МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Л. І. Кублій

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ

ПРАКТИКУМ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для здобувачів ступеня бакалавра за спеціальністю 122 “Комп’ютерні науки”

Київ
КПІ ім. Ігоря Сікорського
2019

Рецензент: *Баранюк О.В.,* к. т. н., доц. каф. АЕС і ІТФ

Відповідальний редактор *Варава І.А.,* к. т. н., доц. каф. АПЕПС

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 10 від 20.06.2019 р.) за поданням Вченої ради Теплоенергетичного факультету (протокол № 10 від 29.05.2019 р.)

Електронне мережне навчальне видання

Кублій Лариса Іванівна, канд. техн. наук, доц.

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ
ПРАКТИКУМ

Алгоритмізація та програмування: Практикум [Електронний ресурс]: навч. посіб. для здобувачів ступеня бакалавра за спеціальністю 122 “Комп’ютерні науки” / Л. І. Кублій; КПІ ім. Ігоря Сікорського. — Електронні текстові дані (1 файл: 28,15 Мбайт). — Київ: КПІ ім. Ігоря Сікорського, 2019. — 209 с.

Кредитний модуль “Алгоритмізація та програмування” студенти вивчають на першому курсі в першому семестрі і він є необхідним практичним і теоретичним фундаментом для фахівців з інформаційних технологій, передбачає навчання студентів проектування алгоритмів і розробки програм з використанням сучасних технологій програмування, освоєння роботи на сучасних обчислювальних засобах. Подано завдання до практичних занять, зразки їхнього виконання. Базовою мовою навчання є мова програмування С.

Посібник призначений для здобувачів ступеня бакалавра за спеціальністю “Комп’ютерні науки”, а також для тих, хто вивчає програмування самостійно.

Зміст

Вступ

Кредитний модуль “Алгоритмізація та програмування” викладається студентам першого року підготовки першого (бакалаврського) рівня вищої освіти за освітньою програмою “Комп’ютерний моніторинг та геометричне моделювання процесів і систем” спеціальності “Комп’ютерні науки” у першому навчальному семестрі.

Дисципліна “Алгоритмізація та програмування” не має дисциплін, які її забезпечують, вона вивчається на основі знань, отриманих у середній загальноосвітній школі. Компетенції, отримані студентами в процесі вивчення цієї дисципліни застосовуються ними в подальшому при вивченні навчальних дисциплін: “Програмування алгоритмічних структур”, “Об’єктно-орієнтоване програмування”, “Комп’ютерна схемотехніка та архітектура комп’ютерів”, “Поглиблена архітектура комп’ютерів”, “Спеціальні засоби мови програмування”.

Метою кредитного модуля є формування у студентів здатностей:

— застосовувати професійні знання й уміння на практиці;

— використовувати математичні методи для прийняття ефективних рішень під час розв’язання професійних задач в процесі розробки ІС та ІТ;

— до проектування архітектури комп’ютерної системи, вибору і інтегруванню компонентів технічного і стандартного програмного забезпечення при реалізації ІС та ІТ;

— використання принципів структурного програмування, основних структур даних під час реалізації алгоритмів професійних завдань;

— володіння алгоритмічним мисленням, методами програмної інженерії для реалізації програмного забезпечення з урахуванням вимог до його якості, надійності, виробничих характеристик.

Згідно з вимогами програми навчальної дисципліни студенти після засвоєння кредитного модуля мають продемонструвати такі результати навчання:

***знання*:**

— основ організації обчислювального процесу на ЕОМ;

— теоретичних основ алгоритмізації, проектування й тестування програм;

— методів структурного програмування;

— алгоритмів розв'язування типових задач;

— засобів програмування алгоритмічної мови С;

вміння:

— застосовувати мови програмування (зокрема мову С), мови опису інформаційних ресурсів, мови специфікацій, інструментальні засоби під час проектування та створення інформаційних систем, продуктів і сервісів;

— програмно реалізувати алгоритми розв'язання задач, розроблення програмного забезпечення інформаційних систем;

— проектувати компоненти програмного забезпечення;

— проектувати людино-машинний інтерфейс інформаційних систем;

— розробляти оптимальні алгоритми для широкого кола задач;

— реалізувати алгоритми мовою програмування С як закінченого програмного продукту;

— виконувати аналіз коректності програм, налагодження та тестування з використанням сучасних технологій програмування;

— застосовувати набуті базові знання з дисципліни в професійній діяльності під час розробки, налагодження та експлуатації ІС та технологій;

— застосовувати принципи структурного програмування при проектуванні і розробці програм;

— застосовувати основні структури даних при програмнуванні алгоритмів;

— застосовувати типові алгоритми обробки даних.

Основні завдання циклу практичних занять полягають в тому, щоб студенти закріпили матеріал лекцій і отримали практичні навички розробки алгоритмів і проектування програм з використанням сучасних технологій структурного програмування, освоєння роботи на сучасних обчислювальних засобах і тестування програм.

Робочою програмою кредитного модуля "Алгоритмізація та програмування" передбачено два види практичних занять:

— заняття, на яких розглядаються й аналізуються типові алгоритми, записується їхній код мовою програмування С;

— заняття, на яких студенти виконують роботи комп'ютерного практикуму за варіантами, пов'язані з програмуванням і налагодженням програм в комп'ютерному програмному середовищі (рекомендоване середовище програмування — Microsoft Visual Studio).

Нижче вказано теми практичних занять і кількість аудиторних годин, яка на них відводиться.

№ з/п	*Назва теми заняття*	*Кількість ауд. годин*
Розділ 1. Організація програм		
1	Системи числення. Словесне подання алгоритмів. Типи даних, визначені стандартом мови С.	2
2	*Комп'ютерний практикум 1.* Основи роботи в середовищі Microsoft Visual Studio — мова С (консольний режим). Приклад найпростішої програми мовою С.	2
3	*Комп'ютерний практикум 2*. Оформлення звітів засобами текстового редактора Word (побудова блок-схем, вставка формул, подання результатів роботи).	2
4	Арифметичні й логічні вирази. Введення-виведення даних. Умовні оператори.	
5	*Комп'ютерний практикум 3.* Програмування найпростіших обчислювальних алгоритмів лінійної структури.	2
6	*Комп'ютерний практикум 4.* Розробка і реалізація алгоритмів розгалужених процесів з використанням послідовних умовних операторів.	2
7	Умовні оператори. Проектування алгоритмів і програм циклічної структури: цикл з параметром, цикл з передумовою.	2
8	*Комп'ютерний практикум 5.* Розробка і реалізація алгоритмів розгалужених процесів з використанням вкладених умовних операторів і оператора вибору.	2
9	*Комп'ютерний практикум 6.* Проектування алгоритмів і програм циклічної структури (оператор циклу з параметром).	2

10	Циклічні структури з післяумовою, ітераційні цикли. Вкладені цикли.	2
11	*Комп'ютерний практикум 7.* Проектування алгоритмів і програм циклічної структури (оператори циклів з передумовою і післяумовою).	2
12	*Комп'ютерний практикум 8.* Проектування алгоритмів і програм з вкладеними циклами.	2
13	*Комп'ютерний практикум 9.* Проектування ітераційних алгоритмів.	2
14	Розробка програм модульної структури: функції без параметрів і з параметрами. Передача параметрів. Рекурсія.	2
15-16	*Комп'ютерний практикум 10-11.* Розробка програм модульної структури з використанням функцій. Налагодження програм.	4
Розділ 2. Структури даних і алгоритми		
17	Обробка одновимірних масивів. Алгоритми сортування в одновимірних масивах.	2
18	*Комп'ютерний практикум 12.* Обробка одновимірних масивів.	2
19	*Комп'ютерний практикум 13.* Сортування в одновимірних масивах.	2
20	Обробка двовимірних масивів.	2
21	*Комп'ютерний практикум 14.* Обробка двовимірних масивів.	2
22	Робота з рядками символів. Побітові операції. Робота зі структурами.	2
23	*Комп'ютерний практикум 15.* Робота з рядками символів.	2
24	*Комп'ютерний практикум 16.* Робота зі структурами.	2
25	*Комп'ютерний практикум 17.* Виконання додаткового завдання.	2
26	Робота з динамічною пам'яттю.	2
27	*Диференційований залік*	*2*
	Усього	**54**

Готуючись до практичних занять, студенти мають проаналізувати подані в навчальному посібнику приклади і продумати алгоритми виконання вказаних завдань.

