрограми.

510. Створити за допомогою підпрограми два масиви A та B, по 12 елементів кожен, заповнивши їх дійсними числами за правилом $x_k = (-1)^k\sqrt{(k^2+p)(k+q^2)}$, де p та q – деякі задані користувачем числа (різні для кожного з масивів), $k = 1,2,\ldots,12$.. Масив C утворити з масиву B, замінивши в ньому всі від'ємні елементи середнім значенням елементів масиву A. Вивести масиви на екран, в кожному масиві знайти елемент, значення якого найближче до числа 1, розробивши для цього окремі підпрограми.

Практикум до § 6. Створення та використання підпрограм

В алгоритмах розв'язування різних задач нерідко можна зустріти деяку послідовність команд, яку доводиться виконувати декілька разів для різних наборів даних. Така послідовність, зазвичай може бути описана як окремий алгоритм і розв'язує деяку «підзадачу», що є частиною основної. Програму, що реалізує такий алгоритм, відповідно, називають ***підпрограмою***. До розв'язування великих та складних задач часто застосовують ***декомпозицію*** – підхід що полягає в розбитті основної задачі на окремі прості задачі, більшість сучасних програм, насправді, є сукупностями великої кількості ***підпрограм***, що взаємодіють між собою.

Можливість створювати та використовувати підпрограми закладена в кожній мові програмування. В залежності від мови та технології програмування ***підпрограми*** можуть називати ***процедурами***, ***функціями*** чи ***методами***. Від першої з цих назв походить назва однієї з ***парадигм програмування***, – ***процедурне програмування***, що розглядає роботу програми як послідовність викликів відповідних процедур. У об'єктно-орієнтованому програмуванні підпрограми є членами класів, тут їх називають ***методами***.

Використання ***підпрограм*** позбавляє програмістів зайвого клопоту з розробкою та супроводом великих програм, оскільки кожну ***підпрограму*** можна розробляти, налагоджувати та вдосконалювати незалежно від решти компонентів програми. Сьогодні використання ***декомпозиції*** належить до базових навичок розробника, а розуміння процедурного підходу важливе для інженера будь-якого фаху нарівні з розумінням основних алгоритмічних конструкцій.

Підпрограма розв'язує частину загальної задачі, тому для своєї роботи повинна отримати необхідні вхідні дані від основної програми. Так само, обчисливши розв'язок поставленої їй задачі, підпрограма повинна повідомити результат своєї роботи основному коду. Перше завдання вирішується за допомогою механізму ***передачі параметрів***, друге, окрім передачі параметрів може використовувати також механізм ***повернення результату функції***. ***Підпрограму***, яка після виклику виконує запрограмовану послідовність дій та просто повертає керування основному потоку виконання називають ***процедурою***. Якщо ж ***підпрограма*** після завершення роботи отримує деякий результат та передає його основній програмі в місце свого виклику, її називають ***функцією***.

В ***C/C++*** усі підпрограми називають ***функціями***, проте тут можна використовувати функції що не повертають конкретного значення (**`void`**), вони відповідають ***процедурам*** в ***Pascal***. Подібна ситуація спостерігається і в ***Python*** та ***Go***. В об'єктно-орієнтованих технологіях (***C#***, ***Java***) підпрограма є частиною класу, її прийнято називати ***методом класу***. Методи, подібно до функцій C, можуть після свого виконання повертати основному потоку визначений результат у вигляді конкретного значення. Окремого терміну для

позначення ***методів***, що не повертають значення (**`void`**) тут також не вживають.

Взаємодія між підпрограмами пов’язана зі спільним опрацюванням деяких даних. Одним з основних способів обміну інформацією між частинами програми є механізм ***передачі параметрів*** у процедуру, функцію чи метод. Однак, з цією метою можна використовувати й інші засоби – ***глобальні змінні*** чи зовнішню пам'ять, тобто, ***файли*** на диску. Але велика кількість глобальних змінних змушує програміста ретельніше узгоджувати імена змінних, збільшує залежність програмних одиниць одна від одної, накладає обмеження на вибір імен ***локальних змінних***. Крім того, загальнодоступність глобальних змінних може призвести до їх неузгодженої зміни різними частинами програмного коду, а використання зовнішньої пам'яті у ряді випадків призводить до значного сповільнення роботи програми.

У ранніх системах програмування на IBM-сумісних комп'ютерах використовувалися два основні ***механізми передачі параметрів*** – через ***стек*** і через спеціально виділені ***регістри*** процесора. Регістрів мало, тому другий механізм застосовують рідко, а основним в більшості технологій є розміщення даних ***підпрограми*** у певній ділянці оперативної пам'яті, яку називають ***стеком***.

Окрім механізму передачі, ***параметри*** підпрограм різняться способом їх використання, – в теорії алголоподібних мов прийнято розрізняти ***вхідні***, ***вихідні*** та ***вхідно-вихідні*** параметри. При цьому розрізняють ***параметри-значення***, та ***параметри-змінні***, отже, можливі 6 різних видів параметрів. На практиці всі ці види не реалізовуються, – залежно від технології використовують 3-4 з них. Так ***вихідний параметр-значення*** має основним призначенням майже у всіх мовах повернення результату функції. А для передачі даних у підпрограму, зазвичай, використовують передачу ***за значенням***, що відповідає ***вхідному параметрові-значенню*** та передачу ***за посиланням***, що відповідає ***вхідно-вихідному параметрові-змінній***.

Підпрограма оперує деякими даними, що зберігаються у змінних. Для даних, переданих підпрограмі «ззовні», тобто параметрів, в її коді також використовують змінні, які називають ***формальними параметрами***. Їх оголошують в заголовку підпрограми при її описі, вони не містять даних, а лише показують, які саме дії виконуватиме підпрограма з отриманими даними. Під час виклику підпрограми основна програма на місце ***формальних параметрів*** передає дані для виконання підпрограми. Їх називають ***фактичними параметрами***, бо вони містять дані, з якими фактично працюватиме підпрограма. Особливості використання підпрограм та їх параметрів в різних технологіях програмування розглянемо в прикладах розв’язування задач далі.

435. Запрограмувати обчислення заданого виразу, розбивши його на частини однотипної структури. Створити підпрограми для обчислення значень окремих частин виразу. В програмі передбачити ввід вхідних даних та вивід результатів. Вхідні дані задати самостійно.

$$p = \left(\sqrt[3]{yb^2} + \sqrt[3]{xa^2}\right) \cdot \left(\sqrt[3]{bx^2} - \sqrt[3]{ay^2}\right).$$

Задачі цієї групи призначені для демонстрації доцільності використання підпрограм на прикладі «декомпозиції арифметичного виразу». Тут завдання самої програми – порахувати значення виразу при заданих значення змінних, що в нього входять. В нашому випадку для обчислення виразу потрібно задати значення a, b, x та y. Записавши оператор присвоєння та заданий вираз в його правій частині та доповнивши його вводом вхідних даних та виводом результату обчислень, отримаємо звичайну лінійну програму.

Запис самого виразу часто може виявитися доволі громіздким, зокрема, через те, що в деяких системах програмування відсутні вбудовані засоби для добування кореня кубічного чи піднесення до степеня. З іншого боку, розглянувши структуру виразу можемо бачити, що в ньому кілька разів зустрічається одна і таж арифметична конструкція вигляду $\sqrt[3]{uv^2}$. Отже, вираз в умові задачі можна записати $p = \big(f(y,b) + f(x,a)\big) \cdot \big(f(b,x) - f(a,y)\big)$, де $f(u,v) = \sqrt[3]{uv^2}$. Така підстановка, підказує програмну реалізацію, яка використовуватиме підпрограму для обчислення $f(u,v) = \sqrt[3]{uv^2}$.

Для обчислення виразів найзручніше використовувати функції, які повертають результат у місце свого викликубез додаткових параметрів для передачі результату.

Програма на Pascal.

В ***Pascal*** підпрограми поділяють на підпрограми-процедури та підпрограми-функції. ***Функція*** по завершенні свого виконання зберігає результат в спеціальну змінну і основна програма отримує його в місці виклику функції. ***Процедура*** виконує запрограмовану послідовність дій з переданими їй, в той чи інший спосіб, даними, але основній програмі конкретних значень явним чином не повертає.

Основні правила синтаксису процедур та функцій в ***Pascal***:

- ✓ *підпрограми (процедури та функції) оголошують та описують (реалізовують) в окремому розділі опису підпрограм, який слідує після опису змінних і перед блоком операторів основної програми;*
- ✓ *процедура має заголовок, що починається зі службового слова* **`procedure`***, за яким вказують її назву, список параметрів в круглих дужках і завершується знаком «`;`»;*
- ✓ *функція має заголовок, що починається зі службового слова* **`function`**, *за яким вказують її назву, список параметрів в круглих дужках та тип результату через двокрапку і завершується знаком «`;`»;*

- ✓ *опис функції чи процедури може містити такі ж розділи, як основна програма: опис констант, опис змінних, опис підпрограм і, обов'язково, блок операторів, що починається з* **begin** *та закінчується* **end**;
- ✓ *підпрограма може мати власний розділ опису процедур та функцій, тобто Pascal дозволяє використовувати вкладені підпрограми;*
- ✓ *до змінних та констант, оголошених в підпрограмі (локальні змінні та константи) можуть звертатися лише оператори з блоку операторів цієї підпрограми (або її власних підпрограм);*
- ✓ *до змінних та констант, оголошених в основній програмі (глобальні змінні та константи) можуть звертатися оператори з блоків операторів усіх її підпрограми;*
- ✓ *список формальних параметрів підпрограми записують у круглих дужках після її назви, якщо параметри одного типу, їх розділяють комами, а після списку через двокрапку вказують тип даних;*
- ✓ *групи параметрів різних типів у списку розділяють знаком "***;***";*
- ✓ *параметри процедур та функцій є вхідними параметрами-значеннями, в якості фактичного параметра їм можна передавати імена змінних, констант чи конкретні значення відповідного типу;*
- ✓ *Pascal дозволяє використовувати вхідно-вихідні параметри-змінні, для цього параметр в списку позначають службовим словом* **var**;
- ✓ *параметрам-змінним при виклику процедури в якості фактичних параметрів не можна передавати конкретні значення, оскільки вони потребують місця в пам'яті для зберігання свого значення по завершенні виконання підпрограми.*

Враховуючи усе сказане вище, реалізуємо програму розв'язування задачі 435 з використанням підпрограми-функції для обчислення значення виразу $f(u,v) = \sqrt[3]{uv^2}$. Для добування кубічного кореня використовуватимемо формулу $\sqrt[3]{t} = t^{\frac{1}{3}} = e^{\frac{1}{3}\cdot ln\, t}$.

```
program Task435;

var  a, b, x, y, p: real;

{функція для обчислення виразів з радикадлами}

function f(u, v: real): real;

begin

    {обчислений результат передається за іменем функції}

    f := exp(1/3 * ln( u * v * v));

end;


{головна програма}

begin
```

```
    writeln('Введіть значення a та b');
    readln(a, b);
    writeln('Введіть значення x і y');
    readln(x, y);
    p := (f(y, b) + f(x, a)) * (f(b, x) - f (a, y));
    writeln('Значення виразу', p:10:4);
end.
```

Неважко переконатися, що програма вище обчислює значення шуканого виразу лише для випадку коли усі вхідні дані є додатними. Достатньо ввести хоча б для однієї з величин a, b, x чи y від'ємне значення, – і ми отримаємо повідомлення про помилку. Причина цього у використанні співвідношення $\sqrt[3]{t} = e^{\frac{1}{3}\cdot ln\, t}$, бо натуральний логарифм визначений лише для додатних чисел.

З іншого боку, $\sqrt[3]{t}$ існує при будь-якого дійсному t, і вираз в умові задачі визначений для будь яких дійсних величин a, b, x та y. Тобто наша програма розв'язує поставлену задачу не для всіх можливих наборів вхідних даних.

Враховуючи, що $\sqrt[3]{0} = 0$, а $\sqrt[3]{-t} = -\sqrt[3]{t}$, перепишемо код функції ***f*** в нашій програмі наступним чином:

```
function f(u, v: real): real;
var t: real;
begin
    t := u * v * v;
    if t = 0 then f := 0;
    if t > 0 then f := exp(1/3 * ln(t));
    if t < 0 then f :=  -exp(1/3 * ln(-t));
end;
```

Тепер отримаємо програму, яка обчислює значення заданого виразу для будь-яких допустимих значень вхідних даних. Нам не довелося змінювати код основної програми, тобто використання підпрограм дозволяє розробляти і вдосконалювати програмні компоненти незалежно один від одного. Якщо б ми реалізували обчислення цього виразу у вигляді монолітної програми, то перевірку підкореневих виразів на додатність нам довелося б повторювати чотири рази.

Програма на C++.

В ***C/C++*** процедурний підхід закладено вже в основі, наприклад, виконання найпростішої програми вже починається з функції **`main`**, та, очевидно, передбачає звертання до інших функцій. В якості підпрограм тут можна використовувати лише ***функції***, проте функціям дозволено оголошувати тип результату **`void`**, який вказує, що в місце свого виклику функція не повинна повертати жодного значення. З іншого боку, будь-яку функцію, що повертає значення визначеного типу можна викликати, проігнорувавши її результат, але виконавши запрограмовану в ній послідовність дій.

Основні правила синтаксису функцій в ***C/C++***:

- ✓ *функції оголошують у будь-якому місці файлу з кодом, але до місця їх першого виклику і за межами тіла інших функцій;*
- ✓ *функція має заголовок, що починається з типу її результату, за яким вказують її назву, список параметрів в круглих дужках;*
- ✓ *опис (реалізація) функції складається з заголовку та тіла функції у фігурних дужках, в якому записують послідовність інструкцій, які повинна виконати функція під час її виклику;*
- ✓ *список формальних параметрів функції записують у круглих дужках одразу після її назви, розділяючи їх комами, опис кожного параметра складається з його типу та імені;*
- ✓ *до змінних та констант, оголошених в тілі функції (локальні змінні та константи) можуть звертатися лише оператори з блоку операторів цієї функції;*
- ✓ *формальні параметри функції є такими ж локальними змінними, як і змінні оголошені в тілі функції, але вони ініціалізуються значеннями фактичних параметрів під час виклику функції;*
- ✓ *до змінних та констант, оголошених поза тілом будь-якої функції (глобальні змінні та константи) можуть звертатися оператори з коду будь-якої функції;*
- ✓ *параметри функцій є вхідними параметрами-значеннями, в якості фактичного параметра їм можна передавати імена змінних, констант чи конкретні значення відповідного типу;*
- ✓ *в C/C++ реалізовано передачу параметрів виключно за значенням, проте для утворення вхідно-вихідні параметрів-змінних використовується передача адреси параметра за допомогою вказівника (*) або посилання (&);*
- ✓ *вихідним параметром підпрограми є результат функції, його значення передається з тіла функції за допомогою оператора* **`return`**, *при цьому тип переданого значення має співпадати з типом результату, оголошеного в заголовку функції;*
- ✓ *тип функції може бути оголошеним як* **`void`**, *в цьому випадку тіло функції може не містити жодного оператора* **`return`**;
- ✓ *тіло функції не може містити оголошення жодної іншої функції, – в C/C++ вкладення функцій заборонене.*

В ***C/C++***, на відміну від ***Pascal***, для знаходження кореня кубічного можна скористатися функцією піднесення до степеня. Проте, проблема від'ємної основи степеня залишається (див. пояснення до програми на ***Pascal***). Тому в нашій підпрограмі відразу закладемо перевірку підкореневого виразу та реалізуємо обчислення кореня від'ємного числа за допомогою співвідношення $\sqrt[3]{-t} = -\sqrt[3]{t}$.

Програма-розв'язок задачі 435 мовою C++:

```
#include <iostream>
#include <math.h>// містить функцію pow
using namespace std;
double f(double u, double v)
{   //підкореневий вираз
    double t = u * v * v;
    double result =
        t < 0? -pow(-t, 1.0 / 3): pow(t, 1.0 / 3);
    //тернарний оператор використовують для
    //короткого запису розгалуження з присвоєнням
    return result;
}
//Основна програма
int main()
{
    double a, b, x, y;
    cout << " a = ";  cin >> a;
    cout << " b = ";  cin >> b;
    cout << " x = ";  cin >> x;
    cout << " y = ";  cin >> y;
    // виклик функції для обчислення  виразу
    double p = (f(y, b) + f(x, a)) * (f(b, x) - f(a, y));
    cout << "Expression value " << p << endl;
    return 0;
}
```

Програма на Java.

Попри синтаксичну схожість ***Java*** із ***C++*** процедурний підхід в цих мовах принципово відрізняється. В ***Java*** немає функцій, – тут усі "функції" належать класам та об'єктам, і можуть бути викликані лише від їхнього імені. Їх називають ***методами***.

Основні правила синтаксису методів в ***Java***:

- ✓ *методи можна оголошувати лише в класах, вони можуть належати тільки класові, тоді їх називають статичними і позначають службовим словом* **`static`***, або належати об'єктам класу, тоді для їх виклику потрібно створювати спеціальну змінну – екземпляр класу (об'єкт);*
- ✓ *метод має заголовок, що починається зі специфікаторів доступу, типу результату, за яким вказують назву методу та список параметрів в круглих дужках;*
- ✓ *реалізація методу складається з заголовку та тіла методу у фігурних дужках, в якому записують відповідну послідовність інструкцій;*
- ✓ *список формальних параметрів методу записують у круглих дужках одразу після його назви, розділяючи їх комами, опис кожного параметра складається з його типу та імені;*
- ✓ *до змінних та констант, оголошених в тілі методу (локальні змінні та константи) можуть звертатися лише оператори з блоку операторів цього методу;*
- ✓ *змінні та константи, оголошені поза методами класу є власністю класу, їх називають полями класу, вони доступні методам класу залежно від «контексту»: статичні поля – усім методам класу, нестатичні – лише нестатичним методам класу;*
- ✓ *формальні параметри методу є такими ж локальними змінними, як і змінні оголошені в його тілі, але вони ініціалізуються значеннями фактичних параметрів під час виклику методу;*
- ✓ *параметри методів примітивних типів є вхідними параметрами-значеннями, в якості фактичного параметра їм можна передавати імена змінних, констант чи конкретні значення відповідного типу;*
- ✓ *вихідним параметром підпрограми може бути результат методу, його значення передається за допомогою оператора* **`return`**, *а його тип має співпадати з типом результату, оголошеним в заголовку методу;*
- ✓ *тип методу може бути оголошеним як* **`void`**, *в цьому випадку тіло може не містити жодного оператора* **`return`**.

Попри доволі складний механізм використання методів класу в ООП, реалізацію процедурного підходу та використання підпрограм ***Java*** можна сформулювати доволі просто. Оскільки в якості основної програми в наших студіях ми використовуємо статичний метод ***`main`*** деякого класу, то в якості підпрограми можемо використовувати інший статичний метод цього ж класу. Якщо нам потрібна підпрограма, яка лише виконає деяку послідовність дій, ми використовуємо статичний метод з типом результату ***`viod`***, якщо ж потрібна

підпрограма-функція, то створюємо статичний метод з результатом відповідного типу та оператором ***return***. Послідовність реалізації методів на рівні класу при цьому до уваги не береться, тобто, підпрограма буде знайдена незалежно від того чи її код записаний до, чи після коду основної програми.

Що ж до самої задачі, то в Java є метод для піднесення числа до степеня, який не працює з від'ємними основами, тому за допомогою співвідношення $\sqrt[3]{-t} = -\sqrt[3]{t}$ та тернарного оператора в нашій підпрограмі реалізовуємо добування кубічного кореня окремо для додатних та для від'ємних чисел:

```
package net.starbasic.tasks.p6;

import java.util.Scanner;

public class Task435 {

  //статичний метод для обчислення радикала

  static double f(double u, double v){

    double t = u * v * v;

    double result =

        t < 0? -Math.pow(-t, 1.0/3): Math.pow(t, 1.0/3);

    return result;

  }

  public static void main(String args[]){

    Scanner scanner = new Scanner(System.in);

    System.out.println("Введіть значення a та b");

    double a = scanner.nextDouble();

    double b = scanner.nextDouble();

    System.out.println("Введіть значення x і y");

    double x = scanner.nextDouble();

    double y = scanner.nextDouble();

    double p = (f(y, b) + f(x, a)) * (f(b, x) - f(a, y));

    System.out.printf("Значення виразу %10.4f\n", p);

  }

}
```

Програма на Python.

Для підпрограм на ***Python*** (як і в ***C++***) використовується термін ***функція***. Подібно до ***C++*** тут можна описувати функції, що повертають певне значення

та функції, які нічого не повертають, а лише виконують закладену в них послідовність команд.

На правилах синтаксису та використання функцій в ***Python*** позначається трансляція шляхом інтерпретації та динамічна типізація:

- ✓ *оголошення функції є виконуваним оператором, його виконання прив'язує назву функції об'єкта функції, відповідно функція має бути реалізована до її виклику;*
- ✓ *функція має заголовок, що починається службовим словом def, за яким вказують її назву та список параметрів в круглих дужках;*
- ✓ *тіло функції слідує за заголовком і відділяється від нього двокрапкою та відповідним горизонтальним відступом у новому рядку;*
- ✓ *список формальних параметрів функції записують у круглих дужках одразу після її назви, розділяючи їх комами, тип параметра не вказується;*
- ✓ *до змінних, оголошених в тілі функції (локальні змінні) можуть звертатися лише оператори з тіла цієї функції;*
- ✓ *в Python реалізовано передачу параметрів виключно за адресою, проте для змінних простих типів (immutable) при передачі фактичних параметрів створюється копія;*
- ✓ *функція може повертати результат, його значення передається з тіла функції за допомогою оператора* **`return;`**
- ✓ *оператор* **`return`** *може повертати окреме значення, або список значень у вигляді кортежу.*

Код програми для розв'язування задачі 435 на мові ***Python***:

```
import math

#Функція для обчислення значення радикала

def f(u,v):

  t = u * v * v

  res = -(-t) ** (1/3) if t < 0 else t ** (1/3)

  return res

a = float(input('Введіть значення a: '))

b = float(input('Введіть значення b: '))

x = float(input('Введіть значення x: '))

y = float(input('Введіть значення y: '))

p = (f(y, b) + f(x, a)) * (f(b, x) - f(a, y))

print('Значення виразу %10.3f' % p)
```

450. Запрограмувати обчислення заданого виразу, реалізувавши потрібні для його обчислення функції (мінімум, максимум, і т.п.) у вигляді окремої підпрограми. В програмі передбачити ввід вхідних даних та вивід результатів. Вхідні дані задати самостійно.

$$p = \frac{min(t,x,y)}{min(s,t,y)} - \frac{max(t,s)}{min(s,t,x)} + \frac{max(x,y)}{min(s,x,y)} - \frac{max(t+s,x+y)}{max(x+s,t+y)}.$$

Задача 450 за своєю постановкою схожа до задачі 435. Основна відмінність в тому, що у цій задачі не потрібно визначати який фрагмент виразу оформити у вигляді підпрограми, – потрібні для обчислення виразу підпрограми містить сама умова задачі. Обчислення цього виразу без реалізації вказаних функцій є неможливе, хоча, багато сучасних технологій програмування мають бібліотечну функцію для знаходження більшого з двох ***max(a, b)***, проте функція «менше з трьох» ***min(a, b, c)*** там відсутня.