Практичне заняття № 1

Системи числення. Словесне подання алгоритмів. Типи даних, визначені стандартом мови С

Мета заняття: закріпити лекційний матеріал, який стосується: систем числення, використовуваних у програмуванні — двійкової, вісімкової, шістнадцяткової; застосування алгоритмів переходу для подання чисел у різних системах числення; набуття навичок побудови алгоритмів і їхнього словесного подання; ознайомлення з основними типами даних мови програмування С і правилами перетворення типів у виразах.

1. Системи числення. У програмуванні, крім десяткової системи числення, використовують подання чисел у двійковій, вісімковій і шістнадцятковій системах.

Двійкова система числення.

Приклад 1.1. Нехай число подано в двійковій (основа 2) системі числення: $11011{,}1001_2$. Знайдемо його подання в десятковій системі. У відповідність цифрам заданого числа поставимо показники степенів основи 2:

$$\overset{4}{1}\ \overset{3}{1}\ \overset{2}{0}\ \overset{1}{1}\ \overset{0}{1}\ ,\ \overset{-1}{1}\ \overset{-2}{0}\ \overset{-3}{0}\ \overset{-4}{1}\ .$$

Двійкове число подамо так:

$$11011{,}1001_2 =$$

$$= 1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-1} + 0\cdot 2^{-2} + 0\cdot 2^{-3} + 1\cdot 2^{-4} =$$

$$= 1\cdot 16 + 1\cdot 8 + 0\cdot 4 + 1\cdot 2 + 1\cdot 1 + 1\cdot \frac{1}{2} + 0\cdot \frac{1}{4} + 0\cdot \frac{1}{8} + 1\cdot \frac{1}{16} =$$

$$= 16 + 8 + 0 + 2 + 1 + 0{,}5 + 0 + 0 + 0{,}0625 = 27{,}5625_{10}\ .$$

Алгоритм переходу від двійкового подання числа до десяткового (словесне подання алгоритму):

К1. Поставити у відповідність цифрам цілої частини двійкового числа, ідучи справа наліво, числа 0, 1, 2, 3…, а цифрам дробової частини, ідучи зліва направо, — числа -1, -2, -3… Ці числа є показниками степенів основи 2.

К2. Знайти суму добутків цифр (0 чи 1) двійкового числа на відповідні їм степені числа (основи) 2. Одержане значення суми є десятковим поданням двійкового числа.

Приклад 1.2. Число подано в десятковій системі числення (основа 10): $123{,}45_{10}$. Знайдемо його подання в двійковій системі. Цілу і дробову частини треба перетворювати окремо. Одержимо двійкові цифри цілої частини:

$123 : 2 =$ 61 і остача **1**;
$61 : 2 =$ 30 і остача **1**;
$30 : 2 =$ 15 і остача **0**;
$15 : 2 =$ 7 і остача **1**; ↑
$7 : 2 =$ 3 і остача **1**;
$3 : 2 =$ 1 і остача **1**;
$1 : 2 =$ 0 і остача **1**.

Остачі від ділення запишемо у зворотному порядку 1111011_2.

Одержимо двійкові цифри дробової частини:

$0{,}45 \cdot 2 =$ 0,9 ціла частина **0**, дробова 0,9;
$0{,}9 \cdot 2 =$ 1,8 ціла частина **1**, дробова 0,8;
$\mathbf{0{,}8} \cdot 2 =$ 1,6 ціла частина **1**, дробова 0,6;
$0{,}6 \cdot 2 =$ 1,2 ціла частина **1**, дробова 0,2; ↓
$0{,}2 \cdot 2 =$ 0,4 ціла частина **0**, дробова 0,4;
$0{,}4 \cdot 2 =$ 0,8 ціла частина **0**, дробова 0,8;
$\mathbf{0{,}8} \cdot 2 =$ 1,6 ціла частина **1**, дробова 0,6...

У даному випадку при перетворенні скінченного дробового числа $0{,}45_{10}$ у двійкове одержано періодичний двійковий дріб $0{,}01(1100)_2$. Отже:

$$123{,}45_{10} = 1111011{,}01(1100)_2.$$

Алгоритм переходу від десяткового подання числа до двійкового:

К1. Виділити в десятковому числі цілу і дробову частини.

К2. Якщо ціла частина дорівнює нулю, то перейти на К3. Якщо ціла частина відмінна від нуля, то поділити її на 2 — буде одержано цілу частину і остачу (0 чи 1); запам'ятати, що виконувалося ділення; перейти на К2.

К3. Якщо ділення виконувалося, то одержані остачі від ділення записати у зворотному порядку — буде одержано двійкове подання цілої частини числа (інакше ціла частина залишається нульовою).

К4. Якщо дробова частина дорівнює нулю, то перейти на К5. Якщо дробова частина відмінна від нуля, то помножити її на 2 — буде одержано цілу частину (0 чи 1) і дробову; запам'ятати, що виконувалося множення. Якщо дробова частина повторюється (двійковий дріб — періодичний), то перейти на К5 чи передбачити іншу дію; інакше перейти на К4.

К5. Якщо множення виконувалося, то одержані після множення цілі частини записати у прямому порядку — буде одержано двійкове подання дробової частини числа (інакше дробова частина залишається нульовою).

К6. Сформувати з двійкових подань цілої і дробової частин двійкове подання числа.

Зауваження 1. Подання цілого числа в двійковому вигляді потребує приблизно в три рази більше розрядів, ніж у десятковому (наприклад, $3_{10} = 11_2$, $4_{10} = 100_2$. $9_{10} = 1001_{10}$, $37_{10} = 100101_2$, $99_{10} = 1100011_2$, $100_{10} = 1100100_2$, $586_{10} = 1001001010_2$, $1000_{10} = 1111101000_2$).

Зауваження 2. При переведенні десяткових дробів у двійкові, як правило, буде одержано періодичний двійковий дріб (наприклад, для чисел 0,1–0,4, 0,6–0,9). Періодичний дріб — нескінченний, а комп'ютер оперує лише зі скінченними числами, поданими скінченними послідовностями 0 і 1. Це призводить до помилок заокруглення (наприклад, після виконання Pascal-оператора `if 0.1+0.6=0.7 then write('true') else write('false')` буде одержано результат `false`; оператори мови C `if (0.1+0.1+0.1==0.3) printf("true"); else printf("false"); if (0.1+0.2==0.3) printf("true"); else printf("false");` — обидва дають `false`). Щоб зменшити ці помилки, в комп'ютерах реалізовано спеціальний алгоритм заокруглення[1]. Але повністю позбутися помилок закруглення при проведенні комп'ютерних обчислень не можливо; можна тільки збільшити точність обчислень, збільшивши кількість розрядів для подання дробів (наприклад, якщо під дробову частину відвести чотири розряди, то значення числа $0{,}2_{10}$ буде

[1] IEEE Standard for Binary Floating-Point Arithmetic. IEEE Std 754-2008 (Revision of

зберігатися як $0{,}0011_2$ і якщо його перевести назад в десяткове, то одержимо $0{,}0011_2 = \frac{1}{8} + \frac{1}{16} = 0{,}1875_{10}$, якщо ж відвести вісім розрядів, то $0{,}00110011_2 = \frac{1}{8} + \frac{1}{16} + \frac{1}{128} + \frac{1}{256} = 0{,}19841875_{10}$).

Зауваження 3. Десяткові дроби, кратні степеневі числа $\frac{1}{2}$, переводяться в скінченні двійкові дроби (наприклад, $\frac{3}{4} = 0{,}75_{10} = 0{,}11_2$, $\frac{11}{16} = 0{,}6875_{10} = 0{,}1011_2$, $\frac{13}{32} = 0{,}40625_{10} = 0{,}01101_2$). Тому робота з ними, як і з цілими числами, до помилок заокруглення не призводить.

Вісімкова й шістнадцяткова системи числення. Записувати числа, а також програмувати мовою машинних кодів, використовуючи 1 і 0, досить складно і треба бути дуже уважним. Тому для спрощення запису застосовують вісімкову й шістнадцяткову системи числення.

Для переходу від двійкового до вісімкового подання числа треба в двійковому поданні цілої частини виділити, ідучи справа наліво, по три розряди (якщо зліва розрядів не вистачає, то дописати потрібну кількість нулів), у двійковому поданні дробової частини іти зліва направо і теж виділити по три розряди (якщо справа розрядів не вистачає, то дописати потрібну кількість нулів), а потім скористатися таблицею відповідностей:

Десяткове число	Двійкове число	Вісімкова цифра
0	000	0
1	001	1
2	010	2
3	011	3

Десяткове число	Двійкове число	Вісімкова цифра
4	100	4
5	101	5
6	110	6
7	111	7

Приклад 1.3. Подамо число $1111011000{,}11101_2$ у вісімковій системі:

$$\underbrace{\overbrace{00}^{\wedge}1}_{1}\underbrace{111}_{7}\underbrace{011}_{3}\underbrace{000}_{0},\underbrace{111}_{7}\underbrace{010}_{2}\overset{\wedge}{}$$

IEEE Std 754-1985). — NY: IEEE, 2008. — 70 p.

Отже, $1111011000{,}11101_2 = 1730{,}72_8$.

Аналогічно здійснюється перехід від двійкового до шістнадцяткового подання, тільки замість трьох треба виділяти по чотири розряди і скористатися такою таблицею відповідностей:

Десяткове число	Двійкове число	Шістнадцяткова цифра
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

Десяткове число	Двійкове число	Шістнадцяткова цифра
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Приклад 1.4. Подамо число $1111011000{,}11101_2$ у шістнадцятковій системі числення:

$$\underbrace{\overbrace{001}}_{3}\underbrace{111}_{D}\underbrace{011}_{8}\underbrace{000}_{E},\underbrace{111}_{8}\underbrace{\overbrace{010}}_{8}\underbrace{\overbrace{00}}$$

Отже, $1111011000{,}11101_2 = 3D8{,}E8_{16}$

Зворотний перехід від вісімкового й шістнадцяткового подання до двійкового виконується аналогічно на основі таблиць відповідностей.

Прямий перехід від десяткового подання до вісімкового і шістнадцяткового виконується, як і для двійкового, тільки ділити чи множити треба відповідно на 8 і 16.

Приклад 1.5. Подамо десяткове число $984{,}90625_{10}$ у шістнадцятковій системі числення. Виконаємо для цілої частини ділення, а для дробової множення на 16:

$$\begin{array}{lll} 984:16 = & 61 & \text{і остача } \mathbf{8}; \\ 61:16 = & 3 & \text{і остача } \mathbf{13} \sim \mathbf{D}; \uparrow \\ 3:16 = & 0 & \text{і остача } \mathbf{3}. \end{array}$$

$0{,}90625 \cdot 16 = \quad 14{,}5$ ціла частина **14** ~ **E**, дробова 0,5; ↓
$0{,}5 \cdot 16 = \quad 8$ ціла частина **8**, дробова 0.

Отже, $984{,}90625_{10} = 3D8, E8_{16}$.

Виконання операцій над числами у двійковій системі числення.

Додавання в десятковій системі числення виконується за модулем 10, а в двійковій — за модулем 2:

x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Як і в десятковій системі числення, у двійковій теж застосовують перенесення значень у старші розряди. При цьому $1_2 + 1_2 = 10_2$, $1_2 + 1_2 + 1_2 = 11_2$.

Приклад 1.6. Знайдемо суму чисел $287_{10} = 100011111_2$ і $94_{10} = 1011110_2$:

переноси
$$\begin{array}{cccc} & {\scriptstyle 1} & {\scriptstyle 1} & \\ + & 2 & 8 & 7 \\ & & 9 & 4 \\ \hline & 3 & 8 & 1 \end{array} \qquad \begin{array}{ccccccccc} & & & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & & \\ + & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ & & & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ \hline & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{array}$$

У комп'ютері віднімання реалізується тією ж схемою (суматором), що і додавання.

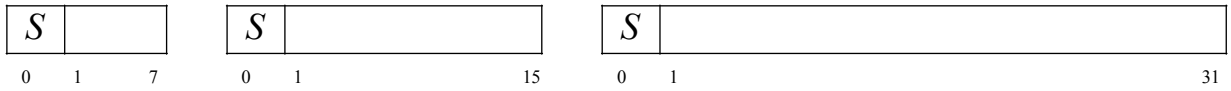
Для *подання від'ємних чисел* у комп'ютері використовується *додатковий код*: додатковий код = зворотний код + 1, де зворотний код є логічним доповненням (1 треба замінити на 0, а 0 — на 1) коду додатного числа.

Приклад 1.7. Одержимо додатковий код числа -5_{10}. Нехай для зберігання цілого числа відводиться 2 байти, тобто 16 бітів (розрядів). Тоді число 5_{10} кодується так: 0000000000000101_2 або для полегшення запису можна використати шістнадцятковий код 0005_{16}.

Код числа 5_{10}	0000000000000101_2
Зворотний код числа 5_{10}	1111111111111010
Додатковий код	1111111111111011,

тобто від'ємне число -5_{10} у комп'ютері кодується так: 1111111111111011.

Перший зліва розряд коду (має номер 0) вказує знак числа: 0 — число додатне, 1 — число від'ємне. Якщо для подання цілого числа виділяється 8, 16 чи 32 розряди, то відповідно пам'ять розподіляється так:



де S — знаковий розряд числа: $S(+)=0$, $S(-)=1$.

Завдання. Подати у двійковій системі числення такі десяткові числа: 252; 34,625; 0,45. Перевірити правильність подання — перейти від двійкової системи числення до десяткової. Подати ці числа у вісьмірковій і шістнадцятковій системаї числення.

2. Словесний опис алгоритмів. Словесне подання алгоритму є по суті його описом звичайною мовою, але з ретельно підібраними словами, так, щоб в записові алгоритму не було зайвих слів, синонімів тощо, доповненою математичними позначеннями і деякими спеціальними домовленостями (про знак присвоювання, про нумерацію кроків алгоритму тощо). Таке подання легко сприймати, від нього легко перейти до кодової форми мовою програмування. Але деякі алгоритми можуть бути дуже складними, щоб зразу перейти від словесного опису до програми, тому може бути потрібною деталізація.

Подані вище алгоритми переходу від двійкового подання чисел до десяткового і навпаки є прикладами словесного подання алгоритмів.

Розглянемо ще приклади простих типових алгоритмів.

Приклад 2.1. Знайти абсолютне значення числа і вивести його:

К1. Ввести число a.

К2. Якщо a<0, то a=–a.

К3. Вивести a.

Приклад 2.2. Знайти більше серед двох чисел:

К1. Ввести числа a і b.

К2. Якщо a>b, то вивести a, інакше вивести b.

Приклад 2.3. Два числа впорядкувати за зростанням:

К1. Ввести числа a, b.

К2. Якщо a>b, то r=a, a=b, b=r.

К3. Вивести a, b.

Будь-які дані (наприклад, числа) зберігаються в пам'яті машини. Залежно від того, якого типу ці дані, вони можуть займати різну величину пам'яті (кількість комірок). Ділянкам пам'яті при розробці алгоритму, написанні програми дають імена:

X Y

Приклад 2.4. Взаємно поміняти значення двох ділянок пам'яті.