Процес розв'язування задачі передбачає:

- *створення власних функцій для знаходження більшого з двох чисел* **max(a, b)**, *та меншого з трьох* **min(a, b, c)**
- *виклик створених функцій з вказаними фактичними параметрами для обчислення вказаних частин виразу;*
- *виконання арифметичних обчислень з отриманими результатами функцій та знаходження значення величини* **p**.

Для визначення вхідних даних, значення яких потрібно буде задавати, достатньо уважно розглянути сам вираз: окрім позначень функцій ***min*** та ***max***, тут зустрічаємо позначення невідомих величин ***s***, ***t***, ***x*** та ***y***. Очевидно, що для обчислення виразу потрібно надати цим змінним деяких числових значень. Щодо типу даних, то, за відсутності конкретних вказівок в умові задачі, в загальному випадку можемо вважати їх дійсними числами. Отже, щоб обчислити заданий вираз, наша програма повинна отримати від користувача чотири дійсні числа для невідомих величин ***s***, ***t***, ***x*** та ***y***.

Програма на Pascal.

Те, що програма повинна використовувати підпрограму-функцію обумовлено самою умовою задачі. Нагадаємо лише, що в ***Pascal*** ім'я функції де-факто є ще однією змінною підпрограми (вихідним параметром) і виконання функції повинно завершуватися присвоєнням значення цій змінній. Тобто, інструкція виду:

```
<ім'я функції> := значення;
```

в ***Pascal*** здійснює вихід з підпрограми та передачу результату основній програмі. Наші підпрограми використовуватимуть розгалуження, де можна реалізувати вихід з функції у кожній гілці, так, як показано в коді нижче на прикладі функції ***max***. Такий підхід небажаний, оскільки незрозумілий для аналізу роботи програми, краще оголосити додаткову змінну (див. функцію ***min***) для результату функції, а на виході з підпрограми передати значення цієї змінної.

Код розв'язування задачі 450 мовою ***Pascal***:

```
program Task450;
var t, s, x, y : real;
    p, p1, p2 : real;
function min(a, b, c: real): real;
var result:real;
begin
    if a < b then result := a
        else result := b;
    if c < result then result := c;
    min := result
end;
function max(a, b : real): real;
begin
    if a > b then max := a
        else max := b
end;
begin
    writeln ('Введіть вхідні дані (s,t,x,y)');
    readln(s, t, x, y);
    {Значення виразу знайдемо як суму двох частин
    віднімемо перші два дороби і наступні два}
    p1 := min(t, x, y) / min(s, t, y) -
        max(t, s) / min(s, t, x);
    p2 := max(x, y) / min(s,x,y) -
        max(t + s, x + y) / max(x + s, t + y);
    p := p1 + p2;
    writeln('Значення виразу ', p:12:5);
end.
```

Програма на C++.

В коді програми, поданому далі, варта звернути увагу на алгоритм знаходження меншого з трьох у функції ***min***: впровадження додаткової змінної ***result*** дозволяє скоротити кількість перевірок, записавши в неї меншу з перших двох величин. Далі це значення порівнюємо з третьою величиною і, за потреби, то переприсвоюємо її значення змінній ***result***.

Ще одним цікавим моментом є використання тернарного оператора у функції ***max*** (порівняйте з першим розгалуженням у функції ***min***, що виконує аналогічну дію). Тернарний оператор тут дозволяє не тільки скоротити запис розгалуження, але й забезпечує єдину точку виходу з підпрограми, оскільки повертає свій результат безпосередньо в оператор ***return***.

Програма-розв'язок задачі 450 мовою C++:

```
#include <iostream>
using namespace std;
double min(double a, double b, double c)
{
  double result;
  if (a < b)
  {
    result = a;
  }
  else
  {
    result = b;
  }
  if (c < result)
  {
    result = c;
  }
  return result;
}
double max(double a, double b)
{
  return a > b ? a : b;
}
```

```
// основна програма
int main()
{
  double s, t, x, y, p, p1, p2;
  cout << "Enter input values for s,t,x,y";
  cin >> s >> t >> x >> y;
  //Значення виразу знайдемо як суму двох частин:
  //  віднімемо перші два дороби і наступні два
  p1 = min(t, x, y) / min(s, t, y) -
    max(t, s) / min(s, t, x);
  p2 = max(x, y) / min(s, x, y) -
    max(t + s, x + y) / max(x + s, t + y);
  p = p1 + p2;
  printf("p =  %12.5f\n", p);
}
```

Програма на Java.

В програмі на ***Java*** для створення підпрограм використовуємо статичні методи ***min*** та ***max*** головного класу. Методи в межах одного і того ж класу можуть викликати один одного лише за іменем та списком фактичних параметрів, що робить їх дуже схожими на функції ***C++***. Але компілятор ***Java*** аналізує одразу код цілого класу, тому, на відміну від ***C++***, метод-підпрограма, не конче мусить бути оголошений раніше головного програми, як це і зроблено у коді далі.

```
package net.starbasic.tasks.p6;

import java.util.*;

public class Task450{

// головний метод - основна прошграма
  public static void main(String[] args) {
    System.out.println("Введіть вхідні дані (s,t,x,y)");
    Scanner scanner = new Scanner(System.in);
    double s = scanner.nextDouble();
    double t = scanner.nextDouble();
    double x = scanner.nextDouble();
```

```
    double y = scanner.nextDouble();
    /*Значення виразу знайдемо як суму двох частин а
      віднімемо перші два дороби і наступні два*/
    double p1 = min(t, x, y) / min(s, t, y) -
        max(t, s) / min(s, t, x);
    double p2 = max(x, y) / min(s,x,y) -
        max(t + s, x + y) / max(x + s, t + y);
    double p = p1 + p2;
    System.out.printf("Значення виразу %12.5f ", p);
  }
// підпрограма для знаходження меншого з трьох чисел
  static double min(double a, double b, double c) {
    double result;// змінна для результату методу
    if (a < b) result = a;
    else result = b;
    if (c < result) result = c;
    return result;
  }
// підпрограма для знаходження більшого з двох чисел
  static double max(double a, double b) {
    double result;// змінна для результату методу
    if(a > b) result = a;
    else result = b;
    return result;
  }
}
```

Програма на Python.

В програмі на ***Python*** для функцій min та max скористаємося класичним оператором розгалуження. Решта особливостей пропонуємо розглянути у коді далі:

```
# Підпрограми для знаходження найменшого з трьох чисел
def min(a, b, c):
  if a < b :
    result = a
  else:
    result = b
  if c < result:
    result = c
  return result
# функція, що повертає більше з двох заданих чисел
def max(a, b):
  if a > b:
    m = a
  else:
    m = b
  return m
# Основна програма
s = float(input('Введіть значення s: '))
t = float(input('Введіть значення t: '))
x = float(input('Введіть значення x: '))
y = float(input('Введіть значення y: '))
p1 = min(t, x, y) / min(s, t, y) - \
      max(t, s) / min(s, t, x)
p2 = max(x, y) / min(s,x,y) - \
      max(t + s, x + y) / max(x + s, t + y)
p = p1 + p2
print('Значення виразу %12.5f'% p)
```

465. Перевірити, чи введене користувачем натуральне число підтверджує гіпотезу Гольдбаха, яка полягає в тому, що будь яке парне натуральне число, більше за 2 можна подати у вигляді суми двох простих чисел (натуральне число називають простим, якщо воно не має жодного натурального дільника, окрім себе та одиниці). В програмі використати підпрограму для перевірки того, чи є вказане натуральне число простим.

В основі використання підпрограм лежить декомпозиція – поділ поставленої задач, на окремі підзадачі. Формулювання алгоритму розв'язування задачі в такому випадку зводиться до розробки алгоритмів розв'язування підзадач, та опису їх взаємодії.

В нашому прикладі маємо дві задачі:

- ✓ *для заданого користувачем числа перевірити різні можливі розклади на суму двох доданків і знайти такий, при якому обидва доданки є простими числами. При цьому вважаємо, що інструмент для перевірки того, чи вказане число є простим ми вже маємо;*
- ✓ *перевірка того, чи вказане число є простим.*

Для перегляду різних можливих записів заданого користувачем числа M у вигляді суми двох натуральних доданків $a + b = M$ будемо користуватися послідовним перебором варіантів $a = 2, 3, \ldots\ b = M - a$. Число 1 простим не вважається, тому значення першого доданка починаємо з 2. Враховуючи, переставну властивість додавання, можемо без зменшення загальності вважати, що $a \leq b$ (в протилежному випадку достатньо перейменувати доданки a та b). Неважко переконатися, що максимальне значення доданка a, при цьому, не може бути більшим за $\frac{M}{2}$.

Враховуючи зроблені вище зауваження, алгоритм роботи основної програми сформулюємо так:

- ➢ *програма отримує від користувача натуральне число M, для якого потрібно перевірити гіпотезу Гольдбаха;*
- ➢ *для послідовних значень $a = 2, 3, \ldots$ не більших за $\frac{M}{2}$ програма виконує такі дії:*
- ➢ *перевіряє, чи a є простим числом, якщо ні, то бере наступне натуральне число і повторяє зазначені дії;*
- ➢ *якщо a є простим числом, знаходимо $b = M - a$ і перевіряємо чи b є простим числом;*
- ➢ *якщо b є простим числом, то виводимо знайдену пару простих доданків повідомляємо про підтвердження гіпотези та завершуємо роботу програми;*
- ➢ *якщо b не є простим числом, то беремо наступне натуральне значення для a та повторюємо зазначені дії;*
- ➢ *якщо чергове значення a стало більшим за $\frac{M}{2}$ повідомляємо, що для M гіпотеза Гольбаха не підтверджується і завершуємо роботу програми.*

Друга задача полягає в тому, щоб для отриманого від основної програми натурального числа n переглянути різні можливі значення дільників.

Вона має кілька аспектів, аналогічних до першої. Так перевірку подільності числа n на деяке натуральне p можна вважати записом n у вигляді добутку двох натуральних множників $p * q = n$, а з урахуванням переставної властивості множення можна вважати, що $p \leq q$. З останньої нерівності бачимо, що максимальне значення p не може бути більшим за $\sqrt{n}$.

Алгоритм роботи підпрограми буде наступним:

- *підпрограма отримує від основної програми натуральне число n, яке потрібно перевірити;*
- *для послідовних значень $p = 2, 3, \ldots$, не більших за $\sqrt{n}$ підпрограма виконує такі дії:*
- *перевіряє, чи n ділиться націло на p, якщо ні, то бере наступне натуральне число і повторяє перевірку;*
- *якщо n ділиться націло на p, то підпрограма повертає негативну відповідь основній програмі (число n не є простим числом) та завершує роботу;*
- *якщо величина p досягнула значення більшого за $\sqrt{n}$ то підпрограма повертає основній програмі відповідь, що n є простим числом та завершує свою роботу.*

Щоб не відволікати увагу читача від процедурного програмування циклічно-розгалуженими конструкціями, графічні зображення описаних вище алгоритмів не подаємо. Перейдемо одразу до їх програмної реалізації.

Програма на Pascal.

Як відомо, програми на ***Pascal*** можуть послуговуватися двома видами підпрограм: процедурами та функціями. В нашому алгоритмі підпрограма повинна повертати результат основній програмі, тобто її поведінка ближча до підпрограми-функції, яку використовуватимемо в коді прикладу.

Оскільки функція виконує певну перевірку, то тип результату функції міг би бути булевим (***true/false***). Такий тип в ***Pascal*** підтримується і має назву ***boolean***, проте в коді далі ми продемонструємо як у такій ситуації організувати взаємодію між підпрограмами за допомогою цілочисельного типу. Для цього домовимося, що підпрограма перевірки числа буде повертати 0, якщо число просте та перший знайдений цілий дільник числа, якщо воно складене.

Решта зауважень щодо розв'язування задачі 465 подано в формі коментарів в наведеному нижче ***Pascal***-коді.

```
program Task465;

var m, s, a, b, max_a: integer;

function isprime(n : integer) : integer;

var p, max_div, result : integer;
```

```
begin
  result := 0; {для початку перевірки припускаємо, що
    отримане число просте, якщо в ході перевірки знайдеться
    який-небудь його дільник, то він буде результатом
    функції, якщо ж ні, то функція поверне 0 }
  max_div := round(sqrt(n));
  for p := 2 to max_div do
    begin
      if n mod p = 0 then
        begin
          result := p;
          break;
      end
    end;
  isprime := result;
end;
begin
  write('Введіть парне натуральне число для ');
  writeln('перевірки гіпотези Гольдбаха >>');
  readln(m);
  max_a := m div 2 + 1;
  for a := 2 to max_a do
  begin
    s := isprime(a);
    {якщо s>0, то a складене, переходимо до
    наступного значення}
    if s > 0 then continue;
    b := m - a;
    s := isprime(b);
    if s > 0 then
```

```
      continue
    {якщо s>0, то b складене, переходимо до
    наступного значення a}
    else
      {якщо s=0 і для a, і для b, то потрібний розклад у
      вигляді суми простих чисел знайдено,
      гіпотезу підтверджено }
      break
  end;
  if a = max_a then
    writeln('Гіпотеза Гольдбаха не підтверджується')
  else
  begin
    writeln(m, ' = ', a, ' + ', b);
    writeln('Гіпотезу Гольдбаха підтверджено')
  end
end.
```

Програма на C++.

Застосуємо такий же підхід, як в коді вище на мові Паскаль. Для підпрограми використовуємо функцію, в ***C/C++*** усі підпрограми є функціями. Результат перевірки простого числа так само, як і в попередньому коді будемо повертати числовим значенням: 0 – число просте, ненульовий результат дорівнює першому знайденому дільнику складеного числа.

В коді нижче функція ***isprime*** задекларована двічі, – до і після головної функції. Це пов'язано з вимогою ***C++*** щоб усе, що використовується в коді програми (змінні, константи, типи даних, функції і т.п.) було оголошене до його використання. Відповідно до цього принципу підпрограми у файлі з кодом потрібно записувати перед програмою (функцією ***main***). Якщо в програмі багато підпрограм, такий підхід псує читабельність коду. Щоб уникнути цього, використовують так звані ***прототипи*** – попередні оголошення функцій. В прототипах вказують лише назву функції, тип її результату та кількість і типи її параметрів. Його записують перед головною функцією і завершують знаком «;». ***Прототип*** вказує компілятору, що програма буде користуватися такою функцією, – компілятор повинен сам знайти де розміщена її реалізація.

Програма-розв'язок задачі 465 мовою ***C++***:

```
#include <iostream>
#include <cmath>
using namespace std;

int isprime(int);//прототип функції isprime
int main()
{
    cout << "Enter an even natural number"
        << "to test the Goldbach hypothesis:\n";
    int m;
    cin >> m;
    int max_a = m / 2 + 1;
    int a, b, s;
    for (a = 2; a < max_a; a++)
    {
        s = isprime(a);
        // якщо s > 0, то a складене, переходимо до
        //    наступного значення
        if (s > 0) continue;
        b = m - a;
        s = isprime(b);
        if (s > 0) continue;
        // якщо s > 0, то b складене, переходимо до
        // наступного значення a
        else
            // якщо s = 0 і для a, і для b, то потрібний
            //  розклад у  вигляді суми простих чисел
            //   знайдено, гіпотезу підтверджено
            break;
    }
```

```
    if (a == max_a)
    {
        cout << "Goldbach's hypothesis is not confirmed\n";
    }
    else
    {
        cout << m << " = " << a << " + " << b << endl;
        cout << "Goldbach's hypothesis confirmed\n";
    }
}

//реалізація функції isprime
int isprime(int n)
{
    int result = 0;
    int max_div = int(sqrt(n));
    for (int p = 2; p <= max_div; p++)
    {
        if (n % p == 0)
        {
            result = p;
            break;
        }
    }
    return result;
}
```

Програма на Java.

Код на ***Java*** буде, як завжди, синтаксично схожим на ***C++***-реалізацію нашого алгоритму, хоча слід звернути увагу на такі моменти:

- ✓ *для підпрограм, аналогічних до С-функцій, в Java використовують статичні методи класу, тобто в класі з програмою окрім методу* **main**, *буде ще однин метод;*
- ✓ *послідовність реалізацій програми і підпрограми для методів класу не має значення, зазвичай, підпрограми записують після main-методу;*
- ✓ *для методів з префіксом* **is-** *в Java прийнято повертати логічний результат типу boolean, але можна реалізувати і використаний вище підхід з цілочисельним результатом.*

Решту нюансів пропонуємо розглянути в наступному коді та коментарях до нього:

```
import java.util.Scanner;
public class Task465 {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Задайте парне натуральне " +
          "число для перевірки гіпотези Гольдбаха:\n");
    int m = scanner.nextInt();
    int max_a = m / 2 + 1;
    int a, b = 0;
    for (a = 2; a < max_a; a++)
    {
      // якщо a складене, переходимо до наступного значення
        // boolean-результат методу можемо використовувати
      // в якості умови для оператора розгалуження
      if (!isPrime(a)) continue;
       // тут вже відомо, що a просте число, бо, якщо a
       // не є простим оператор if вище перерве виконання
       // поточної ітерації і код нижче не виконуватиметься
```

```
      b = m - a;
       // якщо при цьому число b також виявиться простим
       // то цикл пошуку простих доданків можна завершити
       // достроково і перейти до повідомлення результату
      if(isPrime(b)) break;
    }
    if (a == max_a)
    {
      System.out.println("Гіпотезу Гольдбаха " +
                                    "не підтверджено");
    }
    else
    {
      System.out.printf("%10d  = %10d + %10d\n", m, a, b);
      System.out.println("Гіпотезу Гольдбаха" +
                                         " підтверджено");
    }
}

    // для "внутрішнього" використання в методі main
    // метод-підпрограму можна оголошувати з будь-яким
    // рівнем доступу, в т. ч. private
    // статичний public-метод можна використовувати
    // в інших класах
    public static boolean isPrime(int n)
    {
      boolean result = true;
      int max_div = (int) Math.sqrt(n);
```

```
      for (int p = 2; p <= max_div; p++)
      {
        if (n % p == 0)
        {
          result = false;
          break;
        }
      }
      return result;
    }
}
```

Програма на Python.

В коді на ***Python*** також використовуватимемо функцію з булівським результатом. Явним чином тип результату функції тут не вказується, – він визначається з даних, які повертає функція. Передаючи булівські константи ***True*** чи ***Flase*** оператором ***return*** ми власне вказуємо на такий тип функції.

Оголошення функції оператором ***def*** в ***Python***, так само, як в ***C++*** має передувати коду, який її використовує. Це випливає з використання інтерпретації як способу трансляції та виконання коду.

Код програми для розв'язування задачі 465 на ***Python***:

```
import math
def isprime(n):
  result = True
  max_div = round(math.sqrt(n))
  for p in range(2, max_div + 1):
    if (n % p == 0):
      result = False
      break
  return result
```

```
print('Введіть парне натуральне число для ')
print('перевірки гіпотези Гольдбаха >>')
m = int(input())
max_a = int(m / 2) + 1 # максимально можливе
# значення першого доданка шуканої суми
for a in range(2, max_a + 1):
  # якщо a складене, переходимо до
  # наступного значення
  if not isprime(a):
    continue
  b = m - a
  if not isprime(b):
    continue
  # якщо  b складене, переходимо до
  #  наступного значення a
  else:
    break
if (a == max_a):
  # якщо можливе значення першого доданка
  # досягло максимуму, то потріний розклад
  # у вигляді суми не знайдено
  print('Гіпотеза Гольдбаха не підтверджується')
else:
  print(m, ' = ', a, ' + ', b);
  print('Гіпотезу Гольдбаха підтверджено')
```

480. Реалізувати рекурсивну процедуру виводу цілого числа, записаного в системі числення з основою q $(1 < q < 10)$. Використовуючи створену підпрограму вивести введене користувачем натуральне число на екран в двійковому, трійковому, сімковому та десятковому записах.