Нехай ці ділянки пам'яті мають імена X і Y.

Спосіб 1. Використовуємо додаткову ділянку пам'яті (а в алгоритмі додаткову робочу змінну R):

К1. Введення X і Y

К2. R=X.

К3. X=Y.

К4. Y=R.

К5. Виведення X і Y.

Перевіримо, чи правильно працює даний алгоритм. Нехай X=5, Y=3.

Ділянки пам'яті	X	Y	R
Початковий стан пам'яті	∞	∞	∞
К1	**5**	**3**	∞
К2	5	3	5
К3	3	3	5
К4	**3**	**5**	5
К5	**3**	**5**	

Спосіб 2. Не використовуємо додаткову ділянку пам'яті:

К1. Введення X і Y

К2. X=X+Y.

К3. Y=X–Y.

К4. X=X–Y.

К5. Виведення X і Y.

Перевіримо, чи правильно працює даний алгоритм. Нехай X=5, Y=3.

Ділянки пам'яті	X	Y
Початковий стан пам'яті	∞	∞
К1	**5**	**3**
К2	5+3=8	3
К3	8	8–3=5
К4	8–5=**3**	**5**
К5	**3**	**5**

Але цей спосіб не завжди дасть правильну відповідь:

1) може бути переповнення пам'яті при додаванні навіть цілих чисел (наприклад, якщо число дворозрядне, то 51+50=101, але результат буде дворозрядним, тобто 01, і далі алгоритм дасть не той результат);

2) втрата розрядів (наприклад, виділяється два розряди під дійсне число 3.5+0.51= (у пам'яті зберігається як)=$3.5\ 10^0+5.1\ 10^{-1}=3.51\ 10^0$, але результат буде 3.5).

Приклад 2.5. Скласти алгоритм для обчислення площі за формулою Герона $s=\sqrt{p\cdot(p-a)\cdot(p-b)\cdot(p-c)}$, де $p=\frac{a+b+c}{2}$, якщо: а) значення *a, b, c* задано коректно; б) треба перевірити, чи існує трикутник.

а) *К1*. Ввести числа a, b, c.

К2. Обчислити p=(a+b+c)/2.

К3. Обчислити s.

К4. Вивести s.

б) *К1*. Ввести числа a, b, c.

К2. Якщо a<=0, або b <=0, або c<=0, то видати повідомлення, що є від'ємне число. Перейти на К7.

К3. Якщо a+ b <= c, або a+ c <= b, або b+ c <= a, то видати повідомлення, що трикутник не можна побудувати. Перейти на К7. /* Якщо a+ b > c, і a+ c > b, і b+ c > a, то можна побудувати */

К4. Обчислити p=(a+b+c)/2.

К5. Обчислити s.

К6. Вивести s.

К7. Кінець.

Часом алгоритм можна подати одним виразом, використовуючи властивості операцій і функцій.

Приклад 2.6. Знайти третю з кінця цифру заданого цілого числа n.

Якщо в мові С ділити два цілих числа, то і результат буде цілим числом (дробова частина відкидається):

```
(n/100)%10.
```

Наприклад, 3456/100=34, 34%10=4 (остача від ділення).

Приклад 2.7. Знайти першу після крапки цифру заданого дробового числа a:

```
trunc(a*10)%10.
```

Наприклад, 7.89*10=78.9, функція `trunc` відкидає дробову частину числа і дає число 78, 78%10=8 (остача від ділення).

Приклад 2.8. Одержати число e (e=2,71828…).

У мові С:

```
e=exp(1).
```

Завдання. Описати словесно алгоритми.

1. Алгоритм Евкліда знаходження найбільшого спільного дільника (НСД) двох чисел : від більшого числа віднімаємо менше, доки числа не стануть рівними — це і є НСД.

2. Обчислити довжину кола і площу круга з радіусом R, використовуючи константу π .

3. Подати за допомогою логічних виразів такі твердження: а) два числа відрізняються рівно в 2 рази; б) два числа відрізняються менш, ніж у 2 рази; в) два числа відрізняються більш, ніж у 2 рази.

3. Типи даних, визначені стандартом С. У таблиці подано інформацію про основні типи даних:

Тип	Пам'ять (байтів) у Visual Studio	Діапазон значень у Visual Studio
char / signed char	1	–128 … 127 (-2^7 … 2^7-1)
unsigned char	1	0 … 255 (2^8-1)
short int / signed short int	2	–32768 … 32767 (-2^{15} … $2^{15}-1$)
unsigned short int	2	0 … 65535 ($2^{16}-1$)
int / signed int / long int / signed long int	4	–2147483648…2147483647 (-2^{31}… $2^{31}-1$)
unsigned int / unsigned long int	4	0 ... 4294967295 ($2^{32}-1$)
long long int	8	–9223372036854775808 … 9223372036854775807 (-2^{63} … $2^{63}-1$); *стандарт С99*
unsigned long long int	8	0 ... 18446744073709551615 (0 … $2^{64}-1$); *стандарт С99*
Float	4	9.9e–45 … 3.4e+38; точність — 6-7 десяткових цифр; мантиса — 24 біти
double / long double	8	4.94e–324 ... 1.7e+308; точність — 15-16 десяткових цифр; мантиса — 53 біти

Визначити розмір операнда в байтах дає можливість команда `sizeof`. Наприклад:

`sizeof 3` дає результат `4`;

`sizeof(3)` дає результат `4`;

`sizeof 3.` дає результат `8`;

`sizeof(3/2)` дає результат `4`;

`sizeof(int)` дає результат `4`;

`sizeof(unsigned char)` дає результат `1`.

Явне перетворення типів даних здійснюється операцією `(тип)`. Наприклад:

`3/2` дає результат `1`;

`(double)2/3` дає результат `1.5`;

`2/(double)3` дає результат `1.5`;

`(double)2/(double)3` дає результат `1.5`.

При використанні даних різних типів у виразах відбувається *неявне перетворення типів*. Перетворення типів операндів у виразах виконуються відповідно до таблиці при перегляді її згори вниз:

№	Початкові типи операндів	Типи операндів після перетворення	Пояснення
1.	float α	**double α**	**усі** операнди типу **float** перетворюються на **double**
2.	long double α будь-який тип β	long double α **long double β**	якщо один операнд має тип long double, то і другий операнд перетвориться на long double
3.	double α будь-який тип β	double α **double β**	якщо один операнд має тип double, то і другий операнд перетвориться на double
4.	char α	**int α**	**усі** операнди типу **char** або **short** перетворюються на **int**
5.	short int α	**int α**	
6.	unsigned char α	**unsigned int α**	**усі** операнди типу **unsigned char** або **unsigned short** перетворюються на **unsigned int**
7.	unsigned short int α	**unsigned int α**	
8.	unsigned long int α будь-який тип β	unsigned long int α **unsigned long int β**	якщо один операнд має тип unsigned long, то і другий перетвориться на unsigned long
9.	long int α будь-який тип β	long int α **long int β**	якщо один операнд має тип long, то і другий перетвориться на long
10.	unsigned int α будь-який тип β	unsigned int α **unsigned int α**	якщо один операнд має тип unsigned int, то і другий перетвориться на unsigned int

Завдання. Користуючись поданою вище інформацією, визначити, який тип буде мати результат і скільки пам'яті потрібно для його зберігання:

1. `a+1`, якщо змінна a має тип float.

2. `a+1`, якщо змінна a має тип double.

3. `'9'-'0'`; яке значення буде одержано?

4. `(int)3.14`; `(int)3.99`; які значення буде одержано?

5. `(double)1/4`, `(double)(1/4)`; які значення буде одержано?

Практичне заняття № 2 (*комп'ютерний практикум № 1*)

Основи роботи в інтегрованому середовищі Microsoft Visual Studio

Мета заняття: засвоїти основи роботи в ігтегрованому середовищі Microsoft Visual Studio (консольний режим C++); виконати в цьому середовищі просту програму.