Рекурсія в дизайні

Рекурсія це спосіб визначення об'єкта через самого себе з попереднім заданням одного чи декількох його базових випадків чи методів. Цей підхід застосовується в різних галузях знань, але найчастіше у математиці та інформатиці. До прикладу, із шкільного курсу математики кожному відомі рекурентні формули в означенні арифметичної та геометричної прогресій. Ще один загальновідомий приклад з математики, – факторіал натурального числа $n! = n * (n-1) * ... * 2 * 1$, його рекурсивний запис виглядає так:

$$\begin{cases} n! = n * (n-1)!, \\ 1! = 1. \end{cases}$$

У процедурному програмуванні рекурсія будується на виклику підпрограми: підпрограма може викликати іншу підпрограму, в тому числі й саму себе. Так у прикладі з факторіалом можна реалізувати функцію $f(n)$, яка повертатиме 1, якшо $n = 1$, а для всіх інших натуральних $n > 1$ множитиме число n на результат свого ж виклику з параметром $n - 1$. Наприклад, для числа $n = 3$ матимемо:

$$f(3) = 3 * f(2);$$

$$f(2) = 2 * f(1);$$

$$f(1) = 1;$$

$$f(3) = 3 * f(2) = 3 * 2 * f(1) = 3 * 2 * 1 = 6.$$

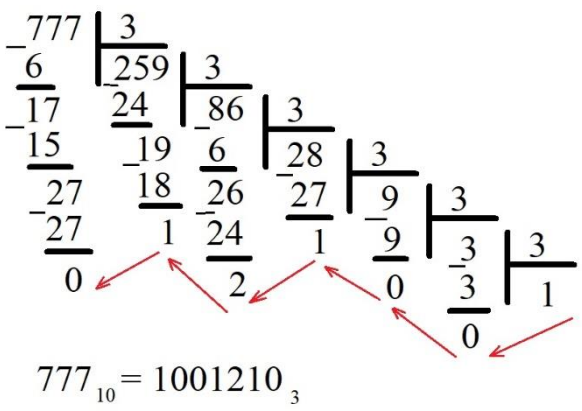
На Python така функція, до прикладу, могла б виглядати наступним чином:

```
def f(n):
    result = 1 if n == 1 else n * f(n - 1)
    return result
```

Рекурсія має прямий та зворотній хід. Поступове розвинення факторіала в добуток факторіалів попередніх натуральних чисел в нашому прикладі є прямим ходом рекурсії, а обчислення отриманого добутку – зворотнім. Кількість рекурсивних викликів при цьому називають глибиною рекурсії. Обчислення факторіала $f(1) = 1$ в цьому прикладі є «точкою переходу» між прямим та зворотним ходом рекурсії. Зрозуміло, що в реалізації кожної рекурсивної процедури/функції така «точка переходу» має бути, інакше послідовність рекурсивних викликів підпрограм може виявитися нескінченною, вони швидко заповнять відведену пам’ять (стек) і програма буде завершена аварійно.

Використання рекурсії пов’язане з додатковим навантаженням на оперативну пам’ять ПК. Тому в професійній розробці рекурсії намагаються уникати, заміняючи її циклічними конструкціями. Наприклад, той же факторіал числа можна обчислити, використовуючи звичайний цикл з лічильником. Але в навчальних курсах рекурсії приділяють належну увагу як цікавому з точки програмування механізму, який до того ж добре ілюструє процес виклику та взаємодії підпрограм.

Задача про перевід десяткового числа в позиційну систему числення з основою q алгоритм полягає в послідовному діленні заданого числа з остачею на основу системи числення. Ділення з остачею продовжується доти, доки не отримаємо число, менше за q, яке і буде першою цифрою шуканого запису заданого числа в новій системі числення. А отримані в ході ділення остачі, записані в зворотному порядку, складають решту його цифр (див. приклад на малюнку).



Такий ітераційний процес зручно виконувати за допомогою рекурсії з огляду на потребу зворотного запису остач від ділення. Прямий хід рекурсій буде виконувати ділення, зберігаючи поточну остачу в поточному рекурсивному виклику підпрограми. Зворотній же запустить вивід цих остач якраз у потрібному порядку, враховуючи, що завдання нашої функції просто вивести отримане число на екран.

Алгоритм роботи нашої підпрограми буде таким:

- *якщо передане підпрограмі ціле число менше за основу системи числення його потрібно вивести на екран без будь-яких інших дій;*
- *якщо число, передане підпрограмі, не менше за основу системи числення, то потрібно виконати ділення цього числа на основу системи числення націло, отриману частку передати в наступний рекурсивний виклик підпрограми, а остачу після цього вивести на екран.*

Програма на Pascal.

Виклик процедур та функцій в ***Pascal*** підтримує стандартний механізм стеку, виклик підпрограми можна здійснити з її власного блоку операторів. З огляду на те, що програма виводитиме результат безпосередньо на дисплей спробуємо в цьому прикладі скористатися підпрограмою-процедурою.

```
program Task480;
var m: integer;
procedure print_posix(q, n : integer);
begin
  {якщо число n в системі з основою q одноцифорове}
  if n < q then write(n) {виводимо його}
  else
    begin
    { рекурсивний виклик функції для виводу старших
    розрядів числа з параметром – часткою від ділення}
    print_posix(q, n div q);
    write(n mod q); {виводимо останню цифру числа n}
    end {у системі числення з основою q}
  end;
begin
  writeln('Введіть натуральне число');
  readln(m);
  print_posix(2, m);
  write('(2) = ');
  print_posix(3, m);
  write('(3) = ');
  print_posix(7, m);
  write('(7) = ');
  writeln(m, '(10)');
end.
```

Програма на C++.

Рекурсивні функції в ***C++***, зазвичай, повертають деякий результат. Враховуючи специфіку задачі – вивід результату роботи на екран, тут використаємо функцію типу ***void*** без оператора ***return***.

```
#include <iostream>
using namespace std;
//прототип функції рекурсивного виводу числа
void print_posix(int, int);
//головна програма (функція)
int main()
{
    int m;
    cout << "Enter number:\n";    cin >> m;
    print_posix(2, m);//вивід числа в двійковій системі
    cout << "(2) = ";
    print_posix(3, m); //вивід числа в трійковій системі
    cout << "(3) = ";
    print_posix(7, m); //вивід числа в сімковій системі
    cout << "(7) = ";
    cout <<m <<"(10)"<<endl;
}
//функція виводу числа n в системі з основою q
void print_posix(int q, int n)
{
    if (n < q)
    {//число n в системіз основою q одноцифорове
        cout << n;
    }
    else
    {//рекурсивний виклик функції для виводу старших
        print_posix(q, n / q);//розрядів числа
        cout << n % q;// виводимо останню цифру
    }
}
```

Програма на Java.

Статичні методи класів ***Java*** допускають рекурсивні виклики. Враховуючи завдання програми вивести результат на екран, використовуємо рекурсивний метод типу void. Решта особливостей Java пропонуємо розглянути в наступному коді:

```
import java.util.Scanner;

public class Task480 {
  static void print_posix(int q, int n) {
    if (n < q){
      System.out.printf("%1d",n);
    }
    else {
      print_posix(q, n / q);
      System.out.printf("%1d",n % q);
    }
  }
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Введіть ціле число");
    int m =scanner.nextInt();
    print_posix(2, m);
    System.out.print("(2) = ");
    print_posix(3, m);
    System.out.print("(3) = ");
    print_posix(7, m);
    System.out.print("(7) = ");
    System.out.println(m + "(10)");
  }
}
```

Програма на Python.

Механізм виклику функцій ***Python*** також допускає рекурсію. Використовуватимемо функцію без оператора ***return***.

Специфіка задачі вимагає виводу цифр числа в стрічку з різних місць коду, для цього в функції виводу ***print()*** використовується додатковий параметр ***end*** зі значенням порожньої стрічки. Нагадаємо також, що ***Python*** для цілочисельного ділення використовує спеціальний оператор "//".

Код програми для розв'язування задачі 480 на мові ***Python***:

```
def print_posix(q, n):
  if n < q :
    #число n в системі з основою q одноцифорове
    print(n, end='')
  else:
    #рекурсивний виклик функції для виводу старших
    # розрядів числа
    print_posix(q, n // q)
    print(n % q, end='') # виводимо останню цифру
    # числа n в системі числення з основою q
# Основна програма
m = int(input('Введіть число ціле : '))
print_posix(2, m) #вивід числа в двійковій системі
print('(2) = ', end='')
print_posix(3, m) # вивід числа в трійковій системі
print('(3) = ', end='')
print_posix(7, m) # вивід числа в сімковій системі
print('(7) = ', end='')
print(m, '(10)')
```

510. Створити за допомогою підпрограми два масиви A та B, по 12 елементів кожен, заповнивши їх дійсними числами за правилом $x_k = (-1)^k\sqrt{(k^2+p)(k+q^2)}$, де p та q – деякі задані користувачем числа (різні для кожного з масивів), $k = 1,2,\ldots,12$.. Масив C утворити з масиву B, замінивши в ньому всі від'ємні елементи середнім значенням елементів масиву A. Вивести масиви на екран, в кожному масиві знайти елемент, значення якого найближче до числа 1, розробивши для цього окремі підпрограми.

Використання підпрограм для опрацювання масивів має певні особливості, бо пов'язане з передачею а якості параметрів цілих наборів даних, а не одиничних змінних. Проте у кожної технології програмування вони індивідуальні, бо випливають зі специфіки фізичної реалізації масивів в цій технології. Тому тут лише коротко зупинимося на постановці самої задачі.

За умовою задачі програма повинна опрацьовувати три масиви чисел по 12 елементів. Для заповнення перших двох потрібно створити підпрограму, заповнення третього масиву C індивідуальне, за окремим правилом. Оскільки код, що заповнює масив C використовуватиметься лише один раз, то створювати для нього окрему підпрограму недоцільно. Після утворення масивів, усі три масиви потрібно вивести на екран, та в кожному знайти елемент, значення якого найближче до 1. А для цього, за умовою, потрібно організувати ще пару підпрограм.

У підсумку наш код міститиме три підпрограми: одну для заповнення масиву за вказаним правилом, другу – для виводу елементів масиву на екран, а третю – для пошуку в масиві елемента, значення якого найближче до 1. Загальна схема роботи програми буде такою:

- *програма за допомогою створеної підпрограми заповнює масиви A та B;*
- *на основі створених масивів A та B програма за вказаним правилом утворює третій масив* C*;*
- *до кожного зі створених масивів програма застосовує підпрограму для виводу на екран, та підпрограму для пошуку елемента, значення якого найближче до 1, а знайдене значення виводить на екран*.

Щодо підпрограм для заповнення нового масиву за заданим правилом, то тут можливі два різні підходи: масив створюється в основній програмі і передається у підпрограму для заповнення даними як вхідно-вихідний параметр, або масив створюється і заповнюється даними безпосередньо у підпрограмі і повертається як результат її роботи у вигляді вихідного параметра. Перший дозволяє програмі контролювати створення об'єктів, проте потребує додаткової передачі даних. Другий, мінімізуючи передачу даних, навпаки позбавляє програму контролю за створенням власних об'єктів. На практиці, вибір одного з цих двох підходів узгоджується з фізичним механізмом зберігання масивів в тій чи іншій технології програмування.

Програма на Pascal.

Концепція опрацювання масивів в ***Pascal*** краще підходить для передачі масиву на заповнення даними з основної програми в підпрограму, ніж для створення масиву у самій підпрограмі. Для того, щоб заповнені підпрограмою значення елементів збереглися в масиві і були доступні в основній програмі, масив потрібно передавати як параметр-змінну (***var***). В двох інших підпрограмах (вивід на екран і пошук елемента) масив достатньо передавати як параметр-значення, оскільки вони не змінюють елементів масиву.

Для зручності оголошення та передачі параметрів-масивів в ***Pascal*** можна оголосити власний тип даних, зафіксувавши в ньому тип елементів масиву та його розмір. Для цього в ***Pascal***-програмах використовують необов'язковий розділ опису типів користувача ***type*** (див. код нижче).

Кількість елементів, за умовою задачі, фіксована і однакова для всіх масивів. Її зручно оголосити глобальною константою, яка буде доступна у всіх процедурах та функціях.

Підпрограму виводу елементів масиву на екран логічніше оформити у вигляді процедури, – вона отримує лише вхідні дані, а результат її роботи буде виведено на дисплей, підпрограму для заповнення масиву також. А от для пошуку елемента, значення якого за величиною найближче до 1, краще створити підпрограму-функцію, оскільки вона повинна повернути після опрацювання масиву шукане значення.

```
program Task510;
const SIZE = 12;
type my_array = array[1..SIZE] of real;

var A, B, C : my_array; i : integer;
    SumA, min_d1, avg: real;
{в підпрограмі для заповнення масиву сам масив
 повинен бути вхідно-вихідним параметром (var)}
procedure create_array(var m: my_array);
var p, q: real; i, sign: integer;
begin
    sign := -1;// -1^n при n = 1
    writeln('p = '); readln(p);
    writeln('q = '); readln(q);
```

```
    for i := 1 to SIZE do
    begin
        m[i] := sign * sqrt((i * i + p)*
            (i + q * q));
        sign := sign * -1; // обчислення -1 ^ (k + 1)
    end;
end;

procedure print_array(m : my_array);
var i: integer;
begin
    for i := 1 to SIZE do
        write(m[i]:7:2);
    writeln(); {завершення стрічки з виводом масиву}
end;

function min_dist_el(m: my_array): real;
var min, dist : real; i: integer;
begin
    min := m[1]; {початкове наближення для пошуку}
    { його відстань до 1}
    dist := abs(min - 1);
    for i := 1 to SIZE do
        if abs(m[i] - 1) < dist then
        begin {знайдено новий елемент найближчий до 1}
            min := m[i];
            dist := abs(min - 1);
        end;
    min_dist_el := min;
end;
```

```
begin
    {заповнення масивів за допомогою підпрограми}
    create_array(A);
    create_array(B);
    {заповненя  третього масиву C}
    SumA := 0;
    for i := 1 to SIZE do
        SumA := SumA + A[i];
    avg := SumA / SIZE;
    for i := 1 to SIZE do
        if B[i] < 0 then
            C[i] := avg {заміняємо середнім}
        else
            C[i] := B[i]; {беремо з масиву B}
    { закінчення циклу заповнення масиву C }

    {опрацювання масивів}
    writeln('Array A:');
    print_array(A); {вивід масиву A}
    min_d1 := min_dist_el(A); {пошук найближчого до 1}
    writeln('Closess to 1 is ', min_d1:10:2);
    writeln('Array B:');
    print_array(B); {вивід масиву B}
    min_d1 := min_dist_el(B); {пошук найближчого до 1}
    writeln('Closess to 1 is ', min_d1:10:2);
    writeln('Array C:');
    print_array(C); {вивід масиву C}
    min_d1 := min_dist_el(C); {пошук найближчого до 1}
    writeln('Closess to 1 is ', min_d1:10:2)
end.
```

Програма на C++.

В ***C/C++*** опрацювання масивів, так само, як і передача їх у функцію реалізована за допомогою адреси даних в пам'яті (вказівників). Це робить усі масиви вхідно-вихідними параметрами функцій автоматично, оскільки при виклику підпрограми передається не сукупність значень елементів масиву, а сама адреса їх розміщення у пам'яті ПК. Підпрограма в такому випадку працює з тими ж даними, що і основна програма. Тобто, тут, як і в програмі на ***Pascal***, оголошуватимемо та створюватимемо масиви в головній функції.

Описувати для масивів власний тип даних не будемо (в ***C++*** така можливість, безперечно, є), а лише оголосимо фіксований розмір масивів за допомогою макропідстановки ***#define***. Підпрограми для заповнення та виводу масивів реалізуємо у вигляді функцій типу ***void***. Функція для пошуку елемента, значення якого за величиною найближче до 1, повертатиме значення того ж типу, що й елементи масиву.

Тут, як і в коді програми на ***Pascal***, ми використовуємо для параметрів підпрограм масиви фіксованого розміру. Цього цілком достатньо в контексті нашої задачі, хоча обидві технології допускають оголошення в якості параметра масиву без визначеного наперед розміру, наприклад, ***array of real*** в Pascal, чи ***double[]*** в C/C++.

```
#include <iostream>
#define SIZE 12
using namespace std;

// прототипи функцій оголошуються до основної програми
void create_array(double m[SIZE]);
void print_array(double m[SIZE]);
double min_dist_el(double m[SIZE]);

void main()
{ // масиви, оголошені а головній функції
  // створюють безпосереднто інструкцією опису
    double A[SIZE], B[SIZE], C[SIZE];
```

```
//заповнення масивів за допомогою підпрограми
create_array(A);
create_array(B);

// заповнення третього масиву C
double SumA = 0, min_d1;
for (int i = 0; i < SIZE; i++)
{
    SumA += A[i];
}
double avg = SumA / SIZE;
for (int i = 0; i < SIZE; i++)
{
    if (B[i] < 0)
    {
        C[i] = avg;// заміняємо середнім
    }
    else
    {
        C[i] = B[i];// беремо з масиву B
    }
}
cout << "Array A:\n";
print_array(A);// вивід масиву
min_d1 = min_dist_el(A);// пошук найближчого до 1
printf("\nClosess to 1 is %10.2f\n", min_d1);
cout << "Array B:\n";
print_array(B);
min_d1 = min_dist_el(B);// пошук найближчого до 1
printf("\nClosess to 1 is %10.2f\n", min_d1);
```

```
    cout << "Array C:\n";
    print_array(C);
    min_d1 = min_dist_el(C);// пошук найближчого до 1
    printf("\nClosess to 1 is %10.2f\n", min_d1);
}

// реалізації функцій, оголошених прототипами вище
void create_array(double m[SIZE])
{
    int sign = -1;// -1^n при n = 1
    double p, q;
    cout << "p = "; cin >> p;
    cout << "q = "; cin >> q;
    for (int i = 0; i < SIZE; i++)
    {
        m[i] = sign * sqrt(((i + 1) * (i + 1) + p)*
            (i + 1 + q * q));
        sign *= -1; // обчислення -1 ^ (k + 1)
    }
}

void print_array(double m[SIZE])
{
    for (int i = 0; i < SIZE; i++)
    {
        printf("%7.2f, ", m[i]);
    }
}
```

```
double min_dist_el(double m[SIZE])
{
    double min = m[0];// початкове наближення
    // його відстань до 1
    double dist = abs(min - 1);
    for (int i = 0; i < SIZE; i++)
    {
        if (abs(m[i] - 1) < dist)
        {// знайдено новий елемент найближчий до 1
            min = m[i];
            dist = abs(min - 1);
        }
    }
    return min;
}
```

Програма на Java.

Зберігання масивів в ***Java*** принципово відрізняється від масивів ***C++***. Тут програма не має доступу до даних в пам'яті ПК, бо всіма його ресурсами розпоряджається віртуальна машина JVM. Масиви тут є специфічним різновидом об'єкта та створюються і контролюються віртуальною машиною. Тому, в контексті задачі, доцільнішим буде створення масиву безпосередньо у підпрограмі та передача його основній програмі в якості результату. При такому підході задавати фіксовану кількість елементів у вигляді окремої константи не обов'язково, тим більше, що масиви в ***Java*** мають поле ***length***, яке містить фактичний розмір масиву. Проте ми створимо поле-константу SIZE аналогічно до прикладів на ***C++*** та ***Pascal***.

Послідовність розміщення методів в Java-класі значення не має, тому спочатку реалізуємо основну програму з викликами підпрограм, а потім вже самі підпрограми. Решта особливостей ***Java*** щодо опрацювання масивів з допомогою підпрограм пропонуємо розглянути в наступному коді:

```
package net.starbasic.tasks.p6;

import java.util.Scanner;


public class Task510 {
  private static final int SIZE  = 12;
```

```
// головна програма – метод main
  public  static  void main(String args[]) {
    //створення масивів за допомогою підпрограми
    double A[] = create_array();
    double B[] = create_array();
    // створення масиву C
    double SumA = 0, min_d1;
    for (int i = 0; i < SIZE; i++) {
      SumA += A[i];
    }
    double avg = SumA / SIZE;
    double C[] = new double[SIZE];
    for (int i = 0; i < SIZE; i++) {
      if (B[i] < 0) {
        C[i] = avg;// заміняємо середнім
      } else {
          C[i] = B[i];// беремо з масиву B
      }
    }

    System.out.println("Масив A:");
    print_array(A);// вивід масиву
    min_d1 = min_dist_el(A);// пошук найближчого до 1
    System.out.printf("\nНайближчий до 1 елемент %10.2f\n",
                                                    min_d1);
    System.out.println("Масив B:");
    print_array(B);
    min_d1 = min_dist_el(B);// пошук найближчого до 1
    System.out.printf("\nНайближчий до 1 елемент %10.2f\n",
                                                    min_d1);
```

```
  System.out.println("Масив C:");
  print_array(C);
  min_d1 = min_dist_el(C);// пошук найближчого до 1
  System.out.printf("\nНайближчий до 1 елемент %10.2f\n",
                                                  min_d1);
}

// метод для створення нового масиву
public static double[] create_array() {
  Scanner scanner = new Scanner(System.in);
  int sign = -1;// -1^n при n = 1
  System.out.print("p = ");
  double p = scanner.nextDouble();
  System.out.print("q = ");
  double q = scanner.nextDouble();
  // створюємо новий масив
  double m[] = new double[SIZE];
  // запоанюємо масив елементами
  for (int i = 0; i < SIZE; i++)
  {
    m[i] = sign * Math.sqrt(((i + 1) * (i + 1) + p)*
                  (i + 1 + q * q));
    sign *= -1; // обчислення -1 ^ (k + 1)
  }
  // повкртаємо створений масив
  return m;
}
// метод пошуку найближчого до 1 елемента масиву
public static double min_dist_el(double m[]) {
  double min = m[0];// початкове наближення
```

```
    // його відстань до 1
    double dist = Math.abs(min - 1);
    for (int i = 0; i < SIZE; i++) {
      if (Math.abs(m[i] - 1) < dist) {
        // знайдено новий елемент найближчий до 1
        min = m[0];
        dist = Math.abs(min - 1);
      }
    }
    return min;
  }
  // метод для виводу масиву на екран
  public static  void print_array(double m[]) {
    for (int i = 0; i < SIZE; i++) {
      System.out.printf("%6.2f, ", m[i]);
    }
        System.out.println();
    }
}
```

Програма на Python.

В якості масивів в ***Python*** використовуються списки, що вже самі по собі є об'єктами. Більше того, динамічна типізація обумовлює передачу усіх параметрів функцій як об'єктів. Тому ідея створення масивів та заповнення їх елементами безпосередньо у підпрограмі виглядає більш природньо.

Інтерпретація коду та безпосереднє виконання інтерпретатором коду реалізації функції зобов'язує нас оголосити та описати підпрограми до основного коду з їх викликом.

Оголошення псевдоконстанти з розміром масивів має на меті, хіба що, контроль циклів зі створення масивів.

Код програми для розв'язування задачі 510 на мові ***Python***:

```
import math
# псевдоконстанта для розміру масивів
SIZE= 12
# функції оголошуються до основної програми
def create_array():
  sign = -1 # -1^n при n = 1
  p = float(input("p = "))
  q = float(input("q = "))
  m = []
  for i in range(SIZE):
    m.append( sign * math.sqrt(((i + 1) * (i + 1) + p)* \
            (i + 1 + q * q)) )
    sign *= -1 # обчислення -1 ^ (k + 1)
  return m

def print_array(m):
  for a in m:
    print("%7.2f, "% a, end='')
  print()

def min_dist_el(m):
  min = m[0] # початкове наближення та його
  dist = abs(min - 1) # відстань до 1
  for a in m: # цикл по елементах масиву
    if abs(a - 1) < dist :
      # знайдено новий елемент найближчий до 1
      min = m[i]
      dist = abs(min - 1)
  return min
```

```
# основна программа
A = create_array()
B = create_array()
# заповнення третього масиву C
SumA = 0
for el in A:
  SumA += el
avg = SumA / SIZE
C = [] # оголошуємо список для масиву C
# і додаємо в нього елементи
for i in range(SIZE):
  if B[i] < 0 :
    C.append(avg) # записуємо середнє
  else :
    C.append(B[i]) # беремо з масиву B

print('Масив A:')
print_array(A) # вивід елементів масиву A
min_d1 = min_dist_el(A) # пошук найближчого до 1
print("Найближче до 1 %10.2f\n" % min_d1)
print('Масив B:')
print_array(B) # вивід елементів масиву B
min_d1 = min_dist_el(B) # пошук найближчого до 1
print("Найближче до 1 %10.2f\n" % min_d1)
print('Масив C:')
print_array(C) # вивід елементів масиву C
min_d1 = min_dist_el(C) # пошук найближчого до 1
print("Найближче до 1 %10.2f\n" % min_d1)
```

Розділ 7. Комбіновані та файлові типи даних

Оголошення та опрацювання структурованих типів даних. Зберігання даних у файлах.

511. Текстовий файл містить код html-сторінки. Прочитати текст файла та вивести його на екран. Перевірити коректність html-коду з файла щодо запису тегів в «косих» дужках, тобто перевірити чи код містить однакову кількість символів “<” та “>”.

512. Текстовий файл містить код програми на Java. Прочитати текст файла та вивести його екран. Визначити, чи є в програмі цикли, якщо є підрахувати їх кількість (в програмах Java кількість операторів циклу буде відповідати сумарній кількості службових слів for та while).

513. Деякий текст збережено на диску в окремому файлі. Скласти програму для підрахунку кількості символів в цьому тексті. Вивести окремо кількість цифр у тексті та кількість букв латинського алфавіту.

514. У файлі на диску збережено текст деякої статті. Скласти програму, що дозволить користувачеві перевірити чи містить стаття вказане ним слово або фразу, а також підрахує скільки разів це слово (фраза) зустрічається в тексті.

515. Текстовий файл містить код вебсторінки. Вивести на екран усі гіперпосилання, що містить сторінка. Гіперпосилання задається тегом ***<a>*** та міститься в його атрибуті ***href*** після знаку дорівнює в подвійних лапках.

516. У текстовому файлі міститься код програми на Pascal. Вивести на екран код програми. Підрахувати довжину (у символах) коду програми та коментарів до програми. Вивести на екран відсоткове співвідношення довжини робочого коду програми до довжини тексту розміщених в програмі коментарів. В програмах на мові Pascal коментарі записуються у фігурних дужках.

517. Текст програми на Python збережено у файлі на диску. Вивести код програми на екран та порахувати загальну довжину (у символах) коментарів до програми. В програмах на мові Python коментарі починаються знаком “#” та завершуються переходом на нову стрічку.

518. Деякий текст збережено на диску в окремому файлі. Скласти програму для підрахунку кількості символів в цьому тексті з врахуванням розділювачів (пробілів, знаків табуляції, розділових знаків), та без них.

519. У текстовому файлі міститься код програми на C++. Вивести на екран лише розміщені в програмі коментарі. В програмах на мові C++ коментарі записують у рядку після знаку подвійний «слеш» (“//”) або між знаками “/*” та “*/”.

520. Текстовий файл містить код вебсторінки. Вивести на екран адреси усіх малюнків, які містить сторінка. Адреса малюнку задається у тегу ***<img>*** в його атрибуті ***src*** після знаку дорівнює в подвійних лапках.

521. У текстовому файлі міститься код програми на Pascal. Вивести код програми на екран, вилучивши з нього усі коментарі. В програмах на мові Pascal коментарі записуються у фігурних дужках.

522. Текст програми на Python збережено у файлі на диску. Вивести на екран лише текст коментарів до програми та порахувати їх загальну довжину у символах. В програмах на мові Python коментарі починаються знаком "#" та завершуються переходом на нову стрічку.

523. Деякий текст збережено на диску в окремому файлі. Скласти програму для підрахунку кількості розділових знаків (крапка, кома, тире, дефіс і т.п.) та пробілів в цьому тексті.

524. У текстовому файлі міститься код програми на Java. Вивести на екран лише код програми без коментарів та «закоментованого» коду, враховуючи, що в програмах на Java коментарі записують у рядку після знаку подвійний «слеш» ("//") або між знаками "/*" та "*/". Підрахувати довжину (у символах) коментарів до програми та їх відсоткове співвідношення до загальної довжини тексту програми.

525. У файлі, збереженому на диску ПК міститься послідовність «стрічок», кожна з яких завершується символом закінчення абзацу. Вивести на екран найдовшу та найкоротшу «стрічки».

526. Текстовий файл містить код програми на Pascal. Прочитати текст файла та вивести його екран. Визначити, чи коректно в програмі утворено складені оператори, тобто, чи кількість службових слів ***begin*** співпадає з кількістю службових слів ***end***.

527. У текстовому файлі міститься код програми на Pascal. Вивести на екран лише розміщені в програмі коментарі. В програмах на мові Pascal коментарі записуються у фігурних дужках.

528. У текстовому файлі міститься код програми на C++. Вивести код програми на екран, вилучивши з нього усі коментарі. В програмах на мові C++ коментарі записують у рядку після знаку подвійний «слеш» ("//") або між знаками "/*" та "*/".

529. Деякий текст збережено на диску в окремому файлі. У файлі міститься як текст, так і числові дані. «Цілим числом» вважатимемо послідовність, утворену лише з цифр та обмежену пробілами з двох боків. Скласти програму для підрахунку кількості «цілих чисел» у тексті файлу.

530. Текстовий файл містить код вебсторінки. Вивести на екран усі заголовки, які містяться на сторінці. Заголовки в html-коді записуються між відкриваючим та закриваючим тегами ***<h1>, <h2>, <h3>, <h4>, <h5>, <h6>***.

531. Деякий текст збережено на диску в окремому файлі. Скласти програму для підрахунку кількості «слів» в цьому тексті. Вважати «словом», довільну послідовність символів що закінчується пробілом або знаком табуляції, окрім останнього «слова» абзацу.

532. Текстовий файл містить код програми на Pascal. Визначити та вивести на екран усі типи даних, що використані в програмі. В Pascal змінні описують у розділі опису змінних ***var***, кожен опис закінчується двокрапкою з вказанням типу змінних.

533. Текст програми на Python збережено у файлі на диску. Вивести на екран лише робочий код без коментарів до програми. В програмах на мові Python коментарі починаються знаком "#" та завершуються переходом на нову стрічку.

534. Текст, що збережено на диску в окремому файлі, розглядатимемо як послідовність «стрічок», кожна з яких завершується символом закінчення абзацу. Скласти програму для підрахунку кількості «стрічок» та визначення кількості символів у найдовшій та найкоротшій «стрічках».

535. Текст програми на Python збережено у файлі на диску. Підрахувати та вивести на екран наступну статистику: загальна довжина коду (у символах), кількість стрічок коду, кількість інструкцій (команд) програми, кількість стрічок з коментарями (та «закоментованого» коду). В програмах на мові Python кожна команда починається з нової стрічки, дозволяється розміщати декілька команд в одній стрічці, розділивши їх знаком ";", коментарі починаються знаком "#" та завершуються переходом на нову стрічку.

536. Текстовий файл містить код програми на Pascal. Визначити скільки змінних використовує програма. В Pascal усі змінні описують у розділі опису змінних ***var***, кожен опис закінчується двокрапкою з вказанням типу змінних, змінні в списку розділяють комами.

537. Деякий текст збережено на диску в окремому файлі. Скласти програму для підрахунку кількості речень в цьому тексті. Вважаємо, що речення, це послідовність символів, що закінчується символами ".", "!", "?", за якими йде пробіл або знак завершення абзацу.

538. У текстовому файлі міститься код програми на Java. Вивести на екран код програми. Підрахувати довжину (у символах) коду програми та коментарів до програми. Вивести на екран відсоткове співвідношення довжини робочого коду програми до довжини тексту розміщених в програмі коментарів. В програмах на мові Java коментарі записують у рядку після знаку подвійний «слеш» ("//") або між знаками "/*" та "*/".

539. Текстовий файл містить код html-сторінки. Прочитати текст файла та вивести на екран html-код сторінки без коментарів та «закоментованого» коду. Коментарі в мові html розміщають між "<! - -" та "- ->".

540. Файл містить код програми на Java. Вивести на екран лише текст коментарів до програми, враховуючи, що в програмах на мові Java коментарі записують у рядку після знаку подвійний «слеш» (“//”) або між “/*” та “*/”. Підрахувати довжину (у символах) коду програми без коментарів та вивести на екран відсоткове співвідношення довжини робочого коду програми до загальної кількості символів у файлі.

541. Створити програму для опрацювання файлу з даними про авіарейси. Структура даних: номер рейсу, пункт призначення, час відправлення, ціна квитка, кількість вільних місць. В програмі передбачити ввід даних користувачем та збереження їх у файл, а також вивід усіх даних з записаного раніше файлу.

542. Написати програму для опрацювання файлу з інформацією про складання екзаменаційної сесії студентами. Структура даних: шифр групи, прізвище студента, назва предмета, оцінка, назва предмета... (всього 5 предметів). В програмі передбачити ввід даних користувачем та збереження їх у файл, а також вивід не екран усіх даних з записаного раніше файлу.

543. У файл записують дані про продаж комп’ютерної техніки наступної структури: код покупця, назва товару, ціна за одиницю товару, продана кількість. Написати програму яка дозволить користувачеві створювати файл з даними, вносити в нього дані, а також виводити раніше збережені у файлі дані на екран (у вигляді таблиці).

544. Створити програму для роботи з файлом, що містить дані про рух пасажирських потягів від деякої станції. Запис містить поля: назва кінцевої станції, номер потяга, час відправлення, кількість наявних місць по категоріях вагонів – СВ, купейний, плацкартний, загальний. В програмі передбачити ввід даних користувачем та збереження їх у файл, а також вивід не екран усіх даних з записаного раніше файлу.

545. У файл записують дані про продаж турів клієнтам туристичної агенції наступної структури: прізвище та ім’я клієнта, назва туру та його тривалість, вартість туру та відмітка про оплату. Написати програму яка дозволить користувачеві створювати файл з даними, вносити в нього дані, а також виводити раніше збережені у файлі дані на екран (у вигляді таблиці).

546. Створити програму для опрацювання файлу з даними про оплату комунальних послуг через платіжну систему банку. Структура даних: прізвище платника, адреса платника, вид послуги, дата оплати, сума. В програмі передбачити ввід даних користувачем та збереження їх у файл, а також вивід усіх даних з записаного раніше файлу.

547. Створити програму для опрацювання файлу з інформацією про запис на прийом протягом поточного тижня відвідувачів поліклініки. Структура даних: прізвище відвідувача, домашня адреса, рік народження, прізвище сімейного лікаря, день тижня, потрібна спеціалізація лікаря. В програмі

передбачити ввід даних користувачем та збереження їх у файл, а також вивід у формі таблиці усіх даних з записаного раніше файлу.

548. У файл записують інформацію про вакансії, подані IT-компаніями, яка містить наступні дані: назва компанії, назва позиції, мова/технологія програмування, вимоги до претендента, орієнтовна заробітна плата. Розробити програму для створення файла з даними, внесення в нього даних, а також виводу усіх даних з файла на екран (у вигляді таблиці).

549. Ріелтор зберігає у файлі інформацію з оголошень про продаж нерухомості у вигляді записів такої структури: адреса помешкання, кількість кімнат, житлова площа, загальна площа, поверх, ціна, телефон власника. Реалізувати програму, яка дозволить ріелторові створювати новий файл з даними, вносити в нього дані, а також виводити раніше збережені у файлі дані на екран.

550. Створити програму для роботи з файлом, що містить дані про продаж вживаних автомобілів. Запис містить поля: марка авто, модель авто, тип кузова, об’єм двигуна, рік випуску, ціна. В програмі передбачити ввід даних користувачем та збереження їх у файл, а також вивід на екран усіх даних з записаного раніше файлу.

551. Рекрутер записує у файлі інформацію щодо резюме, розміщених IT-фахівцями у сервісах пошуку роботи. Записи містять інформацію такої структури: прізвище та ім’я, електронна адреса, місто проживання, перелік технологій (текст, в якому перераховані технології відділяються комами), досвід роботи, бажання працювати віддалено, можливість зміни міста проживання. Реалізувати програму для створення файла з даними, внесення в нього даних, а також виводу усієї інформації з файла на екран.

552. Менеджер паркінгу реєструє у файлі інформацію про парковку авто та оплату послуг їх власниками. Записи містять поля: марка та модель авто, державний номерний знак, телефон власника, година парковки (ціле число) та кількість годин, за яку вноситься передоплата. Скласти програму для вводу даних та збереження їх у файл, а також для виводу на екран усіх даних із записаного раніше файлу.

553. Створити програму для опрацювання файлу з даними про міжміські автобусні рейси. Запис містить поля: назва міста призначення, час відправлення, вартість квитка, загальна кількість місць в автобусі, кількість проданих квитків. В програмі передбачити ввід даних користувачем та збереження їх у файл, а також вивід усіх даних з записаного раніше файлу.

554. Менеджер онлайн-магазину зберігає у файлі дані про замовлення, що надходять. Структура даних: повна назва товару, ціна, кількість, прізвище та ім’я замовника, населений пункт для доставки, спосіб доставки, здійснення оплати. Написати програму для автоматизації роботи менеджера. В програмі передбачити ввід даних менеджером та збереження їх у файлі, а також вивід усіх даних з записаного раніше файлу.

555. Створити програму для опрацювання файлу з даними про наявність книг у книжковому фонді бібліотеки. Запис містить поля: шифр книги, назва, прізвище автора, рік видання, кількість примірників. В програмі передбачити ввід даних користувачем та збереження їх у файл, а також вивід усіх даних з записаного раніше файлу.

556. Дані про доступні для перегляду на онлайн-сервісі фільми зберігаються у файлі у вигляді записів такої структури: назва фільму, жанр, рік виходу на екрани, назва студії, прізвище режисера, загальна тривалість та кількість серій. Розробити програму для створення файлів з даними про фільми, запису даних у файл та для виводу збережених у файлі даних на екран.

557. Файл містить дані про дні народження рідних та друзів. Кожен запис містить поля: ім’я, прізвище, номер мобільного, число, місяць та рік народження. Створити програму для роботи з файлом, що дозволить записувати у нього дані а також виводити збережені у файлі дані на екран.

558. Файл містить дані про виконання робіт працівниками компанії протягом деякого періоду часу. Кожен запис містить поля: відділ, прізвище та ініціали, посада (позиція), кваліфікація, кількість відпрацьованих годин, оплата за годину. Створити програму для роботи з файлом, що дозволить записувати у нього дані а також виводити збережені у файлі дані на екран.

559. Файл містить дані про курси валют комерційних банків. Кожен запис містить поля: назва банку, назва валюти, курс купівлі, курс продажу. Створити програму для роботи з файлом, що дозволить записувати у нього дані, а також виводити збережені у файлі дані на екран.

560. Створити програму для опрацювання файлу з даними про тематичні ресурси мережі Internet (сайти). Запис містить поля: назва сайту, URL головної сторінки, тематика сайту, короткий опис, приблизна кількість сторінок на сайті, середня кількість відвідувачів за добу, рік заснування. В програмі передбачити ввід даних та збереження їх у файл, а також вивід усіх даних з записаного раніше файлу.

561. У файл записують дані про продаж побутової техніки наступної структури: назва товару, виробник, модель, ціна за одиницю товару, кількість одиниць на складі. Розробити програму, яка дозволить користувачеві створювати файл з даними, вносити в нього дані, а також виводити раніше збережені у файлі дані на екран (у вигляді таблиці).

562. Ріелтор зберігає у файлі інформацію з оголошень про оренду нерухомості у вигляді записів такої структури: адреса помешкання, кількість кімнат, житлова площа, поверх, вартість оренди за місяць, чи включена у вартість оренди оплата комунальних послуг, телефон власника. Розробити програму, яка дозволить ріелторові створювати новий файл з даними, вносити в нього дані, а також виводити раніше збережені у файлі дані на екран.

563. Страхова компанія записує у файл дані про продані страхові поліси. Дані мають таку структуру: прізвище та ім’я клієнта, адреса, номер мобільного

телефона, вид страхування, вартість полісу, дата початку страхового періоду, термін дії полісу. Розробити програму, яка дозволить менеджерам компанії створювати файли з даними, вносити в них дані, а також виводити раніше збережені у файлі дані на екран (у вигляді таблиці).

564. Створити програму для опрацювання файлу з інформацією реєстрацію відвідин, протягом поточного тижня, клієнтів салону краси. Структура даних: прізвище відвідувача, домашня адреса, рік народження, день тижня, вид послуги, наявність абонементу. В програмі передбачити ввід даних користувачем та збереження їх у файлі, а також вивід у формі таблиці усіх даних з записаного раніше файлу.

565. Створити програму для опрацювання файлу з даними для оплати електропостачання мешканцями міста. Структура даних: прізвище платника, номер лічильника, № квартири, № будинку, назва вулиці, попереднє значення лічильника, поточне значення лічильника, вартість 1 кВт спожитої електроенергії. В програмі передбачити ввід даних та збереження їх у файл, а також вивід усіх даних з записаного раніше файлу.

566. У файлі зберігають інформацію про тематичні публікації в мережі Internet. Кожен запис містить: назву сайту, URL публікації, назву публікації, прізвище та ім’я автора, дату публікації, тематику, короткий опис(анотацію). Написати програму для формування файлу з даними. В програмі передбачити ввід даних та збереження їх у файл, а також вивід усіх даних з записаного раніше файлу.

567. У файл записують інформацію про вакансії, подані різними компаніями у сервіси пошуку роботи. Інформація містить наступні дані: назва компанії, професія, посада, вид зайнятості (постійна робота, тимчасова, часткова зайнятість), вимоги до претендента, пропонована зарплатня. Розробити програму для створення файла з даними, внесення в нього даних, а також виводу усіх даних з файла на екран (у вигляді таблиці).

568. У файлі зберігають дані про виконання робіт підрядниками компанії. Кожен запис містить поля: назва компанії-підрядника, юридична адреса, номер банківського рахунку, вид робіт, обсяг виконаної роботи, вартість робіт за одиницю вимірювання. Створити програму для роботи з файлом, що дозволить записувати у нього дані а також виводити збережені у файлі дані на екран.

569. Створити програму для роботи з файлом, що містить дані про прокат автомобілів. Запис містить поля: марка авто, модель авто, тип кузова, коробка передач, кількість місць, об’єм багажника, об’єм двигуна, рік випуску, вартість доби оренди. В програмі передбачити ввід даних користувачем та збереження їх у файл, а також вивід на екран усіх даних з записаного раніше файлу.

570. Створити програму для опрацювання файлу з даними про наявність книг у книжковому магазині. Запис містить поля: назва видавництва, назва книги, прізвище автора, рік видання, роздрібна ціна, кількість примірників на

складі. В програмі передбачити ввід даних користувачем та збереження їх у файл, а також вивід усіх даних з записаного раніше файлу.

571. Розробити програму для пошуку даних у файлі, що містить інформацію про авіарейси. Структура даних: номер рейсу, пункт призначення, час відправлення, ціна квитка, кількість вільних місць. В програмі передбачити вивід у вигляді таблиці інформації про всі рейси до заданого користувачем пункту призначення та усі рейси, що відправляються у заданий користувачем проміжок часу.

572. Створити програму для пошуку даних у файлі, що містить дані про складання екзаменаційної сесії студентами. Структура даних: шифр групи, прізвище студента, назва предмета, оцінка, назва предмета... (всього 5 предметів). В програмі передбачити вивід списку студентів, які отримали “5” з усіх предметів, а також рейтингу (середньої оцінки з усіх предметів) в упорядкованому за спаданням вигляді.

573. Файл містить дані про продаж комп’ютерної техніки наступної структури: код покупця, назва товару, ціна за одиницю товару, продана кількість. Розробити програму, яка дозволить користувачеві за введеним з клавіатури кодом покупця виводити список куплених ним товарів та загальну вартість покупок, а за введеною назвою товару, – коди покупців, що його придбали та кількість проданих одиниць.

574. Створити програму для пошуку даних у файлі з інформацією про рух пасажирських потягів від деякої станції. Структура даних: назва кінцевої станції, номер потяга, час відправлення, кількість наявних місць по категоріях вагонів – СВ, купейний, плацкартний, загальний. Програма повинна виводити список потягів за вказаною назвою кінцевої станції та проміжком часу відправлення, а також відбирати лише ті з них, які мають вільні місця вказаної категорії.

575. Файл містить інформацію про продаж турів клієнтам туристичної агенції. Структура даних: прізвище та ім’я клієнта, назва туру та його тривалість, вартість туру та відмітка про оплату. Розробити програму яка дозволить користувачеві за введеною з клавіатури назвою туру знайти усіх клієнтів, що придбали цей тур та порахувати їх кількість та загальну суму виручки від продажі турів, а також вивести список усіх клієнтів, що замовили вказаний тур, але не внесли кошти на його оплату.

576. Створити програму для пошуку даних у файлі, що містить інформацію про оплату комунальних послуг у банку наступної структури: прізвище платника, адреса платника, вид послуги, дата оплати, сума. В програмі передбачити вивід даних про всі оплати послуг за вказаною адресою а також вивід підсумкової таблиці з сумарною оплатою кожного виду послуг.

577. Створити програму для пошуку даних у файлі, що містить дані про запис на прийом, протягом поточного тижня, відвідувачів поліклініки. Структура даних: прізвище відвідувача, домашня адреса, рік народження,

прізвище сімейного лікаря, день тижня, потрібна спеціалізація лікаря. В програмі передбачити вивід списку відвідувачів та підрахунок їх кількості для кожної спеціалізації на вказаний день тижня, а також вивід списку відвідувачів та їх кількості на кожен день тижня для заданої спеціалізації лікаря.