Завдання. Виконати вказані нижче дії.

***1. Виконання основних операцій в інтегрованому середовищі Visual Studio* (*консольний режим C*++)**:

1). Початок роботи в середовищі Visual Studio (C++) — запустити програму ∞ Microsoft Visual Studio через Головне меню, чи кнопку Панелі задач, чи ярлик на Робочому столі.

2). Створення нового консольного проекту (консольної програми без віконного інтерфейсу користувача) для мови C — вибрати команду File – New – Project або натиснути комбінацію Ctrl+Shift+N. У вікні New project вибрати Visual C++ – Win32 Console Application, у полі Name вказати ім'я проекту, в полі Location — шлях до папки (можна скористатися кнопкою Browse), у якій розміститься проект, у полі Solution name — ім'я рішення (може збігатися з іменем проекту; рішення може включати кілька проектів), OK. Буде створено шаблон проекту. У вікні Win32 Application Wizard вибрати Application Settings чи натиснути кнопку Next, зняти опцію Precompiler header і відмітити опції Console application і Empty project, натиснути кнопку Finish.

Вибрати команду Project – Add New Item, чи у вікні Solution Explorer в контекстному меню (права клавіша) пункту Source Files вибрати команду Add – New Item, чи натиснути комбінацію клавіш Ctrl+Shift+A. У вікні Add New Item вибрати C++ File (.cpp), у полі Name вказати ім'я файла з розширенням .c (ім'я.c; якщо без розширення .c, то буде .cpp і дещо інший синтаксис коду) і натиснути кнопку Add.

Для налаштування проекту (вибору компілятора C) вибрати команду Project – Properties чи натиснути комбінацію Alt+F7. У вікні властивостей в полі Configuration вибрати All Configurations; розкрити Configuration Properties – C/C++ – Advanced – Compile as – Compile as C Code (/TC).

У вікні редактора коду на вкладці ім’я.c набирати текст програми.

3). Відкриття існуючого проекту — команда File – Open – Project/Solution, чи комбінація Ctrl+Shift+O, чи відповідний інструмент на стандартній панелі інструментів, відкрити папку проекту і в папці відкрити файл ім’я.sln. У вікні коду відкриється текст програми ім’я.c, який можна редагувати.

4). Робота з текстом програми. *Увага*: у тексті програми розрізняються малі й великі букви.

Редагування тексту здійснюється стандартними Windows-командами і комбінаціями клавіш.

Коментарі є однорядкові, які позначаються `// коментар`, і багаторядкові — /* коментар */. Коментарі можна набирати українською мовою.

Між лексичними одиницями можна вставляти будь-яку кількість пробілів.

Якщо ключові слова чи назви бібліотечних модулів набрано з помилками чи є невизначені змінні, то текст підкреслюється червоною хвилястою лінією; при підведенні до нього курсора відобразиться повідомлення про помилку.

Виділені фрагменти виділяються голубою заливкою. Якщо курсор встановити перед відкриваючою круглою чи фігурною дужкою, то ця дужка разом з відповідною закриваючою виділяються сірою заливкою.

Розділити вікно на дві частини — команда Window – Split.

Нумерація рядків тексту — команда Tools – Options – розкрити список Text Editor – розкрити список C/C++ – General – Line numbers – OK. Кількість пробілів структуризації тексту — Tools – … – Tabs – Tab size 2, Indent size 2 – OK.

5). Збереження коду програми – вибрати команду File – Save ім’я.c чи Ctrl+S (виконувати кожні 5-10 хвилин) або команду File – Save all чи Ctrl+Shift+S або відповідні кнопки на стандартній панелі інструментів.

6). Компіляція й виконання проекту. Для компіляції проекту вибрати команду Build – Compile чи комбінацію клавіш Ctrl+F7; для побудови рішення — Build – Build Solution, чи клавішу F7, чи відповідну кнопку на панелі побудови або Build – Build ім’я чи відповідну кнопку.

Є два способи виконання програми (з попередньою компіляцією): режим

налагодження — команда Debug – Start Debugging, чи кнопка зелений трикутник на стандартній панелі інструментів, чи клавіша F5 (для вставляння/прибирання точок зупинки при налагодженні — клацнути мишкою зліва від рядка програми на сірій смузі чи натиснути клавішу F9); без налагодження — Debug – Start Without Debugging чи комбінація клавіш Ctrl+F5.

7). Очистка проекту (пов'язано з тим, що при побудові рішення створюється ряд папок і багато файлів; після очистки файлів стає приблизно в 4 рази менше) — команда Build – Clean Solution або Build – Clean ім'я.

8). Налаштування параметрів вікна виведення — у контекстному меню заголовка вікна вибрати команду Properties і на вкладках Шрифт, Розташування, Кольори встановити потрібні значення відповідних параметрів.

9). Вихід з програми — команда File – Exit або комбінація Alt+F4.

Основні комбінації клавіш:

Ctrl+Shift+N — створення нового проекту;

Ctrl+Shift+O — відкривання наявного проекту;

Ctrl+Shift+A — створення пункту проекту;

Ctrl+S (при наборі тексту програми виконувати кожні 5-10 хвилин!) — збереження файла програми;

F5 — компіляція й виконання програми в режимі налагодження або продовження виконання програми після точки зупинки;

Ctrl+F5 — компіляція й виконання програми без налагодження;

Alt+F4 — закінчення роботи;

Ctrl+Shift+U — усі букви виділеного фрагменту зробити великими;

Ctrl+U — усі букви виділеного фрагменту зробити малими;

Ctrl+K, C — закоментовування виділених рядків програми;

Ctrl+K, U — розкоментовування виділених рядків програми;

Ctrl+K, F — структуризація виділеного фрагменту тексту.

2. Виконання програми. На диску D: створити папку lab1, де буде зберігатися проект цього завдання. Набрати текст програми (без коментарів), яка видає привітання:

```
/* Прізвище Ім’я По батькові, група Тх-хх, варіант № хх
 * Комп’ютерний практикум № 1. Основи роботи в середовищі
 * Visual Studio */
#include <stdio.h>   /* директива підмикання заголовочного
     файла стандартної бібліотеки stdio.h */
#include <stdlib.h>   /* підмикання стандартної бібліотеки
     stdlib.h (також можна вказати бібліотеку windows.h) */
#define N 255   /* директива означення макросу -- визначає
     константу з іменем N і значенням 255 */
int main()
{ // тіло програми
  char st[N];   /* оголошення рядкової змінної st */
  printf("\n\n\n\t\t\tPryvit! Tse -- C\n\n");   /* функція
     бібліотеки stdio.h для виведення у стандартний потік;
     символ керування \n переводить курсор на новий рядок;
     символ \t задає табулювання на 8 позицій */
  printf("Yak tebe zvaty?\n");
  gets(st);   /* функція бібліотеки stdio.h для введення рядка
     тексту */
  system("color 0B");   /* функція бібліотеки stdlib.h;
     параметр зміни кольору: 0 - колір фону (від 0 до F),
     B – колір шрифту */
  printf("Pryvit, %s!\a",st);   /* формат %s вказує, що
     виводиться значення рядкової змінної st,
     символ \a задає звуковий сигнал */
  getchar();   /* функція бібліотеки stdio.h для введення
     символа чи затримки екрану — чекає натискання клавіші
     Enter */
  system("color 0E"); printf("Bazhaiu uspikhiv!\n");
  getchar();
  return 0;   /* повертає код завершення функції main */
}
```

Виконати цю програму. Подати повідомлення різними кольорами.

Створити звіт, у якому в тексті налагодженої програми коментарі подати з використанням транслітерації (див. Додаток А).

Практичне заняття № 3 (*комп'ютерний практикум № 2*)
Оформлення звітів з робіт комп'ютерного практикуму засобами текстового редактора Word

Мета заняття: навчитися оформляти звіти з робіт комп'ютерного практикуму засобами текстового редактора Word (у звітах подавати блок-схему алгоритму, текст програми, скриншоти результатів роботи).