578. Файл містить інформацію про вакансії, подані IT-компаніями, яка містить наступні дані: назва компанії, назва позиції, мова/технологія програмування, вимоги до претендента, орієнтовна заробітна плата. Розробити програму, яка дозволить користувачеві відбирати за файла вакансії за введеною з клавіатури назвою мови програмування та діапазоном для розміру орієнтовної зарплатні, а також переглядати список усіх вакансій, поданих вказаною компанією.

579. Файл містить дані про продаж нерухомості у вигляді записів такої структури: адреса помешкання, кількість кімнат, житлова площа, загальна площа, поверх, ціна, телефон власника. Скласти програму яка дозволить користувачеві шукати у файлі та виводити на екран інформацію за такими параметрами: за введеною з клавіатури кількістю кімнат та діапазоном цін, або за кількістю кімнат та орієнтовною загальною чи житловою площі помешкання.

580. Скласти програму для пошуку даних у файлі з інформацією про продаж вживаних автомобілів. Запис містить поля: марка авто, модель авто, тип кузова, об’єм двигуна, рік випуску, ціна. Програма повинна виводити список авто за такими вказаними користувачем параметрами: марка, модель та діапазон цін або марка, модель та період випуску (у роках, від… до…).

581. Рекрутер зберігає у файлі інформацію щодо резюме, розміщених IT-фахівцями у сервісах пошуку роботи. Записи містять інформацію такої структури: прізвище та ім’я, електронна адреса, місто проживання, перелік технологій (текст, в якому перераховані технології відділені комами), досвід роботи, бажання працювати віддалено, можливість зміни міста проживання. Розробити програму яка дозволить йому відбирати для офісу із заданого міста резюме, що відповідають вказаній технології, розділивши їх на три частини: претенденти, що проживають у заданому місті, претенденти, готові до віддаленої роботи та претенденти, що погодяться переїхати у вказане місто.

582. Менеджер паркінгу реєструє у файлі інформацію про парковку авто та оплату послуг їх власниками. Записи містять поля: марка та модель авто, державний номерний знак, телефон власника, година парковки (ціле число) та кількість годин, за яку вноситься передоплата. Скласти програму для пошуку даних у файлі, яка дозволить менеджеру, вказавши поточний час (годину) дізнатися список авто, власникам яких доведеться доплатити за парковку та за скільки годин, а також дізнатися, чи авто може без доплати покинути паркінг, вказавши його номер.

583. Створити програму для пошуку даних у файлі, що містить інформацію про міжміські автобусні рейси. Запис містить поля: назва міста призначення, час відправлення, вартість квитка, загальна кількість місць в автобусі, кількість проданих квитків. Запрограмувати визначення заповненості (у %) рейсів до вказаного міста чи обчислення сумарної виручки для кожного рейсу (за вибором користувача програми) та вивід результатів у вигляді таблиці.

584. Менеджер онлайн-магазину опрацьовує замовлення, дані про які зберігаються у файлі. Структура даних: повна назва товару, ціна, кількість, прізвище та ім’я замовника, населений пункт для доставки, спосіб доставки, здійснення оплати. Запрограмувати вивід з файлу усіх оплачених замовлень, які слід доставити у зазначене місто, та вивід усіх неоплачених замовлень за заданим способом доставки товару.

585. Створити програму для пошуку книг у бібліотеці, дані якої зберігаються у файлі, що містить записи такої структури: шифр книги, назва, прізвище автора, рік видання, кількість примірників. Запрограмувати вивід списку усіх книг, які є в бібліотеці в єдиному примірнику, а також пошук книг за прізвищем автора чи назвою (або її частиною).

586. Дані про доступні для перегляду на онлайн-сервісі фільми зберігаються у файлі у вигляді записів такої структури: назва фільму, жанр, рік виходу на екрани, назва студії (кінокомпанії), прізвище режисера, загальна тривалість та кількість серій. Скласти програму для пошуку фільмів у файлі. Програма повинна за вказаним жанром виводити на екран два списки окремо одно-, а окремо багатосерійні фільми цього жанру, а також за вказаною назвою кінокомпанії та діапазоном років виводити інформацію про фільми, зняті компанією у цей період.

587. Файл містить дані про дні народження рідних та друзів. Кожен запис містить поля: ім’я, прізвище, номер мобільного, число, місяць та рік народження. Створити програму для пошуку даних у файлі. Програма повинна знаходити дані про дату народження за вказаними прізвищем та ім’ям, а також виводити список даних про дні народження у вказаному місяці.

588. Файл містить дані про виконання робіт працівниками компанії протягом деякого періоду часу. Кожен запис містить поля: відділ, прізвище та ініціали, посада (позиція), кваліфікація, кількість відпрацьованих годин, оплата за годину. Створити програму для формування відомостей для оплати праці на основі даних з файла. Додатково реалізувати пошук у файлі даних за вказаними посадою та кваліфікацією працівника.

589. Файл містить дані про курси валют комерційних банків. Кожен запис містить поля: назва банку, назва валюти, курс купівлі, курс продажу. Створити програму, що дозволить користувачеві формувати на екрані таблицю курсів валют для вказаного ним банку, а також шукати банк з найвищим курсом покупки та банк з найнижчим курсом продажу вказаної користувачем валюти.

590. Створити програму для пошуку даних у файлі, що містить описи тематичних ресурсів мережі Internet (сайтів). Запис містить поля: назва сайту, URL головної сторінки, тематика сайту, короткий опис, приблизна кількість сторінок на сайті, середня кількість відвідувачів за добу, рік заснування. Запрограмувати вивід списку сайтів заданої тематики та пошук і вивід даних про сайти, опис яких містить заданий користувачем текст.

591. Файл містить дані про продаж побутової техніки наступної структури: назва товару, виробник, модель, ціна за одиницю товару, кількість одиниць на складі. Розробити програму яка дозволить менеджерові за введеною з клавіатури назвою товару виводити список пропозицій з вказанням виробників, моделей та ціни, а також шукати у файлі товари, кількість яких на складі менша за вказане число.

592. Файл містить дані про оренду нерухомості у вигляді записів такої структури: адреса помешкання, кількість кімнат, житлова площа, поверх, вартість оренди за місяць, чи включена у вартість оренди оплата комунальних послуг, телефон власника. Скласти програму яка дозволить користувачеві за введеною з клавіатури кількістю кімнат та вартістю оренди (діапазон від-до) вивести на екран два списки: помешкання з заданими параметрами, у вартість оренди яких включено комунальні послуги, та помешкання, в яких комунальні платежі доведеться сплачувати додатково.

593. Страхова компанія зберігає у файлі дані про продані страхові поліси. Дані мають таку структуру: прізвище та ім’я клієнта, адреса, номер мобільного телефона, вид страхування, вартість полісу, дата початку страхового періоду, термін дії полісу. Написати програму, яка дозволить менеджерам компанії виводити підсумкову таблицю загальної вартості полісів за різними видами страхування, а також знаходити усі поліси клієнта за його прізвищем чи номером телефону.

594. Створити програму для пошуку даних у файлі, що містить дані про реєстрацію відвідин, протягом поточного тижня, клієнтів салону краси. Структура даних: прізвище відвідувача, домашня адреса, рік народження, день тижня, вид послуги, наявність абонементу. В програмі передбачити вивід списку відвідувачів та підрахунок їх кількості для кожного виду послуги на вказаний день тижня, а також вивід для вказаного виду послуги списку відвідувачів та їх кількості на кожен день тижня.

595. Створити програму для пошуку даних у файлі, що містить інформацію для оплати електропостачання мешканцями міста. Структура даних: прізвище платника, номер лічильника, № квартири, № будинку, назва вулиці, попереднє значення лічильника, поточне значення лічильника, вартість 1 кВт спожитої електроенергії. В програмі передбачити вивід за вказаним прізвищем платника, або його адресою (вулиця, будинок, квартира) а також формування «платіжок» для мешканців заданого будинку чи вулиці.

596. У файлі записана інформація про тематичні публікації в мережі Internet. Кожен запис містить: назву сайту, URL публікації, назву публікації, прізвище та ім’я автора, дату публікації, тематику, короткий опис(анотацію). Написати програму для виводу переліку усіх публікацій вказаного автора, а також пошуку публікацій, опис яких містить заданий користувачем текст.

597. Файл містить інформацію про вакансії, подані різними компаніями у сервіси пошуку роботи. Інформація містить наступні дані: назва компанії, професія, посада, вид зайнятості (постійна робота, тимчасова, часткова зайнятість), вимоги до претендента, пропонована зарплатня. Реалізувати програму, яка дозволить користувачеві відбирати з файла вакансії за введеною з клавіатури назвою посади та мінімальним значенням орієнтовної зарплатні, а також переглядати список усіх вакансій за вказаним видом зайнятості.

598. Файл містить дані про виконання робіт підрядниками компанії. Кожен запис містить поля: назва компанії-підрядника, юридична адреса, номер банківського рахунку, вид робіт, обсяг виконаної роботи, вартість робіт за одиницю вимірювання. Створити програму для формування підсумкового звіту для фінансових розрахунків з підрядниками. В програмі передбачити також пошук відомостей виконавця вказаного виду робіт.

599. Скласти програму для пошуку даних у файлі з інформацією про прокат автомобілів. Запис містить поля: марка авто, модель авто, тип кузова, коробка передач, кількість місць, об’єм багажника, об’єм двигуна, рік випуску, вартість доби оренди. Програма повинна шукати авто за заданими клієнтом технічними параметрами (тип кузова, кількість місць, об’єм багажника, к/п, тощо) або за маркою, роком випуску та орієнтовною ціною добової оренди.

600. Створити програму для пошуку книг у книжковій крамниці, дані якої зберігаються у файлі, що містить записи такої структури: назва видавництва, назва книги, прізвище автора, рік видання, роздрібна ціна, кількість примірників на складі. Запрограмувати пошук книг за прізвищем автора чи назвою (або її частиною), а також вивід списку усіх книг, які вийшли у вказаному видавництві.

Практикум до розділу 7. Комбіновані та файлові типи даних

Терміном ***файл*** зазвичай позначають іменовану послідовність байтів збережену в запам'ятовуючому пристрої, але в програмуванні ***файл*** – це, насамперед, певна структура даних, в якій дані записуються послідовно і, взагалі кажучи, в необмеженій кількості. Так само послідовно ці дані програма може зчитати. Файли можуть зберігатися в запам'ятовуючих пристроях або передавати дані між технічними та програмними компонентами за допомогою інших засобів. Наприклад, в программах на ***Pascal*** базова система вводу-виводу утворена з файлів ***`input`*** та ***`output`***, що є потоками символів від клавіатури та даних, що виводяться на монітор, відповідно.

Часто файли ділять на два типи: ***файли послідовного доступу*** та ***файли довільного доступу***. Такий поділ є умовним, – фізично усі файли є лише послідовностями байтів, об'єднаних спільною назвою.

До файлів ***послідовного доступу*** традиційно зараховують текстові файли. Вони зберігають послідовність байтів, кожен з яких є кодом деякого символа, бо читання-запис даних в таких файлах можна здійснювати посимвольно. Основною перевагою текстових файлів є універсальність формату зберігання інформації – числові дані в символьному вигляді доступні для будь-якої програми на будь-якому комп'ютері, а за необхідності їх може прочитати навіть людина. Задачі пов'язані з обробкою таких файлів, зазвичай мають безпосередній стосунок до опрацювання текстів, де послідовність сисмволів має певне змістове тлумаченння, як наприклад в задачах 511 - 540 цього розділу.

Якщо дані записують у файл та читають з файла фіксованим порціями, що за розміром дорівнює розміру деякого визначеного набору змінних різних типів, то говорять про ***довільний доступ*** всередині файлу. Оскільки можна змістити вказівник зчитування з файлу на задану кількість таких порцій, то можна прочитати екземпляр згаданого набору за його порядковим номером в послідовності. Такі файли деколи ще називають структурованими. Читання окремих байтів структурованого файлу, взагалі кажучи, позбавлене інформаційного контексту, але може виконуватися, як виняток, наприклад, якщо програма виконує копіювання файлів, переписуючи послідовність байтів з одного місця диску в інше.

Шаблон набору даних за допомогою якого структуруються послідовність байтів у файлах довільного доступу, в загальному випадку, прийнято називати ***комбінованим типом***. В алголоподібних мовах, вт. ч. Pascal, до нього використовують термін ***запис***, в C та C++ – ***структура***, при переході до об'єктно-орієнтованих технологій стало актуальним зберігання даних в ***об'єктах***, тобто, в екземплярах ***класів***.

540. Файл містить код програми на Java. Вивести на екран лише текст коментарів до програми, враховуючи, що в програмах на мові Java коментарі записують у рядку після знаку подвійний «слеш» (“//”) або між “/*” та “*/”. Підрахувати довжину (у символах) коду програми без коментарів та вивести на екран відсоткове співвідношення довжини робочого коду програми до загальної кількості символів у файлі.

Розв’язування задач 510 – 540 полягає в наступному: програма повинна відркити заданий файл з текстом, прочитати з нього послідовно увесь текст, шукаючи в ньому вказані в умові елементи і, за потреби, виводячи їх на екран. Паралелльно з виводом коментарів програма повиння порахувати кількості виведених та пропущених символів.

Окрім стандартних дій для читання текстового файлу в нашому алгоритмі буде присутня логіка аналізу прочитаного тексту щодо присутності в ньому java-коментарів. Тут проаналізуємо окремо два випадки:

- ✓ *стрічковий коментар може починатися в будь-якому місці стрічки тексту парою символів “//”, тоді увесь текст, починаючи з цих символів і до завершення стрічки є коментарем, його потрібно виокремити зі стрічки і вивести на екран;*
- ✓ *блочний коментар. Тут потрібно перевірити стрічку на наявність символів “/*”, якщо їх знайдено, то далі слід шукати “*/” і тут маємо ще дві ситуації: “/*” та “*/” записані в одній стрічці, тоді слід видобути, та вивести частину стрічки між “/*” та “*/”. Якщо ж стрічка з “/*” не містить “*/”, то текст стрічки слід вивести цілком, починаючи зі знайденого знака початку коментаря. Далі слід послідовно вивести усі наступні стрічки тексту, що не містять знака закінчення коментаря аж поки він не знайдеться. Зі стрічки, в якій виявили знак закінчення коментаря (“*/”) слід вибрати та вивести лише текст, розташований від початку стрічки до цього знака.*

З огляду на сказане вище, в нашій програмі доведеться використати елементарний функціонал для опрацювання текстових стрічок, який реалізований у кожній з розглядуваних в практикумах до розділів технологій програмування. Отож, алгоритм розв’язування задачі буде таким:

- ➢ *на початку роботи програм відкриває вказаний текстовий файл та починає читати його стрічка за стрічкою;*
- ➢ *кожна прочитана з файлу стрічка тексту перевіряється на наявність знаку “//”, якщо він в стрічці знаходиться, то зі стрічки вибирається увесь текст після “//” і програма виводить його на екран;*
- ➢ *кожна прочитана з файлу стрічка тексту перевіряється на наявність знаку “/*”, якщо він в стрічці знаходиться, то програма переходить до пошуку “*/”, якщо ні, то вона читає наступну стрічку тексту з файлу;*
- ➢ *якщо стрічка, що містить знак “/*”, містить також і “*/”, то програма вибирає текст між цими знаками і виводить його на екран та переходить до аналізу наступної стрічки;*

- *якщо стрічка, що містить знак "/*", не містить "*/", то програма вибирає текст від "/*" до кінця стрічки, виводить його екран, потім читає наступні стрічки з файлу і виводить їх на екран, доки не прочитає стрічку, яка містить "*/". З останньої стрічки на екран виводиться лише текст до з "*/", далі програма переходить читання та перевірки наступних стрічок тексту;*
- *читаючи чергову стрічку з файлу програма повинна порахувати кількість символів в ній та додати це число до загальної кількості сисволів файлу;*
- *виводячи черговий фрагмент коментаря, програма має визначити кількість символів в ньому та додати її до загальної довжини коментарів у файлі;*
- *після того, як усі стрічки тексту з файлу прочитані та проаналізовані, програма визначає та виводить довжину тексту файла без коментарів та її відношення до загальної кількості символів у файлі, закриває файл, та завершує свою роботу.*

Реалізація цього алгоритму в конкретній технологій програмування залежатиме від специфіки засобів читання текстових файлів та опрацювання стрічок тексту, реалізованих в цій технології.

Програма на Pascal.

Мова програмування ***Pascal*** для опрацювання тестових файлів використовує спеціальний тип даних, – ***text***. Читання тексту з файлу не дуже відрізняється від зчитування введеного користувачем тексту з консолі (читання з консолі в ***Pascal*** також відбувається через службовий текстовий файл ***output***). Для читання використовують ті ж процедури ***read*** та ***readln***, але у випадку зовнішнього файла першим параметром їм передають дескриптор цього файлу.

Щоб за допомогою оголошеної змінної типу ***text*** взаємодіяти з фізичним файлом на диску, їх слід зв'язати інструкцією вигляду:

```
assign(<назва файлової змінної>, <назва фізичного файла>);
```

Для опрацювання тестових стрічок ***Pascal*** використовує вбудований тип ***string***. Дані цього типу можна трактувати як масиви, в яких окремий символ доступний за порядковим номером в стрічці (індексом) та опрацьовувати по-символьно з використанням циклів. Проте, для даних цього типу ***Pascal*** має набір стандартних підпрограм, в нашому коді ми використовуємо функції:

```
function trim ( const line : string ) : string;
```

яка повертає переданий їй рядок вилучивши з нього зайві пробіли на початку та вкінці;

```
function pos ( const needle, line : string ) : integer;
```

яка визначає позицію перше входження тексту ***needle*** в стрічку ***line*** (якщо вказаний текст не знайдено, – функція повертає -1);

```
function copy (source : string; pos, n : integer): string;
```

що створює копію частини стрічки, починаючи з символа з номером ***pos***, довжиною ***n*** символів.

```
program Task540;
Uses sysutils; { підключення модуля з деякими
    системними підпрограмами, зокрема, функцією trim }
var comment, line, tmp: string;
    javacode: text;
    c_len, f_len, p_len: integer; comment_koef: real;
    start, stop, len  : integer;
begin
  assign(javacode, 'Task360.java');
  reset(javacode);
  c_len := 0; {кількість символів в коментарях}
  f_len := 0; {загальна довжина тексту у файлі}
  while not eof(javacode) do
  begin
  {читаємо чергову стрічку тексту}
    readln(javacode, tmp);
  {читаємо код програми без зайвих
    пробілів на початку та в кінці стрічки}
    line := trim(tmp);
    {підраховуємо довжину тексту у файлі}
    f_len := f_len + length(line);
  {Вивід однострічкових коментарів}
    start := pos('//',line);
    len := length(line);
    if start > 0 then {гілка виконується, }
    begin {якщо в стрічці знайдено "//"}
```

```
    comment := copy(line, start, len - start + 1);
    writeln(comment);
    {підраховуємо загальну довжину коментарів}
    c_len  := c_len + length(comment);
  end;
{Вивід багатосрічкових та блочних коментарів}
  start := pos('/*',line);
  stop := pos('*/',line);
  len := length(line);
  if start > 0 then{ вітка розгалуження}
  begin{виконується лише, якщо в стрічці }
    {знайдено символ початку блоку коментаря }
    if stop > 0 then {якщо коментар закінчується}
    begin {в тій же стрічці}
      comment := copy(line, start, stop-start + 3);
      writeln(comment);
      c_len  := c_len + length(comment);
    end
    else
    {коментар багаторядковий, потрібно читати
    наступні рядки з файла доки не зустрінеться
    знак завершення коментаря }
    begin
      {перша стрічка знайденого коментаря}
      comment := copy(line, start, len-start + 1);
      repeat
          writeln(comment);
          c_len  := c_len + length(comment);
          readln(javacode, tmp);
          line := trim(tmp);
```

```
                {врахуємо довжину чергової стрічки з файлу
                 до зальної довжтини тексту у файлі}
                f_len := f_len + length(line);
                comment := line;
                stop := pos('*/',line);
            until stop > 0; {вихід з циклу, якщо
            знайдено ознаку закінчення коментаря
            виводимо частину стрічки до закінчення коментар}
            comment := copy(line, 1, stop + 2);
            writeln(comment);
            c_len  := c_len + length(comment);
          end
        end
    end;
    p_len := f_len - c_len;
    writeln('Code without comment has ', p_len, ' symbols');
    comment_koef := 100 * p_len / f_len;
    writeln('Working code is ', comment_koef:6:2,
                  '% of the total text character');
    close(javacode);
end.
```

Програма на C++.

Мова програмування ***C++*** дає нам широкий спектр інструментів для реалізації описаного вище алгоритму. По-перше, ***C++*** підтримує дві парадигми опрацювання файлів: процедурну з використанням класичних функцій мови ***C*** та об'єктно-орієнтовану з використанням файлових потоків. По-друге, для опрацювання текстових стрічок тут можна використовувати масиви символів або відповідний бібліотечний клас ***string***.

Зупинимося на використанні ***файлових потоків***, їх функціонал дуже схожий до функціоналу їхніх близьких «родичів», ***cin*** та ***cout***, які ми використовували в більшості прикладів для вводу даних з клавіатури, та

виводу тексту на екран. Для потокового читання тексту з файла нам потрібен екземпляр класу ***ifstream***, оголошеного у заголовковому файлі ***fstream***.

Для опрацювання тексту, прочитаного з файла скористаємося масивами символів (тип ***char***), в яких закінчення стрічки тексту позначається спеціальним символом ***'\0'*** (його прийнято називати «***нуль-байтом***»). Такими масивами можна користуватися як в програмах на ***C***, так і в програмах на ***C***++. Для їх опрацювання використовується набір функцій з заголовкового файлу ***string.h***. В нових стандартах ***C***++ для цього ж заголовкового файла використовується назва ***cstring***.

В нашому коді зі ***string.h*** використовуються функції:

для визначення кількості символів в рядку (враховуються лише символи зліва від ***нуль-байта***, розмір самого масиву може бути більшим);
```
size_t strlen( const char* str );
```

для пошуку стрічки ***substr*** в стрічці ***str***:
```
char* strstr( const char* str, const char* substr );
```

для копіювання стрічки , або її частини в іншу стрічку:
```
char* strcpy( char* dest, const char* src );
```

для копіювання заданої кількості символів стрічки в іншу стрічку:
```
char* strncpy( char* dest, const char* src, size_t count);
```

Дві останні функції мають відповідники для «захищеного режиму» роботи з пам'яттю ***strcpy_s*** та ***strncpy_s***, саме їх ми використовуємо в коді нижче, хоча на роботу алгоритму це жодним чином не впливає. Тип ***size_t*** насправді є іншою назвою типу ***int***, тут він використовується для задання чи визначення кількості символів.

Усі зазначені функції працюють з ***вказівниками***, наприклад, пошук підстрічки повертає не сам номер символа з якого починається входження шуканого тексту, а адресу пам'яті, де цей символ розташовано. Аналогічно, функція копіювання тексту отримує не текст і номер символа, з якого слід почати копіювання, а адресу символа, з якого слід почати копіювання, у пам'яті. Це зручно для роботи програми і суттєво підвищує її швидкодію, але вимагає від розробника розуміння фундаментальних концепцій використання ***вказівників***. Тип даних ***char**** власне задає вказівник на символьну змінну, що, враховуючи специфіку реалізації масивів в ***C***++, дозволяє працювати з масивами символів. Тип ***const shar**** дає можливість передавати функціям в якості параметра не тільки змінні відповідного типу, але й фрагменти тексту, записані в подвійних лапках (***стрічкові літерали***).

Програма-розв'язок задачі 540 мовою ***C++***:

```
#include <iostream>
#include <fstream>
#include <string.h>
//Прогнозована максимальна довжина
//стрічки тексту
#define MAX_LINE 512
using namespace std;

int main()
{
  char file_name[] = "E:/JavaSolutions/Task360.java";
  //Відкриваємо файловий потік для читання
  ifstream myfile(file_name);
  //Оголошуємо змінні для підрахунку кількостей символів
  int total_len = 0, commented_text_len = 0;
  if (myfile.is_open())
  {
    while (!myfile.eof())
    {//цикл читання стрічок до закінчення файлу
      //масиви символів для зберігання стрічок
      char tmp[MAX_LINE] = { 0 };
      char line[MAX_LINE] = { 0 }, comment[MAX_LINE] =
{ 0 };
      myfile.getline(tmp, MAX_LINE);
      int n;//копіюємо прочитану стрічку без
      // зайвих пробілів на початку
      for (n = 0; isspace(tmp[n]); n++);
      int len = strlen(tmp) - n;//довжина line
      // без початкових пробілів
      strncpy_s(line, tmp + n, len);
```

```
// враховуємо довжину прочитаної стрічки в
// загальну кількість символів у файлі
total_len += len;
//Вивід однострічкових коментарів
char* start = strstr(line, "//");
if (start != NULL)
{
  strcpy_s(comment, start);
  cout << comment << endl;
  // підраховуємо кількість символів
  // у виведеному коментарі
  commented_text_len += strlen(comment);

}
//Вивід багатосрічкових та блочних коментарів
start = strstr(line, "/*");
if (start != NULL)
{
  char* end = strstr(line, "*/");
  if (end != NULL)//коментар в одній стрічці
  {
    int comment_len = end - start + 2;
    strncpy_s(comment, start, comment_len);
    cout << comment << endl;
    // підраховуємо кількість символів
    // у виведеному коментарі
    commented_text_len += strlen(comment);

  }
  else
```

```
{
  strcpy_s(comment, start);
  cout << comment << endl;
  // підраховуємо кількість символів
  // у виведеному коментарі
  commented_text_len += strlen(comment);
  do
  {//читаємо наступні стрічки коментаря доки
   //не зустрінемо символів його закінчення
    myfile.getline(tmp, MAX_LINE);
    int n;
    for (n = 0; isspace(tmp[n]); n++);
    int len = strlen(tmp) - n;//довжина line
    // без початкових пробілів
    strncpy_s(line, tmp + n, len);
    end = strstr(line, "*/");
    // враховуємо довжину прочитаної стрічки в
    // загальну кількість символів у файлі
    total_len += len;
    if (end != NULL)
    {
      int comment_len = end - line + 2;
      //вибираємо з рядка лише коментар
      strncpy_s(comment, line, comment_len);
    }
    else
    {
      strcpy_s(comment, line);
    }
    cout << comment << endl;
```

```
            // підраховуємо кількість символів
            // у виведеному коментарі
            commented_text_len += strlen(comment);
          } while (end == NULL);
        }
      }
    }
    myfile.close();
  }
  else cout << "Unable to open file";
  int code_len = total_len - commented_text_len;
  // запис числа 100 у вигляді дісного літерала 100.0
  // вказує компілятору, що ділення кількостей тут
  // слід виконувати за правилом ділення дісних чисел
  double koef = (100.0 * code_len) / total_len;

  cout << "Code without comments length: " << code_len <<
endl;
  cout << "Working code percentage: " << koef << " %" <<
endl;

  return 0;
}
```

Програма на C#.

Об’єктно-орієнтовані технології програмування ***Java*** та ***C#*** для опрацювання фрагментів тексту використовують спеціальний бібліотечний клас ***String***. Він забезпечений власними методами для визначення довжини стрічки, пошуку в ній тексту, копіювання частини тексту, тощо. В ***C#*** цьому класові відповідає тип даних ***string***. Кількість символів в тексті об’єкта ***string*** контролюється властивістю ***Length***, а для утворення нового об’єкта-стрічки з частини заданої використовують метод ***Substring***. Для пошуку вказаної послідовності символів в об’єкті ***string*** використовують методи ***Contains*** та ***IndexOf***, – перший повертає булеве значення і лише дає відповідь на питання чи присутня вказана послідовність символів у стрічці, а другий дозволяє визначити порядковий номер початкового символа входження (першого зліва) вказаної підстрічки у заданий текст. На відміну від аналогічних функцій ***C/C++***, що використовують вказівники, методи класу ***String*** оперують індексами, тобто порядковим номерами, символів. Слід не забувати, що індекс першого символа стрічки дорівнює 0, а 1.

```
using System;
using System.IO;
using System.Text;

namespace Task540
{
  class Program
  {
    static void Main(string[] args)
    {
      string FilePath = "E:/JavaSolutions/Task360.java";
      using (FileStream stream =
            new FileStream(FilePath, FileMode.Open))
      {
        using (StreamReader reader =
            new StreamReader(stream,
            Encoding.UTF8))
        {
          int commentLen = 0;
          int totalLen = 0;
```

```
while(!reader.EndOfStream)
{
  string line = reader.ReadLine();
  line = line.Trim();
  //Підраховуємо загальну кількість символів
  totalLen += line.Length;
  //Вивід однострічкових коментарів
  if (line.Contains("//"))
  {
    int position = line.IndexOf("//");
    int endPositoin = line.Length;
    string comment = line.Substring(
          position, endPositoin - position);
    Console.WriteLine(comment);
    //Підраховуємо довжину коментарів
    commentLen += comment.Length;
  }
  //Вивід багатосрічкових та блочних
  //коментарів
  if (line.Contains("/*"))
  {
    int position = line.IndexOf("/*");
    int endPositoin = line.IndexOf("*/")>0?
          line.IndexOf("*/") + 2:line.Length;
    string comment = line.Substring(
           position, endPositoin - position);
    Console.WriteLine(comment);
    //Підраховуємо довжину коментарів
    commentLen += comment.Length;
    while (!line.Contains("*/"))
    {
      line = reader.ReadLine();
      line = line.Trim();
```

```
                    //Підраховуємо загальну кількість символів
                       totalLen += line.Length;
                       if (line.Contains("*/"))
                       {
                         int length = line.IndexOf("*/");
                         //вибираємо з рядка лише коментар
                         comment = line.Substring(0, length+2);
                       }
                       else
                       {
                         comment = line;
                       }
                       Console.WriteLine(comment);
                       //Підраховуємо довжину коментарів
                       commentLen += comment.Length;
                     }
                   }
                 }
                 int codeSymbols= totalLen - commentLen;
                 Console.WriteLine( $"Довжина коду без " +
                       "коментарів {codeSymbols} символів");
                 double codePercentage =
                           (100.0 * codeSymbols) / totalLen;
                 Console.WriteLine($"він становить " +
                         "{codePercentage} % тексту файла");
               }
             }
           }
         }
       }
```

Програма на Python.

В реалізації нашого алгоритму на ***Python*** скористаємося з можливості ***Python-а*** завантажувати увесь текст з файла у вигляді списку стрічок з подальшим опрацюванням елементів цього списку (окремих стрічок тексту) у циклі. Це спростить нам опрацювання тексту файла, але тоді програма, знайшовши початок багатострічкового коментаря, не зможе зчитати наступні його стрічки в новому циклі. Тому скористаємося додатковою булевою змінною, що передаватиме ознаку продовження багатострічкового коментара на наступні ітерації загального циклу перегляду стрічок файла.

Для опрацювання стрічок тексту ***Python*** також має потужний інструментарій, – починаючи від функції ***len(<текст>)*** для визначення довжини стрічки та оператора ***in***, що перевіряє наявність заданого фрагмента у стрічці і закінчуючи широким набором методів, до яких належить метод ***find()*** для пошуку позиції заданої підстрічки в тексті. Для копіювання частин тексту скористаємося так званим «усіченням списку».

Код програми для розв'язування задачі 540 на мові ***Python***:

```
with open('E:/JavaSolutions/Task360.java','r',256,'UTF-8')
as f:
  lines = f.readlines()
  # в булевій змінній multiline будемо фіксувати
  # ситуації коли коментар поширюється
  # на наступну стрічку
  multiline = False
  f_len = 0 # загальна кількість символів у файлі
  c_len = 0 # кількість символів в коментарях
  # Переглядаємо в циклі усі стрічки тексту файла
  for line in lines:
    # читаємо чергову стрічку тексту в змінну line
    # без зайвих пробілів на початку та в кінці стрічки
    code_line = line.strip()
    f_len += len(code_line) # загальна кількість символів
    # Вивід однострічкових коментарів
    if '//' in code_line :
      start = code_line.find('//')
      comment = code_line[start:]
```

```
  print(comment)
  c_len += len(comment)
# Вивід багатосрічкових та блочних коментарів
# якщо стрічка є продовженням багатострічкового
# комантаря, то шукаємо його закінчення, або
# виводимо стрічку повністю
if multiline:
  if '*/' in code_line:
  # якщо знайдено ознаку закінчення коментаря
  # виводимо частину стрічки до закінчення коментаря
    end = code_line.find('*/')
    comment = code_line[:end+2]
    print(comment)
    c_len += len(comment)
    multiline = False
  else:
    comment = code_line
    print(comment)
    c_len += len(comment)
else:
  # стрічка не є продовженням багатострічкового
  # коментаря, тоді виконуємо пошук символів
  # початку коментаря
  if '/*' in code_line:
    start = code_line.find('/*')
    tmp = code_line[start:]
    if '*/' in code_line:
      end = tmp.find('*/')
      comment = tmp[:end+2]
      print(comment)
```

```
            c_len += len(comment)
        else:
            comment = tmp
            print(comment)
            c_len += len(comment)
            multiline = True
# кількість пробілів повинна відповідати
# рівню на якому відбулося зчитування стрічок
# файла та створення f_len і c_len
p_len = f_len - c_len # кількість символів
# робочого коду у файлі
print('Довжина робочого коду ', p_len, ' символів')
percent = p_len / f_len * 100
# для виводу познасення відсотків використовуємо %%
print('це становить %6.2f %% від '%percent, end='')
print( 'загальної довжини тексту')
```

570. Створити програму для опрацювання файлу з даними про наявність книг у книжковому магазині. Запис містить поля: назва видавництва, назва книги, прізвище автора, рік видання, роздрібна ціна, кількість примірників на складі. В програмі передбачити ввід даних користувачем та збереження їх у файл, а також вивід усіх даних з записаного раніше файлу.

В підручниках з інформатики деколи можна зустріти думку, що програми для роботи зі структурованими файлами ілюструють елементарні прийоми роботи з найпростішими базами даних. Така думка не позбавлена рації, – одним реченням умову цієї задачі можна було б сформулювати так: розробити програму для керування базою даних про книги у книжковому магазині. Якщо говорити про реляційні БД, то ***комбінований тип даних*** (запис) в такому контексті якраз описуватиме структуру таблиці з даними про книги.

Програма повинна надавати користувачеві такий функціонал:

- *зберігати інфомацію про перелік книг у магазині у формі даних вказаної структури;*
- *створювати новий файл (базу даних) та заповнювати його даними проо книги;*

- *записувати інформацію про окрему книгу в існуючий, створений раніше файл, з даними;*
- *виводити на екран перелік книг, що зберігаються у файлі (базі даних) та дані про них.*

Не будемо обтяжувати програмний код не надто важливим функціоналом для вибору файла з даними, а реалізуємо роботу програми з одним і тим же файлом, ім'я якого запишемо безпосередньо в коді програми. Для зручного структурування коду організуємо програму у вигляді набору підпрограм, кожна з яких вирішує окреме задання: створення файла і внесення даних, вивід усього переліку книг, чи додавння даних про нову книгу в існуючий перелік. В головній програмі, при такому підході, можна реалізувати «вічний» цикл в ході якого користувач обиратиме потрібну йому дію та переглядатиме її результат.

Для коректної роботи програми слід описати відповідний комбінований тип даних (***запис*** про окрему книгу), що поєднуватиме в собі різні базові типи даних під спільною назвою та дозволятиме оперувати цим набором даних як одним цілим.

Програма на Pascal.

Комбінований тип даних в ***Pascal*** називають ***записом*** та оголошують за допомогою службового слова record. Всередині оголошення вказують назви змінних, що будуть доступні, як частини запису (***поля запису***) та відповідний тип даних за схемою, аналогічною до оглошення змінних програми. Оголошення запису закінчується службовим словом ***end***, його можна застосовувати до позначення типу будь-якої змінної, але зручніше задавати у вигляді іменованого типу даних, як у коді нижче.

Для роботи зі структурованими файлами в Pascal використовується тип ***file of <base type>***, в нашому прикладі ***file of book***. Такі файли зберігають дані в двійковому форматі і дозволяють зчитування їх цілими записами.

Для початку роботи з типізованими файлами потрібно виконати зв'язування змінної з фізичним файлом на диску, за допомогою інструкції вигляду:

```
assign(<file variable>, <file name>);
```

Далі файл потрібно відкрити для читання:

```
reset(<file variable>);
```

або для запису:

```
rewrite(<file variable>);
```

Режим запису дозволяє створювати нові файли і записувати в них дані, при записі в існуючий файл, дані, що були в ньому раніше, видаляються. В ***Pascal*** можна відкривати файли і для дозапису (***append***), проте в такому режимі працювати можна лише з текстовими файлами. Тому в коді нижче, додавання

нового запису в існуючий файл реалізовано шляхом переписування усіх даних в тимчасовий файл разом з новим записом. Пізніше оновлений список записів перезаписується назад в основний файл.

Записують дані у типізовані файли процедурою:

```
write(<file variable>, <data record>);
```

зчитують процедурою:

```
read(<file variable>, <data record>);
```

По завершенні роботи з даними файл потрібно закрити інструкцією:

```
close(<file variable>);
```

Код Pascal-програми для розв'язування задачі 570.

```
program Task570;

uses sysutils;

{Оголошуємо тип даних для опису однієї книги}

type book = record

    publisher : string[50];

    book_title : string[255];

    author : string[100];

    publish_year : integer;

    price : real;

    count : integer

    end;

const db_name : string = 'books_shop.db';

var tmp : book;

    database : file of book;

    choise : integer;

procedure readall(filename: string);

    var data: book;

begin

    assign(database, filename);

    reset(database);

    while eof(database)=false do

    begin
```

```
        read(database,data);
        writeln(data.author,'  ',data.book_title,
        ' ', data.publisher,' ',data.publish_year);
    end
end;
function inputbook() : book;
    var result : book;
begin
    writeln('Введіть про нову книгу');
    write('Автор(автори):'); readln(result.author);
    write('Назва:'); readln(result.book_title);
    write('Видавництво:'); readln(result.publisher);
    write('Рік видання:'); readln(result.publish_year);
    write('Ціна:'); readln(result.price);
    write('Кількість:'); readln(result.count);
    inputbook := result;
end;
procedure writenewfile(filename: string);
begin
    assign(database, filename);
    rewrite(database);
    repeat
    tmp := inputbook();
    write(database, tmp);
    writeln('Додати ще одну книгу?');
    writeln('1 - так, додати ще;');
    writeln('0 - зберегти дані і вийти');
    readln(choise);
    until(choise<>1);
    writeln('нову базу книг стоврено');
```

```
    close(database);
end;

procedure writebook(filename: string; data: book);
const tmp_filename = 'database.dump';
var dump_file: file of book;
begin
    assign(database, filename);
    reset(database);
    assign(dump_file, tmp_filename);
    rewrite(dump_file);
    while eof(database)=false do
    begin
        read(database,tmp);
        write(dump_file,tmp)
    end;

    write(dump_file, data);
    close(dump_file);
    rewrite(database);
    reset(dump_file);
    while eof(dump_file)=false do
    begin
        read(dump_file,tmp);
        write(database,tmp)
    end;
    writeln('книга додана в базу');
    close(database);
    close(dump_file);
    deletefile(tmp_filename)
```

```
end;


begin
  writeln('Оберіть потрібну дію:');
  writeln('1 - створити/перестворити базу книг.');
  writeln('2 - дописати нову книгу в базу');
  writeln('3 - переглядnути всі книги в базі');
  readln(choise);
  case (choise) of
    1:writenewfile(db_name);
    2:writebook(db_name, inputbook());
    3:readall(db_name);
  end;
end.
```

Програма на C++.

Комбінований тип даних мови ***C***, має назву ***структура*** (**`struct`**), він дозволяє об'єднувати різнотипні дані в одне ціле, та отримувати доступ частин об'єднаних даних за оголошеними в описі структури іменами. Структури можна використовуати для оголошення змінних та констант в якості нових типів даних.

В ***C***++ структури отримали можливість зберігати окрім даних ще й функції для обробки цих даних, ставши засобом для реалізації об'єктно-орієнтованої парадигми на рівні з класами, але, в контексті цієї задачі нас цікавитиме лише класичний підхід (як в ***C***).

Оголошення структури в якості нового типу даних в ***C***++ позначають ключовим словом **`struct`**, після якого вказують назву цього типу. Далі, у фігурних дужках описують поля структури, що будуть доступні, як частини даних. Поля оголошують за правилами, аналогічними до оглошення змінних чи параметрів функцій:

```
<тип даних> <назва поля>;
```

Оголошення структури завершують знаком "**`;`**" після закриваючої фігурної дужки. Оголошувати структуру, як тип даних можна поза межами функцій, тоді цей тип є глобальним і його можна використовувати для оголошення змінних та констант в коді будь-якої з функцій. Оголошення нового

комбінованого типу може бути і в коді функції, тоді цей тип можна використовувати лише в цій функцій (локально).

Для зберігання структур в ***C++*** використовують бінарні файли, хоча насправді від текстових вони відрізняються лише режимом читання-запису: текстові файли читаються посимвольно, бінарні побайтово. «Типізованим» двійковий файл робить сам спосіб запису даних та їх зчитування, – при записі структури у файл, туди поміщають копію двійкових даних з пам'яті, яку відведено для зберігання структурної змінної. Для зчитування структури з файла достатньо скопіювати відповідну кількість байтів із файла та помістити в пам'ять, яку займає змінна-структура відповідного типу. При такому підході слід чітко розподіляти дані на «порції», що відповідають розміру структури (його можна визначити оператором ***sizeof(<тип>)***). Зрозуміло, що при цьому тип змінної, яка зчитується з файлу повинен співпадати з типом структури, яка була туди записана, включно з послідовнітю опису її полів.

Для опрацювання файлів ***C++*** використовує файлові потоки, але можна користуватися й функціями для роботи з файлами, розробленими в C. В наших прикладах ми застосуємо другий спосіб, він простіший, та краще ілюструє загальний підхід.

В класичному процедурному підході ***C/C++*** для роботи з файлами використовується спеціальна структура ***FILE***, в яку при відкритті фізичного файла записуються його атрибути. Отримати програмний доступ до файла можна за допомогою функції:

```
fopen_s(&<FILE_pointer>, <file_name>, <mode>);
```

Для відкриття файла передається не сама змінна типу ***FILE*** а вказівник на цю змінну ***FILE_pointer***, тобто адреса її розташування у пам'яті. З метою безпечної роботи з пам'ятю функція ***fopen_s*** приймає першим параметром адресу цього вказівника на файлову змінну, – її можна отримати за допомогою оператора "***&***".

В якості імені файла ***file_name*** використовується рядкова константа, тобто текст у подвійних лапках. Якщо вказати назву самого файла, як у прикладі нижче, – тоді він буде розташований у папці з програмою. Можна також використовувати повне ім'я файла, включно з міткою логічного диску та послідовністю вкладених папок, що веде до його розташування у файловій системі.

Режим ***mode*** також задається рядковою константою у подвійних лапках, він може містити від одного до трьох симолів. Перший вказує напрям роботи з даними: "***r***" – читання даних, "***w***" – запис даних (якщо файл фізично не існує, то його буде створено), "***a***" – дозапис даних у кінець існуючого файлу. Другим символом в режимі відкриття може бути позначка бінарного читання (запису) "***b***", наприклад, як у коді нижче: "***rb***". Якщо позначка "***b***" відсутня, то з даними файлу працюють посимвольно. Тут двійковий режим "***ab***" дозволяє

нам дозаписувати новий екземпляр структури в кінець існуючого файла з даними, проте, для цілісності викладу ми скористаємося перезаписування даних в тимчасовий файл, так само, як і у прикладі на ***Pascal*** вище.

Для запису екземпляра структури у файл використовуємо функцію:

```
fwrite(&<buff>, <size>, <count>, <file_pointer>);
```

Вона записує ***count*** порцій байтів розміру ***size*** із змінної ***buff*** у файл заданий вказівником на файлову змінну ***file_pointer***. Знак "***&***" перед змінною ***buff*** вказує, що функції передається не сама змінна а її адреса (вказівник). Для зворотнього процесу, тобто для зчитування байтів з файлу в структуру, застосовують функцію:

```
fread(&<buff>, <size>, <count>, <file_pointer>);
```

з аналогічним списком прпаметрів. У разі успішного читання даних функція fread повертає кількість прочитаних порцій.