Завдання. У робочому документі виконати форматування тексту, створити формули і подати фрагмент блок-схеми. Набрати текст поданої програми, транслітеруючи коментарі символами англійського алфавіту (дотримуватися правил транслітерації — див. Додаток А). Виконати цю програму з кількома різними значеннями кодів символів. Створити звіт.

1. Вимоги до оформлення звітів. При оформленні звітів треба вміти:

1). Виконувати форматування тексту (нумерувати сторінки, вставляти колонтитули, стискати текст, форматувати абзаци — відступ першого рядка, вирівнювання, міжрядковий інтервал, не відривати від наступного).

2). Створювати і редагувати формули (*у звітах усі формули набирати в Microsoft Equation 3.0* — для полегшення роботи в Microsoft Equation ознайомтеся з довідкою щодо використання клавіатури: Довідка – Використання клавіатури).

3). Зображати блок-схеми (додавати текст в автофігури, збільшувати поля тексту в автофігурах, використовувати сітку, робити невидимими границі автофігур, використовувати порядок фрагментів об'єкта, вирівнювати автофігури).

4). Вставляти текст програми в звіт, подавати результати роботи програми (одержувати копії з екрана, інвертувати колір зображення, вставляти зображення в документ Word).

2. Виконання завдання. Наберіть текст поданої нижче програми, транслітеруючи коментарі символами англійського алфавіту (дотримуйтеся правил транслітерації — див. Додаток А). Виконайте цю програму з кількома різними значеннями кодів символів.

Текст програми:

```
/* Прізвище Ім'я По батькові, група ТХ-ХХ, варіант № хх
 * Комп'ютерний практикум №2. Оформлення звітів засобами
 * текстового редактора Word */
#include <stdio.h>   /* підмикання бібліотеки stdio.h; у
     програмі використано функції ХХХ,… (вкажіть, які) */
#include <stdlib.h>   /* підмикання stdlib.h; у програмі
     використано функції ХХХ,… ХХХ (вкажіть, які) */
/* За введеним користувачем числовим кодом від 32 до 255
     (кодування ASCII) виводить символ */
int main()
{
  int kod;   /* код символа */
  /* символи з кодами 0-31 є службовими; пробіл має код 32;
     код 13 вводиться при натисканні на клавішу Enter,
     27 — на Esc */
  printf("vvedit ASCII-kod symvola vid 32 do 255\n");
  scanf("%d",&kod);
  if (kod<32 || kod>255)
    printf("nevirnyi kod\n");
  else printf("symvol   %c\n",/*(char)*/kod);
  system("pause");  /* затримка екрана */
  /* або для затримки екрана випробуйте ще 2 варіанти:
     fseek(stdin,0,SEEK_END); getchar(); — очистка буфера вводу
     (stdin — стандартний потік вводу) і очікування Enter; або
     while (getchar()!='\n'); getchar(); */
  return 0;
}
```

Оформте звіт (зразок подано в Додатку Б) з комп'ютерного практикуму: вкажіть його мету, завдання, подайте текст програми, намалюйте блок-схему алгоритму програми, подайте не менше 3 зображень результату роботи, сформулюйте висновки, у верхніх колонтитулах подайте номер комп'ютерного практикуму, його назву, номер своєї групи, своє прізвище, номер варіанту, в нижні колонтитули вставте нумерацію сторінок.

Практичне заняття № 4

Арифметичні й логічні вирази. Введення-виведення даних. Умовні оператори

Мета заняття: закріпити лекційний матеріал і поглибити знання щодо запису арифметичних і логічних виразів; поглиблено розглянути можливості роботи з форматами й модифікаторами функцій printf і scanf; розробити алгоритми з використанням умовних операторів.

1. Арифметичні й логічні вирази. В арифметичних і логічних виразах можуть використовуватися змінні, константи, звернення до функцій.

Усі змінні перед використанням повинні бути оголошені (перед головною функцією — глобальні змінні, на початку тіла функції чи на початку блока у функції — локальні змінні):

тип список_змінних;

Тут тип означає один із стандартних або оголошених програмістом типів, а список змінних складається з одного або більше ідентифікаторів, розділених комами.

Приклад 1.1. Оголошення змінних:

```
int k, l, m; short int si;
unsigned int ui;
double length, suma;
```

Змінна при оголошенні може бути ініціалізована:

тип ім'я_змінної = константа;

Приклад 1.2. Ініціалізація змінних:

```
char ch='a';
int i=0,j,n=20; // ініціалізуються і й n —, j — ні.
```

Неініціалізовані локальні змінні до першого присвоєння мають довільне значення (сміття з пам'яті). Використання змінних, які не ініціалізовані і яким не присвоєно ніякого значення, недопустиме — програма аварійно закінчує роботу.

Константи можуть бути цілими й дійсними.

Цілі константи визначаються як числа (зі знаком чи без) без дробової частини. Цілі константи в С можуть бути:

десятковими — 547, -15;

вісьмірковими — записана цифрами 0-7 і перша цифра 0: 063, 0257; (*Увага!* Оскільки цифра 0 вказує на вісьміркове подання, то 0123 ($0123=123_8=83_{10}$) і 123 — це різні числа)

шіснадцятковими — записана цифрами 0-9, буквами A-F або a-f і починається з 0X чи 0x: 0X235, 0x307.

Цілі константи мають той мінімальний тип int чи long long int (4 чи 8 байтів), в діапазон значень якого їхні значення поміщаються.

Приклад 1.3. Записати цілі константи:

135 — `135` (десяткове подання), або `0207` (вісьміркове подання), або `0x87` (шістнадцяткове подання);

-10 — `-10` (десяткове подання), або `-012` (вісьміркове подання), або `-0xA` (шістнадцяткове подання).

Тип цілої (а також і дійсної) константи може визначити програміст: Можна явно вказати компілятору, який тип має ціла константа, використовуючи відповідні модифікатори-суфікси (регістр не має значення), приписувані після молодшої цифри константи:

L чи l — тип long,

LL чи ll (стандарт C99) тип long long,

U чи u — unsigned.

Приклад 1.4. Явно вказати тип цілої константи: `823L`, `042l`, `0x7D3Al`, `846U`, `054U`, `0x7A4Fu`, `235lu`, `0578ul`, `0x5ul`.

Дійсні константи — це десяткові дроби. Дійсні числа треба подавати *тільки десятковими цифрами* (у випадку вісьміркового чи шіснадцяткового подання результат буде непередбачуваним). Вони можуть бути записані в двох формах:

з фіксовано точкою (*знак ціла_частина.дробова_частина*): `-123.4567`, `561.9`, `.78` (відсутня ціла частина), `2.` (відсутня дробова частина);

з порядком (плаваючою точкою) — *мантиса***E***порядок* або *мантиса***e***порядок* : `0.1345e3` (134,5), `-1.34e-1` (-0,134), `-45E3` (-45000,0), `22E-2` (0,22), `.45e2` (45,0).

Усі дійсні константи, навіть дуже малі, мають тип double і вони займають 8 байтів пам'яті, якщо їхні значення виходять за діапазон даних типу double, то вони мають тип long double і займають 10 байтів пам'яті (у VS цей тип не відрізняється від типу double і значення займають 8 байтів пам'яті).

Приклад 1.5. Записати дійсні константи:

1,567 — `1.567`, `1.567e0`, `0.1567E1`, `1567.e-3`, `1567e-3`;

-234,71 — `-234.71`, `-234.71E0`, `-2.3471E+2`, `-.23471E3`;

$23{,}75 \cdot 10^{-3}$ — `23.75e-3`; $0{,}28 \cdot 10^{5}$ — `0.28e5`.

Можна явно вказати компілятору, який тип має дійсна константа, використовуючи відповідні модифікатори-суфікси (регістр не має значення): F чи f — тип float і їй виділяється 4 байти пам'яті; L чи l — тип long double і виділяється 10 байтів пам'яті незалежно від значення дійсної константи.

Приклад 1.6. Явно вказати тип дійсної константи: `5.6F`, `2.7e-10f`, `3.85L`, `8.074e+32L`.

Якщо при оголошенні змінної, значення якої ініціалізується, додати кваліфікатор const, то її значення в програмі змінювати не можна — вона стає іменованою константою вказаного типу:

const тип ім'я_змінної = константа;

Наприклад:

Приклад 1.7. Визначити іменовані константи:

```
const double PI=3.14159265358979;
const int n=10, k=-2;
```

Також константу можна визначити директивою препроцесора:

Приклад 1.8. Визначити іменовані константи:

```
#define PI=3.14159265358979;
```

```
#define N 5.
```

Константи, визначені командами препроцесора, як і оголошені в функції константи, можна використовувати у виразах.

Щоб у виразі використати стандартну функцію, треба директивою препроцесора підімкнути відповідну бібліотеку:

#include <*ім'я_заголовочного_файла_бібліотеки*.h>

При зверненні до функції треба вказати її ім'я, після якого в дужках подати параметри (якщо функція не має параметрів, то дужки залишаються порожніми). Інформацію про основні математичні функції і константи подано в Додатку В, п. 4.

Вираз у широкому розумінні — це і окрема константа, і окрема змінна, і окреме звернення до функції, це і побудовані за певними правилами конструкції, які містять константи, змінні, знаки операцій, круглі дужки. Вирази записують в рядок, враховуючи пріоритети операцій (див. Додаток В, п. 4), для зміни порядку операцій використовують дужки.

Арифметичні вирази — це числа, змінні, функції, з'єднані знаками арифметичних операцій і (при необхідності) круглими дужками. Результатом арифметичного виразу є число. Всі операції з арифметичними виразами виконуються зліва направо з врахуванням пріоритету операцій і дужок.

Приклад 1.9. Записати мовою С арифметичні вирази: $\frac{1-x^2}{1-\sqrt{x+2}}-5\log_6|x-4|$, $\frac{\ln(a+x)+1}{ctg(bx-2)}-4e^{-0,1|x+1|}$:

```
#include <math.h>
...
a=(1-x*x)/(1-sqrt(x+2))-5*log(fabs(x-4))/log(6.);
b=(log(a+x)+1)/cos(b*x-2)*sin(b*x-2)-4*exp(-0.1*fabs(x+1));
```

або

```
b=(log(a+x)+1)*tan(b*x-2))-4*exp(-0.1*fabs(x+1));
```

Логічний вираз — це вираз, результатом обчислення якого є логічне значення "істина" або "хибність". Значення "істина" або "хибність" є результа-

том операцій порівняння (<, <=, >, >=, ==, !=) або логічних операцій (!, &&, ||). При цьому в мові С “істина” подається як 1, а “хибність” як 0. Проте в мові С будь-яке числове значення, відмінне від нуля (включаючи й від’ємні числа), інтерпретується як “істина” і лише 0 як “хибність”.

Приклад 1.10. Визначити, яких результатів набувають логічні вирази:

`33>=8` — дає результат 1;

`33==8` — дає результат 0;

`5>8||8<10` — дає результат 1;

`5>8&&8<10` — дає результат 0;

`5>6>3` — дає результат 0 (5>6 дає результат 1 і 1>3 дає результат 0);

`5<6<3` — дає результат 1 (5<6 дає результат 1 і 1<3 дає результат 1).

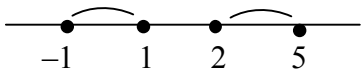
Приклад 1.11. Записати логічний вираз для визначення, чи є вказаний рік високосним:

```
r%4==0 && (r%100!=0 || r%400==0)
```

Приклад 1.12. Записати за допомогою логічних виразів обчислення максимуму й мінімуму з двох чисел:

```
max=(a>=b)*a+(b>a)*b;
min=(a<=b)*a+(b<a)*b;
```

Приклад 1.13. Записати за допомогою логічних виразів умови а) $x \in [-1;1] \vee x \in [2;5]$; б) x лежить поза цим відрізком:



а) `(abs(x)<=1) || (x>=2) && (x<=5)`

б) `(x<-1) || (x>1) && (x<5) || (x>5)`

або

`!(abs(x)<=1) || (x>=2) && (x<=5)`

Завдання. 1. Записати мовою С арифметичні вирази:

$$\sqrt{\frac{5x-tga}{x-2}}+\log|x-2|-\frac{3}{5}x^{0,7\pi};\quad \frac{5}{4}-\frac{4}{20}x-2e^{\sin x}.$$

2. Записати вирази для обох частин закону де Моргана $\overline{a \vee b} = \overline{a} \wedge \overline{b}$.

2. Введення-виведення даних. Для форматованого виведення і введення даних можна скористатися функціями printf і scanf (див. Додаток В, п. 6).

У програмах мовою С дуже важливим є правильне використання форматів ввенення й виведення при зверненні до функцій scanf і printf. Від цього залежить і правильність обчислень, у яких беруть участь вхідні дані, і правильність візуалізації результатів — програма може порахувати все правильно, а виведення за невідповідним до типу одержаного значення форматом відобразить зовсім інший результат. Крім того, оскільки формати виведення вказуються "на відстані" (у першому параметрі — рядку формату) від аргумента (змінної чи виразу у списку параметрів), то не завжди вистачає уважності швидко виявити помилку.

Приклад 2.1. Вплив формату виведення на відображення результату:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
  int n=7;
  printf(" 1. %d  \n",5);
  printf(" 2. %f  \n",5.);
  printf(" 3. %d  \n",5.);
  printf(" 4. %f  \n",5);
  printf(" 5. %c  \n",5.);
  printf(" 6. %c  \n",5);
  printf(" 7. %s  \n",5.);
  printf(" 8. %f\n",n);
  printf(" 9. %d\n",n);
  printf("10. %f\n",cos((double)0));
  printf("11. %d\n",cos((double)0));
  system("pause");
  return 0;
}
```

```
 1. 5
 2. 5.000000
 3. 0
 4. 0.000000
 5.
 6. ♣
 7. (null)
 8. 0.000000
 9. 7
10. 1.000000
11. 0
Press any key to continue .
```

Тут тільки у випадках 1,2, 9 і 10 виведено правильний результат.

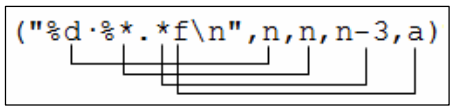
У функції printf для виведення даних типу float у формі з фіксованою точкою застосовують формат %f, для виведення даних типу double у формі з

фіксованою точкою можна застосовувати формати %f і %lf.

При виведенні даних у функції printf модифікатором формату, який вказує мінімальну ширину поля виведення, може бути ціле число чи зірочка * (вказує на відповідний за порядком параметр зі списку виведення).

Приклад 2.2. Застосування модифікатора * у функції printf:

```
int n=5;
double a=7.;
printf("%d %*.*f\n",n,n,n-3,a);
```



Цей оператор еквівалентний такому:

```
printf("%d %5.2f\n",n,a);
```

У функції scanf для введення даних типу float застосовують формат %f, для введення даних типу double застосовувати формат %lf (застосування формату %f призводить до введення неправильного значення).

Приклад 2.4. Написати програму обчислення довжини кола і площі круга, у якій з клавіатури вводиться величина радіуса і на екран виводяться обчислені значення в форматі з фіксованою точкою.