Програма-розв'язок задачі 570 мовою C++:

```
#include <iostream>

#include <fstream>

#include "Windows.h"

using namespace std;

//Оголошуємо тип даних для опису однієї книги

struct book

{

  char publisher[50];

  char book_title[255];

  char author[100];

  int publish_year;

  double price;

  int count;

};

//назва файлу з даними

const char db_name[] = "books_shop.db";

//прототип функції для читання усіх книг

void readall(const char *);

//прототип функції для вводу даних про окрему книгу
```

```
book inputbook();
//функція для створення нового файлу з книгами
void writenewfile(const char* filename);
//прототип функції для дозапису нової книги у файл
void writebook(const char*, book);

//головна функція (програма)
int main()
{
  SetConsoleCP(1251); SetConsoleOutputCP(1251);
  int choise;
  do {
    system("cls");
    cout << "Оберіть потрібну дію : \n";
    cout << "1 - створити/перестворити базу книг. \n";
    cout << "2 - дописати нову книгу в базу \n";
    cout << "3 - переглянути всі книги в базі \n";
    cout << "0 - завершити роботу програми \n";
    cin >> choise; cin.get();
    switch (choise) {
    case 1:
      writenewfile(db_name);
      break;
    case 2:
      writebook(db_name, inputbook());
      break;
    case 3:
      readall(db_name);
      break;
    }
```

```
  } while (choise != 0);

  return 0;
}

void readall(const char* filename)
{
  book data;
  FILE* database;
  fopen_s(&database, filename, "rb");
  while (true)
  {
    int count = fread(&data, sizeof(data), 1, database);
    if (count == 0) break;
    cout << data.author << " " << data.book_title
      << " " << data.publisher << " "
        << data.publish_year << endl;
  }
  system("pause");
}

book inputbook()
{
  book result;
  cout << "Введіть інформацію про нову книгу\n";
  cout << "Автор(автори):";
  cin.getline(result.author, 100);
/* функція getline використовується для читання з консолі
    тексту влючно з пробілами та розділовими знаками,
    результат читання поміщається в змінну, передану першим
```

```
    параметром, а другий її параметр обмежує максимальну
    кількість символів, яку можна прочитати*/
  cout << "Назва:";
  cin.getline(result.book_title, 255);
  cout << "Видавництво:";
  cin.getline(result.publisher, 50);
  cout << "Рік видання:"; cin >> result.publish_year;
  cout << "Ціна:"; cin >> result.price;
  cout << "Кількість:"; cin >> result.count;
  cin.get();//зчитує символ закінчення рядка
  //який залишається після вводу числових даних
  //і може бути сприйнятий програмою як текст
  //при наступному вводі текстових даних
  return result;
}

void writenewfile(const char* filename)
{
  FILE* database;
  int choise;
  book tmp;
  fopen_s(&database, filename, "wb");
  do {
    tmp = inputbook();
    fwrite(&tmp, sizeof(tmp), 1, database);
    cout << "Додати ще одну книгу ? \n";
    cout << "1 - так, додати ще;\n";
    cout << "0 - зберегти дані і вийти.\n";
    cin >> choise; cin.get();
  }while(choise == 1);
```

```
  cout << "нову базу книг створено"<< endl;
  fclose(database);
}

void writebook(const char* filename, book data)
{
  const char tmp_filename[] = "database.dump";
  FILE *dump_file, *database;
  book tmp;
  fopen_s(&database, filename, "rb");
  fopen_s(&dump_file, tmp_filename, "wb");
  //перезаписуємо існуючі дані в тимчасовий файл
  while (true)
  {
    int count = fread(&tmp, sizeof(tmp), 1, database);
     if (count == 0) break;
    fwrite(&tmp, sizeof(tmp), 1, dump_file);
  }
  //дописуємо нову книгу в тимчасовий файл
  fwrite(&data, sizeof(data), 1, dump_file);
  fclose(database);
  remove(filename);//видаляємо файл, щоб при
  // наступоному перезаписі створити новий
  fclose(dump_file);// закриваючм файл
  // в який записуємо дані, підтверджуємо
  // для файлової системи збереження
  // усіх записаних даних на диску
  fopen_s(&database, filename, "wb");
  fopen_s(&dump_file, tmp_filename, "rb");
  //відкриваємо файли для перезапису усіх
```

```
  //даних з тимчасового файлу в основний
  while (true)
  {
    int count = fread(&tmp, sizeof(tmp), 1, dump_file);
    if (count == 0)break;
    fwrite(&tmp, sizeof(tmp), 1, database);
  }
  fclose(database);
  fclose(dump_file);
  cout << "книга додана в базу" << endl;
  remove(tmp_filename);
}
```

Програма на C#.

Об'єктно-орієнтовані технології програмування в якості основного засобу для створення комбінованих типів даних використовують ***класи***. ***Клас*** в ООП це більше, ніж просте об'єднання з набору різнотипних даних, – класи підтримують ***спадкування***, ***поліморфізм***, ***узагальнення*** і інші об'єктно-орієнтовані механізми. Використання об'єктів класу в констектсі нашої задачі для елементарного зберігання комбінованих даних вимагатиме задіювання відповідних механізмів ООП. Зберігання бітової копії об'єкта з можливістю його подальшого відновлення в ООП відбувається шляхом ***серіалізації*** даних об'єкта в файл. ***Серіалізація***, зазвичай виконується методами потокових класів, або за допомогою спеціальних класів- серіалізаторів.

Об'єкти напростіших класів, що містять лише поля з елементарними даними легко піддаються ***серіалізації,*** але, взагалі кажучи, не всі об'єкти можна серіалізувати. Зворотній процес, тобто, створення об'єкта в програмі на основі збереженої бітової копії, називають ***десеріалізацією***. В кросплатформних технологіях, таких як ***Java*** та ***.Net (C#), серіалізація*** є безальтернативним способом зберігання комбінованих даних ще й тому, що програма не має прямого доступу до пам'яті, в якій зберігаються її дані, – створенням і розміщенням об'ктів в пам'яті ПК тут займається віртуальна машина.

На відміну від ***Java***, де для створення комбінованого типу даних доступні лише класи, розробники ***C#*** надали програмістам можливість використовувати структури, як в ***C++***. Хоча структури в ***C#*** мають ряд відмінностей від класів, вони не зовсім аналогічні до структур ***C++***, наприклад, для запису екземплярів структури у файл тут слід використовувати серіалізацію. В нашому прикладі ми скористаємося саме структурою а не класом.

Оголошення структури в ***C#*** (як і в ***C/C++***) позначають ключовим словом ***struct***, після якого вказують назву нового типу. Оголошення полів структури, на відміну від ***C++***, починається з модифікатора доступу ***public***, що вказує їх доступність для методів, реалізованих поза структурою. Без вказання модифікатора доступу поле структури вважається закритим (***private***), так само як і класу, і працювати з ним зможуть лише методи, реалізовані всередині цієї ж структури. Стосовно самої структури, як оголошеного типу даних, також діють обмеження в доступі і загальнодоступні ***типи*** (класи чи стрктури) в ***C#*** слід оголошувати з модифікатором ***public*** у заголовку. Проте дефолтний (без модифкатора) рівень доступу стосовно ***типу*** визначається як ***internal***, що робить їх доступними в межах проєкту, а цього нам цілком достатньо. Оголошення структури завершують знаком "***;***" після закриваючої фігурної дужки.

Для опрацювання даних у файлах в ***C#*** використовують файлові потоки та класи для роботи з ними із простору імен ***System.IO***. Наприклад, клас ***FileStream***, об'єкти якого використовуються в коді далі. Для створення файлових потоків та для роботи з файловою системою ПК зручно використовувати клас ***System.IO.File*** та його статичні методи. В нашому прикладі, зокрема, викороистовуємо метод ***File.OpenRead(<file name>)*** для отримання файлового потоку, з якого читаємо дані та метод ***File.Create(<file name>)*** файлового потоку, який записує дані у вказаний файл, створюючи (перестворюючи) його на диску.

Відкриті тим чи іншим способом файлові потоки після завершення роботи з ними слід закривати. Насамперед це стосується потоків, в які записують дані, – якщо потік не буде вчасно закритий, то дані можуть «загубитися» в процесі передачі між програмою та файловою системою. Крім цього, незакриті файли можуть бути проблемою і для самої файлової системи. Для закриття потоку відповідні об'єкти повинні викликати метод ***Close()***, але в роботі з файлами можуть виникати непередбачувані ситуації з аварійним завершенням програми, при цьому до виклику методу ***Close()*** програма може просто не дійти. В цій ситуації задіюється система опрацювання ***виняткових ситуацій*** (***exception handling***), але вона вимагає передбачення різних ситуацій в коді програми, що робить його доволі громіздким. Простіше вирішення цієї проблеми реалізоване у ***C#*** у вигляді наступної конструкції:

```
using (FileStream <змінна потоку> = <об'єкт потоку>)
{
    < інструкції з опрацювання даних у потоці>;
}
```

Вона забезпечує автоматичне закриття файлового потоку у будь-якому випадку, – як при кореткному, так і при аварійному завершенні програми.

Нашу програму оформимо у вигляді класу з набором статичних методів, як ми це робили для створення підпрограм на основі об'єктно-орієнтованих технологій. Для серіалізації-десеріалізації даних використаємо екземпляр спеціального класу ***BinaryFormatter*** з відповідними методами. Читання серіалізованої у файл послідовності окремих об'єктів буде проблемним, бо у файлі окрім даних зберігаються деякі параметри самого об'єкта, тому зберігатимемо у файлі тільки одини об'єкт, – список структур типу book (***List<book>***). При такому підході для дозапису в базу нової книги не потрібен допоміжний файл, його роль відіграватиме сам список у пам'яті програми. Решту особливостей серіалізації структур в ***C#*** пропонуємо розглянути в наступному коді:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;

namespace Task570
{
  // Структура для зберігання даних про книгу
  // Serializable атрибут вказує, що екземпляри структури
  // будуть серіалізуватися у файл і системі потрібно
  // перевірити, чи серіалізація цієї структури можлива
  [Serializable]
  struct book
  {
    public string publisher;
    public string book_title;
    public string author;
    public int publish_year;
    public double price;
    public int count;
  };
```

```
class Program
{
  //назва файлу з даними
  const string db_name = "books_shop.db";
  //головний метод (програма)
  static void Main(string[] args)
  {
    int choise;
    do
    {
      Console.Clear();
      Console.WriteLine("Оберіть потрібну дію :");
      Console.WriteLine("1 – створити базу книг;");
      Console.WriteLine("2 - дописати книгу в базу;");
      Console.WriteLine("3 - переглянути всі книги;");
      Console.WriteLine("0 - завершити роботу.");
      choise = Convert.ToInt32(Console.ReadLine());
      switch (choise)
      {
        case 1:
          WriteNewFile(db_name);   break;
        case 2:
          WriteBook(db_name, InputBook());   break;
        case 3:
          ReadAll(db_name);   break;
      }
    } while (choise != 0);
  }
  //статичний метод для вводу даних про окрему книгу
  private static book InputBook()
```

```
{    // Створюємо порожній екземпляр стьруктури
  book result = new book();
  // і заповняємо його введеними даними
  Console.Clear();
  Console.WriteLine("Введіть про нову книгу");
  Console.WriteLine("Автор(автори):");
  result.author = Console.ReadLine();
  Console.WriteLine("Назва:");
  result.book_title = Console.ReadLine();
  Console.WriteLine("Видавництво:");
  result.publisher = Console.ReadLine();
  Console.WriteLine("Рік видання:");
  result.publish_year = Convert
                          .ToInt32(Console.ReadLine());
  Console.WriteLine("Ціна:");
  result.price = Convert.ToDouble(Console.ReadLine());
  Console.WriteLine("Кількість:");
  result.count = Convert.ToInt32(Console.ReadLine());
  return result;
}
//статичний метод для читання усіх книг
private static void ReadAll(string filename)
{
  List<book> books = new List<book>();
  BinaryFormatter binaryFormatter =
    new BinaryFormatter();
  using (FileStream fileStream =
                              File.OpenRead(filename))
  {      // десеріалізуємо колекцію книг з файла та
    // приводимо її до типу список структур типу book
```

```
    books = (List<book>)binaryFormatter
                              .Deserialize(fileStream);
    for (int i = 0; i < books.Count; i++)
    {
      Console.WriteLine(books[i].author + " "
        + books[i].book_title
      + " " + books[i].publisher + " "
        + books[i].publish_year);
    }
    Console.ReadKey();
  }
}
//статичний метод для створення нового файлу з книгами
private static void WriteNewFile(string filename)
{     //Створюємо список в який записуватимемо книги
  List<book> books = new List<book>();
  int choise;
  book tmp;
  BinaryFormatter binaryFormatter =
    new BinaryFormatter();
  using (FileStream fileStream = File.Create(filename))
  {
    do
    {
      tmp = InputBook();
      Console.WriteLine("Додати ще одну книгу ?");
      Console.WriteLine("1 - так, додати ще;");
      Console.WriteLine("0 - зберегти дані і вийти.");
      choise = Convert.ToInt32(Console.ReadLine());
      books.Add(tmp);
```

```
        } while (choise == 1);
        // серіалізуємо об'єкт списку книг у файл
        binaryFormatter.Serialize(fileStream, books);
      }
      Console.WriteLine("нову базу книг стоврено");
    }
    //статичний метод для дозапису нової книги у файл
    private static void WriteBook(string filename,
                                          book data)
   {
     List<book> books = new List<book>();
     BinaryFormatter binaryFormatter=new BinaryFormatter();
     using (FileStream fileStream = File
                                .OpenRead(filename))
     {
       books = (List<book>)binaryFormatter
                           .Deserialize(fileStream);
     }   // відкриваємо файл для перезапису
     using (FileStream fileStream = File.Create(filename))
     {    // додаємо у прочитаний список книг нову книгу
       books.Add(data);
       // і серіалізуємо об'єкт списку книг у файл знову
       binaryFormatter.Serialize(fileStream, books);
     }
     Console.WriteLine("книга додана в базу");
   }
  }
}
```

Програма на Python.

В ***Python*** немає структури даних схожої за функціоналом до записів ***Pascal*** чи структур ***C++***. Ця технологія підтримує об'єктно-орієнтовану парадигму, відповідно, для реалізації комбінованого типу в контексті цієї задачі можна було б скористатися ***класом***. З іншого боку, серед базових типів ***Python*** є ***списки***, ***кортежі*** та ***словники***, які цілком можна було б «приладнати» до зберігання набору даних різного типу.

Кортежем, зокрема, можна скористатися, якщо пронумерувати поля запису і звертатися до їхніх даних за номером поля. Такий підхід не зовсім вкладається в загальну схему розв'язування нашої задачі, тут зручніше використовувати ***асоціативні масиви***, – в них кожен елемент доступний не за його номером, а за його назвою. Функціонал асоціативного масиву у ***Python*** надає тип даних **`dict`** (***словник***).

Словник утворюється зі списку елементів, кожен з яких утворений з пари елементів: ***ключ***, тобто назва даних, та ***значення***, тобто самі дані. Щоправда з використанням словників в контексті нашої задачі також треба бути обережним. Словник, на відміну від кортежу, дозволяє додавати нові елементи в уже заповнений асоціативний масив. Тобто в коді програми потрібно слідкувати, щоб до даних про одну книжку в екземплярі словника не дописувалися ще якісь властивості, наприклад, дані наступної книжки зі списку. Але в нашому прикладі ми зупинимося саме на використанні словників.

Оголошення та створення словника в ***Python*** доволі просте:

```
<змінна> = {'<ключ 1>' : <значення 1>, '<ключ 2>' :
            <значення 2>, ..., '<ключ n>' : <значення 2>}
```

Для кращого контролю за полями запису винесемо код вводу даних про книжку в окрему функцію і вводитимемо дані лише через цю функцію.

Можливість змінювати словники є потенційним місцем проблем для запису-читання даних в двійковому форматі. Насправді, дані комбінованих типів не завжди зберігають в такому вигляді. До прикладу, в тій же програмі на C# ми мали можливість замість бінарного серіалізатора скористатися класом який серіалізує дані структур в текстовий формат xml (***eXtensible Markup Language***), що успішно використовується для передачі та зберігання структурованих даних в мережі ***Internet***. Тут ми скористаємося з можливостей модуля csv, який дозволяє Python-програмам зберігати структуровані дані в текстових файлах у спепціальному форматі csv (***Comma-Separated Values***). Такий формат зручний своєю універсальністю, до прикладу, один з діалектів csv читається процесорами електронних таблиць MS Excel та OOo Calc і його можна перетворити на електронну таблицю.

Код програми для розв’язування задачі 570 на мові ***Python***:

```
import csv
db_name = 'books_shop.csv'
# функція для вводу даних про окрему книжку
# формує і повертає словник з даними
def input_book():
    book_title = input('Назва книги: ')
    authors = input('Автори: ')
    publisher = input('Видавництво: ')
    publish_year = int(input('Рік видання: '))
    price = float(input('Ціна: '))
    count = int(input('Кількість: '))
    book = {'book_title' : book_title, 'authors': authors,
            'publisher' : publisher, 'publish_year':
            publish_year, 'price' : price, 'count' : count}
    return book

# функція для створення\перестворення бази
# ввід даних про книжки і запис їх у файл
def write_new_file(filename):
    books = [] # список книг для запису у файл
    again = 1
    with open(filename, 'w', newline='',
                  encoding='cp1251') as file:
        columns = ['book_title', 'authors', 'publisher',
            'publish_year', 'price', 'count']
        writer = csv.DictWriter(file, fieldnames=columns)
        writer.writeheader()
        while again > 0:
            books.append(input_book())
            print('Додати ще одну книгу?')
```

```
            print('1 - так, додати ще;')
            print('0 - зберегти дані і вийти')
            again = int(input('>>>'))
        writer.writerows(books)
        print('нову базу книг створено')

# зчитування даних про книги з csv-файла
# та вивід їх на консоль
def read_all(filename):
    with open(filename, 'r', newline='',
              encoding='cp1251') as file:
        reader = csv.DictReader(file)
        for row in reader:
          print(row['authors'], ' ', row['book_title'],
                ', - ', row['publisher'], ': ',
                row['publish_year'],'. ', row['price'],
                ' грн., ', row['count'], ' в наявності.')

# дозапис даних про нову книгу
def write_book(filename, data):
    with open(filename, 'a', newline='',
              encoding='cp1251') as file:
        columns = ['book_title', 'authors', 'publisher',
            'publish_year', 'price', 'count']
        writer = csv.DictWriter(file, fieldnames=columns)
        writer.writerow(data)
```

```
# Основна програма
choise = 4 # ненульове значення для входу в цикл
# за умовою в меню 0 - вихід з циклу, і цикл не
# виконається жодного разу, якщо ініціалізувати цю
# змінну значенням 0, а оголошення та
# ініціалізація змінної choise потрібні до циклу
while choise != 0:
    # вивід меню програми
    print('Оберіть потрібну дію:')
    print('1 - створити/перестворити базу книг.')
    print('2 - дописати нову книгу в базу')
    print('3 - переглядnути всі книги в базі')
    print('0 – завершити роботу програми')

    # зчитування пункту меню обраного користувачем
    # та виконання відповідної дій з даними
    choise = int(input('>>>'))
    if choise == 1:
        write_new_file(db_name)
    elif choise == 2:
        write_book(db_name, input_book())
    elif choise == 3:
        read_all(db_name)
```

600. Створити програму для пошуку книг у книжковій крамниці, дані якої зберігаються у файлі, що містить записи такої структури: назва видавництва, назва книги, прізвище автора, рік видання, роздрібна ціна, кількість примірників на складі. Запрограмувати пошук книг за прізвищем автора чи назвою (або її частиною), а також вивід списку усіх книг, які вийшли у вказаному видавництві.

Задачі 571-600 є продовженням відповідних задач 541-570, тобто, кожна пара задач з різницею в 30 номерів, в тому числі задачі 570 та 600, є однією задачею, яку символічно можна назвати «програмування простої інформаційної системи для заданої предметної області». В першій частині цієї задачі реалізовуються інструменти для наповнення системи інформацією, в другій, – для пошуку потрібної користувачеві інформації.

Зрозуміло, що для програми до цієї задачі нам потрібен файл з даними, заповнений програмою для розв'язування задачі 570 на відповідній мові. При цьому слід прослідкувати, щоб структура записів в коді обидвох програм була ідентичною включно з типами даних її полів, а для серіалізації-десеріалізації повинні повністю співпадати назви класів з точністю до простору імен.

Стосовно функціоналу інформаційної системи, який слід реалізувати в цій частині, то її можна реалізувати на основі підпрограми для виводу усіх записів. Пошук інформації в контексті зчитування послідовності усіх даних з файла буде полягати у виводі лише тих записів, які відповідають заданому користувачем критерію.

Алгоритм роботи програми полягатиме в такій послідовності дій:

- *програма за допомогою меню запитує в користувача варіант пошуку даних, який потрібно виконати;*
- *користувач вводить число, що відповідає обраному пункту меню і програма переходить до виконання відповідної підпрограми;*
- *далі підпрограма запитує в користувача параметри пошуку;*
- *отримавши потрібні для пошуку дані підпрограма зчитує послідовно усі записи з файлу та виводить на екран лише ті, які відповідають запиту користувача.*

Тут, як і в попередній задачі, можна «загорнути» вивід меню та виконання обраної користувачем підпрограми у «вічний цикл», отримавши програму, що працює у режимі діаллогу з користувачем.

Програма на Pascal.

В ***Pascal***-реалізації слід точно продублювати оголошення структури ***book*** з прикладу до задачі з 570 (можна зробити копії відповідних фрагментів коду). Процедура ***readall*** залишена також з попередньої програми для того, щоб можна було переглянути усі записи з файла та переконатися, що програма правильно виводить результати пошуку.

```
program Task600;
uses sysutils;
{Оголошуємо тип даних для опису однієї книги}
type book = record
    publisher : string[50];
    book_title : string[255];
    author : string[100];
    publish_year : integer;
    price : real;
    count : integer
    end;
const db_name : string = 'books_shop.db';
var tmp : book;
    database : file of book;
    choise : integer;

procedure readall(filename: string);
    var data: book;
begin
    assign(database, filename);
    reset(database);

    while eof(database)=false do
    begin
        read(database,data);
        writeln(data.author,'  ',data.book_title,
        ' ', data.publisher,' ',data.publish_year);
    end
end;
```

```
procedure find_by_author(filename: string);
    var data: book;
    author : string[100]; {змінна для пошуку
    співпадає за типом з відповідним полем запису}
begin
    assign(database, filename);
    reset(database);
    write('Введіть прізвище автора:');
    readln(author);
    while eof(database)=false do
    begin
        read(database,data);
        if pos(author, data.author) > 0 then
            writeln(data.author,'  ',data.book_title,
            ' ', data.publisher,' ',data.publish_year);
    end
end;

procedure find_book(filename: string);
    var data: book;
    book_title : string[255];{локальна змінна для
        назви книги, або фрагмента назви}
begin
    assign(database, filename);
    reset(database);
    write('Введіть назву книги, або її частину:');
    readln(book_title);
    while eof(database)=false do
    begin
        read(database,data);
```

```
        if pos(book_title, data.book_title) > 0 then
            writeln(data.author,'  ',data.book_title,
            ' ', data.publisher,' ',data.publish_year);
    end
end;
procedure read_by_publisher(filename: string);
    var data: book;
    publisher : string[50]; {локальна змінна для пошуку
    співпадає за типом з відповідним полем запису}
begin
    assign(database, filename);
    reset(database);
    write('Видавництво:'); readln(publisher);
    while eof(database) = false do
    begin
        read(database,data);
        if data.publisher = publisher then
        writeln(data.author,'  ',data.book_title,
        ' ', data.publisher,' ',data.publish_year);
    end
end;

begin
  repeat
    writeln('Оберіть потрібну дію:');
    writeln('1 - шукати книгу за назвою.');
    writeln('2 - шукати книги за прізвищем автора.');
    writeln('3 - вивести книги вказаного видавництва');
    writeln('4 - переглядnути всі книги в базі');
    writeln('0 - завершити роботу');
```

```
    readln(choise);
    case (choise) of
      1:find_book(db_name);
      2:find_by_author(db_name);
      3:read_by_publisher(db_name);
      4:readall(db_name);
    end;
  until(choise = 0)
end.
```

Програма на C++.

В програмі на ***C++*** для коректного читання структурованих даних з файла записаного іншою програмою достатньо повністю продублювати оголошення структури з програми, якою цей файл було створено. Тут є два аспекти: загальний розмір у байтах екземпляра структури (***sizeof(book)***), що забузпечує читання даних з файла такими ж «порціями», та послідовність оголошення полів у самій структурі, що задає відповідність кожної частини байтів збереженим в них даним структури.

Програма-розв’язок задачі 600 мовою C++:

```
#include <iostream>
#include <fstream>
#include "Windows.h"
using namespace std;
// Тип даних для опису однієї книги повинен співпадати з
// типом, використаним для запису даних у файл
struct book
{
  char publisher[50];
  char book_title[255];
  char author[100];
  int publish_year;
  double price;
  int count;
};
```

```
//назва файлу з даними
const char db_name[] = "books_shop.db";
//прототип функції для читання усіх книг
void readall(const char*);

//прототип функції для пошуку книг за авторами
void show_by_authors(const char*, const char*);

//прототип функції для пошуку книг за назвою
void show_by_name(const char*, const char*);

//прототип функції для пошуку книг за видавництвом
void show_by_publisher(const char*, const char*);

//головна функція (програма)
int main()
{
  SetConsoleCP(1251); SetConsoleOutputCP(1251);
  int choise;
  do {
    system("cls");
    cout << "Оберіть потрібну дію : \n";
    cout << "1 - переглянути всі книги в базі \n";
    cout << "2 - шукати книги за прізвищем автора \n";
    cout << "3 - шукати книги за назвою \n";
    cout << "4 - шукати книги за видавництвом \n";
    cout << "0 - завершити роботу програми \n";
    cin >> choise; cin.get();
    char buff[255] = {};//фрагмент тексту для пошуку
```

```
    switch (choise) {
    case 1:
      readall(db_name);
      break;
    case 2:
      cout << "Введіть прізвище автора: ";
      cin.getline(buff, 255);
      show_by_authors(db_name, buff);
      break;
    case 3:
      cout << "Введіть назву книги або її частину: ";
      cin.getline(buff, 255);
      show_by_name(db_name, buff);
      break;
    case 4:
      cout << "Введіть назву видавництва: ";
      cin.getline(buff, 255);
      show_by_publisher(db_name, buff);
      break;
    }
  } while (choise != 0);
  return 0;
}

// реалізації функцій, оголошених прототипами на початку
void readall(const char* filename)
{
  book data;
  FILE* database;
  fopen_s(&database, filename, "rb");
```

```
  while (true)
  {
    int count = fread(&data, sizeof(data), 1, database);
    if (count == 0) break;
    cout << data.author << " " << data.book_title
      << " " << data.publisher << " "
      << data.publish_year << endl;
  }
  system("pause");
}

void show_by_authors(const char* filename,
                              const char* author)
{
  book data;
  FILE* database;
  fopen_s(&database, filename, "rb");
  while (true)
  {
    int count = fread(&data, sizeof(data), 1, database);
    if (count == 0) break;
    if (strstr(data.author, author) != NULL)
    {/*Функція strstr повертає покажчик на перший символ
     першого входження вказаного фрагмента
     у задану стрічку. Якщо співпадіння не
     знайдено, - повертається нульовий вказівник NULL*/
      cout << data.author << " " << data.book_title
        << " " << data.publisher << " "
        << data.publish_year << endl;
    }
```

```
  }
  system("pause");
}

void show_by_name(const char* filename,
                            const char* title_fragment)
{
  book data;
  FILE* database;
  fopen_s(&database, filename, "rb");
  while (true)
  {
    int count = fread(&data, sizeof(data), 1, database);
    if (count == 0) break;
    if (strstr(data.book_title, title_fragment) != NULL)
    {
      cout << data.author << " " << data.book_title
        << " " << data.publisher << " "
        << data.publish_year << endl;
    }
  }
  system("pause");

}

void show_by_publisher(const char* filename,
                                 const char* publisher)
{
  book data;
  FILE* database;
```

```
  fopen_s(&database, filename, "rb");
  while (true)
  {
    int count = fread(&data, sizeof(data), 1, database);
    if (count == 0) break;
    if (strcmp(data.publisher, publisher) == 0)
    {
      cout << data.author << " " << data.book_title
        << " " << data.publisher << " "
        << data.publish_year << endl;
    }
  }
  system("pause");
}
```

Програма на C#.

В програмі на ***C#*** до задачі 570 ми для запису даних про книги використовували серіалізацію засобами класу ***BinaryFormatter***, тому тут маємо використати його для отримання (десеріалізації) даних з файла. Крім цього слід врахувати, що попередньою програмою у файлі записано колекцію структур типу ***Task570.book***. Такий самий тип ми отримаємо десеріалізуючи дані з файла, тому опис структури потрібно помістити у відповідний простір імен (***Task570***).

При записі у файл ми зберегли дані у вигляді списку, тому пошук будемо виконувати не у самому файлі, а у десеріалізованому з файла списку в оперативній пам’яті. Функцію для виводу усіх записів залишимо, щоб можна було перевірити коректність роботи прогами.

Решта особливостей C#-реалізації пошуку записів пропонуємо розглянути в наступному коді:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using Task570;// дає можливість використовувати
```

```
//коротку назву типу структури book в інших просторах імен

namespace Task570// простір імен з попередньої
         // задачі потрібен для коректної десеріалізації
{
  // Структура для зберігання даних про книгу
  [Serializable]
  struct book
  {
    public string publisher;
    public string book_title;
    public string author;
    public int publish_year;
    public double price;
    public int count;
  };
}
namespace Task600   //простір імен класу з програмою
{
  class Program
  {
    //назва файлу з даними
    const string db_name = "books_shop.db";

    //головний метод (програма)
    static void Main(string[] args)
    {
      int choise;
      do
      {
```

```
    Console.Clear();
    Console.WriteLine("Оберіть потрібну дію :");
    Console.WriteLine("1 - знайти книги автора;");
    Console.WriteLine("2 - пошук книг за назвою;");
    Console.WriteLine("3 - список книг видавництва;");
    Console.WriteLine("4 - переглянути всі книги;");
    Console.WriteLine("0 - завершити роботу.");
    choise = Convert.ToInt32(Console.ReadLine());
    switch (choise)
    {
      case 1:
        FindByAuthors(db_name);
        break;
      case 2:
        FindByName(db_name);
        break;
      case 3:
        FindByPublisher(db_name);
        break;
      case 4:
        ReadAll(db_name);
        break;
    }
  } while (choise != 0);
}
//статичний метод для читання усіх книг
private static void ReadAll(string filename)
{
  List<book> books = new List<book>();
  BinaryFormatter binaryFormatter =
```

```
    new BinaryFormatter();
  using (FileStream fileStream =
                        File.OpenRead(filename))
  {
    books = (List<book>)binaryFormatter
                       .Deserialize(fileStream);
    for (int i = 0; i < books.Count; i++)
    {
      Console.WriteLine(books[i].author + " "
        + books[i].book_title
        + " " + books[i].publisher + " "
        + books[i].publish_year);
    }
    Console.ReadKey();
  }
}
//статичний метод виводу книг вказаного видавництва
private static void FindByPublisher(string filename)
{
  List<book> books = new List<book>();
  BinaryFormatter binaryFormatter =
    new BinaryFormatter();
  using (FileStream fileStream =
                        File.OpenRead(filename))
  {
    // завантажуємо список книг з файла
    // в пам'ять програми, в змінну books
    books = (List<book>)binaryFormatter
                       .Deserialize(fileStream);
    Console.WriteLine("Введіть назву видавництва:");
```

```
    // отримуємо назву видавництва для пошуку
    string publisher = Console.ReadLine();
    // виводимо лише книги з відповідним значенням
    // поля publisher
    for (int i = 0; i < books.Count; i++)
    {
      // введену назву видавництва та зчитану з файла
      // для порівняння приводимо до нижнього регістра
      // (малих літер) оскільки цю назву могли записати
      // великими літерами, чи з великої літери, а ми
      // вважатимемо що це одне і те ж видавництво
      if(books[i].publisher.ToLower() ==
              publisher.ToLower())
      {
        Console.WriteLine(books[i].author + " "
         + books[i].book_title
         + " " + books[i].publisher + " "
         + books[i].publish_year);
      }

    }
    Console.ReadKey();
  }
}
//статичний метод для пошуку книг
// за прізвищем автора / авторів
private static void FindByAuthors(string filename)
{
  List<book> books = new List<book>();
  BinaryFormatter binaryFormatter =
```

```
    new BinaryFormatter();
  using (FileStream fileStream =
                          File.OpenRead(filename))
  {
    // завантажуємо список книг з файла
    // в пам'ять програми, в змінну books
    books = (List<book>)binaryFormatter
                          .Deserialize(fileStream);
    Console.WriteLine("Введіть прізвище автора:");
    // отримуємо прізвище автора для пошуку
    string author = Console.ReadLine();
    for (int i = 0; i < books.Count; i++)
    {
      // перевіряємо чи список авторів книги
      // містить вказане прізвище
      if(books[i].author.Contains(author))
      {
        Console.WriteLine(books[i].author + " "
        + books[i].book_title
        + " " + books[i].publisher + " "
        + books[i].publish_year);
      }
    }
    Console.ReadKey();
  }
}
//статичний метод пошуку книг за назвою
private static void FindByName(string filename)
{
  List<book> books = new List<book>();
```

```
      BinaryFormatter binaryFormatter =
        new BinaryFormatter();
      using (FileStream fileStream =
                              File.OpenRead(filename))
      {  // завантажуємо список книг з файла
        // в пам'ять програми, в змінну books
        books = (List<book>)binaryFormatter
                             .Deserialize(fileStream);
        Console.WriteLine("Введіть назву книги, " +
                                  "або її частину:");
        // отримуємо фрагмент назви для пошуку
        string title = Console.ReadLine();
        for (int i = 0; i < books.Count; i++)
        {
          if(books[i].book_title.Contains(title))
          {
            Console.WriteLine(books[i].author + " "
            + books[i].book_title
            + " " + books[i].publisher + " "
            + books[i].publish_year);
          }
        }
        Console.ReadKey();
      }
    }
  }
}
```

Програма на Python.

Нагадаємо що в програмі на ***Python*** до задачі 570 для запису даних про книги ми скористалися ***словниками***, а зберігали дані не у бінарному файлі а в текстовому зі спеціальним форматом запису даних ***csv*** (***Comma-Separated Values***). Такий формат має ряд переваг, перша з яких, – файл з даними можна читати за допомогою звичайного текстового редактора, за потреби, дані у csv-файл так само можна записувати за допомогою звичайного текстового редактора. Основна вимога при цьому, щоб кожен рядок даних містив однакову кількість фрагментів, розділених комами і вона відповідала кількості заголовків у першому рядку. Тобто, файл, створений програмою до задачі 570, за такого підходу, для розв'язування цієї задачі з не коче потрібен, – його можна створити заново.

Словник утворюється зі списку елементів, кожен з яких утворений з пари елементів: ***ключ***, тобто назва даних, та ***значення***, тобто самі дані. Для запису словників до файла модуль ***csv*** використовує спеціальний список, що складається з ключів словника, – його записують в першій стрічці файла. При зворотному процесі ***csv.DictReader*** буде використовувати першу стрічку для формування ключів до даних словників, записаних в наступних стрічках.

Як і в усіх інших прикладах до останніх двох задач, реалізуємо програму у вигляді окремих функцій для кожного пункту завдання. А їх виконання в основній програмі будемо запускати за вибором користувача з допомогою тектсового меню.

Код програми для розв'язування задачі 600 на мові ***Python***:

```
import csv


db_name = 'books_shop.csv'


# функція для пошуку записів про книги
# за вказаним прізвищем одного з авторів
def find_by_author(filename):
  author = input('Введіть прізвище автора\n>> ')
  with open(filename, 'r', newline='',
        encoding='cp1251') as file:
    reader = csv.DictReader(file)
    for row in reader:
      if author in row['authors']:
        print(row['authors'],' ', row['book_title'],', - ',
```

```
          row['publisher'], ': ', row['publish_year'],'. ',
          row['price'],' грн., ' , row['count'],
          ' в наявності.')
# функція для пошуку записів про книги
# за назвою, або її частиною
def find_by_title(filename):
  book = input('Введіть назву книги\n>> ')
  with open(filename, 'r', newline='',
        encoding='cp1251') as file:
    reader = csv.DictReader(file)
    for row in reader:
      if book in row['book_title']:
        print(row['authors'],' ', row['book_title'],', - ',
          row['publisher'], ': ', row['publish_year'],'. ',
          row['price'],' грн., ' , row['count'],
          ' в наявності.')
# функція для пошуку записів про книги
# за вказаним видавництвом
def find_by_publisher(filename):
  publisher = input('Введіть видавництво\n>> ')
  with open(filename, 'r', newline='',
        encoding='cp1251') as file:
    reader = csv.DictReader(file)
    for row in reader:
      if publisher == row['publisher']:
        print(row['authors'],' ', row['book_title'],', - ',
          row['publisher'], ': ', row['publish_year'],'. ',
          row['price'],' грн., ' , row['count'],
          ' в наявності.')
```

```
# Основна програма
# ненульове значення для входу в цикл
choise = 4
# за умовою в меню 0 - вихід з циклу, якщо
# ініціалізувати choise нулем, цикл не виконається
# жодного разу, а оголошення та ініціалізація
# змінної choise потрібні до початку циклу
while choise != 0:
  print('Оберіть потрібну дію:')
  print('1 - пошук книг за автором')
  print('2 - пошук книг за назвою')
  print('3 - пошук книг за видавництвом')
  print('0 - завершити роботу програми')
  choise = int(input('>>>  '))
  if choise == 1:
    find_by_author(db_name)
  elif choise == 2:
    find_by_title(db_name)
  elif choise == 3:
    find_by_publisher(db_name)
```

Рекомендована література:

1. Брнакевич І. Є., Вагін П. П. Програмування мовою Java: використання фундаментальних класів: Тексти лекцій. – Львів: Видавничий центр ЛНУ імені Івана Франка, 2002. – 75 с.
2. Збірник задач з дисципліни "Інформатика і програмування" / Вакал Є. С., Личман В. В., Обвінцев О. В., Бублик В. В., Довгий Б. П., Попов В. В. -2-ге видання, виправлене та доповнене –К.: ВПЦ "Київський університет", 2006.– 94 с.
3. Глинський Я. М., Анохін В. Є., Ряжська В. А. Г54. C++ і C++ Builder.- Львів: Деол, СПД Глинський, 2003. - 192 с.
4. Глинський Я. М. Паскаль. Turbo Pascal and Delphi /Я. М. Глинський, В. Є. Анохін, В. А. Ряжська. – 10-те вид. – Львів : СПД Глинський, 2009. – 196 с.
5. Глинський Я. М. Інформатика. Основи алгоритмізації іпрограмування мовою Visual Basic / Я. М. Глинський. – Львів : СПДГлинський, 2011. – 272 с.
6. Програмування на С(С++). Парадигма процедурного програмування. Навчальний посібник. /Гнатів Б.В., Гнатів Л.Б. . – Львів: Видавництво «Растр-7», 2017. – 264 с.
7. Демків І.І., Кравець І.Т., Каленюк П.І., Клюйник І.І.,Петрович Р.Й. Основи програмування в середовищі турбо-паскаль. Лекції та завдання до лабораторних робіт. Навчальний посібник. Рекомендований Міністерством Освіти і Науки України. Львів: Каменяр — 2003. 202 с
8. П. Каленюк, І. Клюйник, І. Кравець, Л. Гнатів, Л.Демків, І. Подун Основи програмування в середовищі С. Лекції та завдання до лабораторних робіт. Львів 2007 р., 98 с
9. Практикум з програмування на VBA: Навч. посібник / П.І. Каленюк, А.Ф. Обшта, Н.М.Гоблик, Н.Ф.Клочко, С.М.Ментинський. Львів: Видавництво Національного університету «Львівська політехніка», 2005. -208 с.
10. Практикум з програмування (Turbo PASCAL, Object PASCAL – Delphi): Навч. посібник / П.І. Каленюк, А.Ф. Обшта, Н.М.Гоблик, Н.Ф.Клочко, С.М.Ментинський. Львів: Видавництво Національного університету «Львівська політехніка», 2005. – 176 с.
11. Караванова Т.П. Основи алгоритмізації та програмування: 750 задач з рекомендаціями та прикладами: Посіб. –К.: Форум, 2002. – 287 с.
12. Караванова Т.П. Інформатика. Збірник вправ та задач з алгоритмізації та програмування (процедурне програмування). Навч. посіб. Доп. та випр. – Шепетівка: Аспект, 2004.–160 с.

13. Караванова Т.П. Основи алгоритмізації та програмування: 777 задач з рекомендаціями та прикладами: Навч. посіб. Доп. та випр. – К.: Генеза, 2006. – 288 с.
14. Копей В. Б. Мова програмування Python для інженерів і науковців: навчальний посібник / В. Б. Копей – Івано-Франківськ : ІФНТУНГ, 2019. – 272 с.
15. Копитко М.Ф., Іванків К.С. Основи програмування мовою Java: Тексти лекцій. – Львів: Видавничий центр ЛНУ ім. Івана Франка, 2002. – 83 с.
16. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020. 180 с.
17. Кравець П.О. Об’єктно-орієнтоване програмування: навч. посібн. / П. О. Кравець.— Львів: Вид-во Львівської політехніки, 2012.— 624 с.
18. Креневич А. П. С у задачах і прикладах : навчальний посібник із дисципліни "Інформатика та програмування" / А. П. Креневич, О. В. Обвінцев. – К. : Видавничо-поліграфічний центр "Київський університет", 2011. – 208 с.
19. Кукурудз С.Ф.Процюк В.Р., Ваврик Т.О. Збірник задач з програмування. Навчальний посібник. – Івано-Франківськ: Факел, 2005. – 247с.
20. Основи програмування на С++. Навчальний посібник з курсу «Основи інформатики і програмування, частина 2» спеціальності 105 – “Прикладна фізика та наноматеріали” для першого (бакалаврського) рівня освіти/ С. М. Ментинський, Я. М. Пелех. – Львів: Галицька Видавнича Спілка, 2021. – 256 с.
21. Петрович Р. Й., Тумашова О. В. “Основи програмування мовою Сі.”. Навчальний посібник. Видавництво НУ”ЛП”, Львів, 2005р. 100с.
22. Семотюк В.Програмування в середовищі Турбо Паскаль. Львів: БаК, 2000. - 248с.
23. Програмування мовою С: навчальний посібник для вузів / Зореслава Ярославівна Шпак . — Львів: Оріяна-Нова, 2006 . — 431 с.
24. Шпак З.Я. Програмування мовою С. – Львів: Видавництво Львівської політехніки, 2011. – 436 с.

Навчальне видання

Ментинський Сергій Мирославович

Пелех Ярослав Миколайович

ЗБІРНИК ЗАДАЧ З ОСНОВ АЛГОРИТМІЗАЦІЇ ТА ПРОГРАМУВАННЯ.

Навчальний посібник з курсів
«Обчислювальна техніка та програмування», «Інформатика»,
«Основи інформатики і програмування»,
для студентів технічних спеціальностей
для першого (бакалаврського) рівня освіти

Комп'ютерне верстання: – *С. Ментинський*

Підписано до друку 1.04.2023 р. Формат 60*84/16.
Гарнітура Times New Roman. Папір офсетний. Друк офсетний.
Ум. друк. арк. 18,6. Наклад 100 прим. Зам. № 2964

Видавництво ТзОВ «Колір ПРО»
вул. Галицької Армії, 1А, м. Львів, 79012
тел. (032) 238-63-81, (032) 238-63-82
Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції серія ДК № 3794 від 31.05.2010 р.

Друк «Компанія "Манускрипт"»
вул. Руська, 16/3, м. Львів, 79008
тел./факс: (032) 235-60-00.
Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції серія ДК № 3628 від 19. 11. 2009 р.