```
#include <stdio.h>
#include <stdlib.h>
#define _USE_MATH_DEFINES
#include <math.h>
int main()
{ // Обчислення довжини кола і площі круга
  double r,l,s;
  printf("Vvedit r\n");
  scanf("%lf",&r);
  l=2*M_PI*r;
  s=M_PI*r*r;
  printf("Rezultat:\n\tl=%0.3f  s=%0.3f\n",l,s);
  system("pause");
  return 0;
}
```

```
Vvedit r
5
Rezultat:
        l=31.416  s=78.540
Press any key to continue . . .
```

```
Vvedit r
1
Rezultat:
        l=6.283  s=3.142
Press any key to continue . . .
```

При введенні рядкових даних, щоб не зчитувалося більше символів, ніж відведено пам'яті під рядок символів, треба у форматі задати модифікатор — кількість символів. Наприклад, якщо є рядок символів `char st[10]`, то, щоб не зчитувалося більше, ніж 9 символів у st, треба вказати формат %9s (буде зчитано або менше дев'яти символів — до першого пробільного, або рівно дев'ять, якщо пробільний символ розміщується далі).

У функції scanf модифікатор *, який ставиться між знаком % і кодом формату, вказує на пропуск введеної з клавіатури інформації.

Приклад 2.3. Застосування модифікатора * у функції scanf:

```
#include <stdio.h>
#include <stdlib.h>
int main() { // використання модифікатора *
  int n;
  printf("Vvedit symvol i tsile chyslo:\n");
  scanf("%*c%d",&n);
  printf("Zchytana informatsia:\n\t%d\n",n);
  system("pause");
}
```

```
Vvedit symvol i tsile chyslo:
A    1234
Zchytana informatsia:
        1234
Press any key to continue . . .
```

```
Vvedit symvol i tsile chyslo:
1234
Zchytana informatsia:
        234
Press any key to continue . . .
```

```
Vvedit symvol i tsile chyslo:
 1234
Zchytana informatsia:
        1234
Press any key to continue . . .
```

У мові програмування С часом виникає проблема з введенням даних функцією scanf: коли натискати кдавішу Enter, то символ переходу на новий ря-

док зберігається в буфері вводу і присвоюється при наступному виклику функції scanf символьній змінній. Тому зайвого символу можна позбутися, використовуючи в форматі модифікатор *.

Якщо при використанні функції scanf з клавіатури вводити “зайву інформацію”, то вона також має бути подана в рядку формату. Наприклад, якщо при введенні дати використовувати формат з крапкою *день.місяць.рік*, то в рядку форматів теж треба подати крапки: `scanf("%d.%d.%d",&d,&m,&r);`; якщо з клавіатури вводити інформацію виду `i=3, j=5`, то треба використати такий оператор: `scanf("i=%d, j=%d",&a,&b);`.

Завдання. Навести приклади використання:

1. Форматів %e і %g у функціях printf і scanf.

2. Модифікаторів -, +, # у функції printf.

3. Умовні оператори. Використання умовних операторів дає можливість створювати програми, які реалізують розгалуження, тобто аналізують різні ситуації і відповідно до цього виконують певну обробку даних.

Приклад 3.1. Написати програму для розв’язування квадратного рівняння $ax^2+bx+c=0$, скориставшись формулами $D=b^2-4ac$, $x_{1,2}=\frac{-b\pm\sqrt{D}}{2a}$. Рівняння має розв’язок, якщо $D\geq 0$, і не має розв’язку при $D<0$.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{ // Розв’язування квадратного рівняння
  double a,b,c,D,x_1,x_2;
  system("chcp 1251 & cls");   /* установка кодової
    сторінки win-cp 1251 (кодування ANSI) в потоки вводу
    і виводу (change codepage)*/
  printf("Розв'язування рівняння ax2+bx+c=0,c\n");
  printf("Введіть a,b,c\n\t");
```

```
  scanf("%lf%lf%lf",&a,&b,&c);
  D=b*b-4*a*c;
  if (D<0)
    printf("Рівняння не має розв'язку");
  else {
    x_1=(-b-sqrt(D))/(2*a);
    x_2=(-b+sqrt(D))/(2*a);
    printf("Розв'язок:\n");
    printf("\tx1=%0.3f;  x2=%0.3f\n",x_1,x_2);
  }
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Розв'язування рівняння ax2+bx+c=0
Введіть a,b,c
        1 2 3
Рівняння не має розв'язку
```

```
Розв'язування рівняння ax2+bx+c=0
Введіть a,b,c
        1 2 1
Розв'язок:
        x1=-1.000;  x2=-1.000
```

```
Розв'язування рівняння ax2+bx+c=0
Введіть a,b,c
        1 -5 6
Розв'язок:
        x1=2.000;  x2=3.000
```

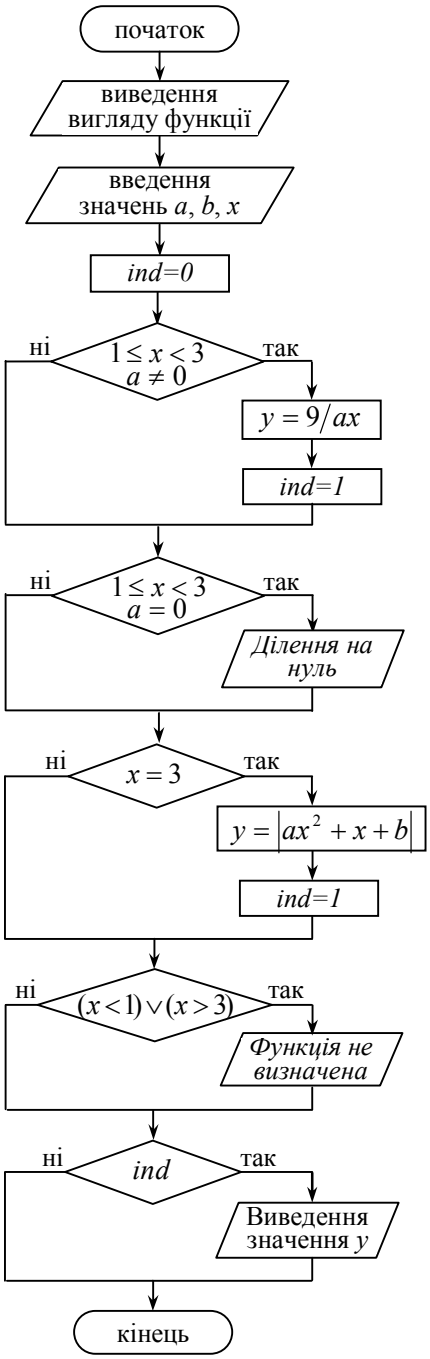
Приклад 3.2. Розробити алгоритм з послідовною перевіркою умов, подати його блок-схемою й написати програму обчислення значення заданої логічною залежністю функції

$$f(x)=\begin{cases}\dfrac{9}{ax} & \text{при } x\in[1,3),\\ \left|ax^2+x+b\right| & \text{при } x=3\end{cases}$$

при довільних значеннях параметрів a і b і незалежної змінної x. В алгоритмі передбачити перевірку, чи немає ділення на нуль. Вважається, що функція визначена на заданому проміжку, а поза ним не визначена. Видачу повідомлень про помилки здійснювати зразу ж при їхньому виявленні, а видачу результату обчислення значення функції $f(x)$ — у кінці програми.

Щоб реалізувати таку видачу повідомлень, треба використати робочу змінну-індикатор (назвемо її, наприклад, ind), яка вказує на те, чи значення функції було обчислене.

Блок-схема алгоритму:



Алгоритм з використанням послідовних умовних операторів має просту структуру, він легкий для сприйняття й реалізації:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
/* Обчислення значення функції, заданої логічною залеж-
 * ністю. Використання послідовних умовних операторів. */
  float x,   // значення аргумента
```

```
        y,   // значення функції
        a,b; // параметри функції
  int ind;   // індикатор друку значення функції
  system("chcp 1251 & cls");
  /* Виведення вигляду функції на екран */
  printf("f(x)=9/(ax)          при x [1,3)\n");
  printf("     abs(ax^2+x+b)   при x=3\n");
  /* Введення вхідних даних */
  printf ("\nВведіть значення a ");
  scanf("%f",&a);
  printf ("Введіть значення b ");
  scanf("%f",&b);
  printf ("Введіть значення x ");
  scanf("%f",&x);
  ind=0;
  /* Обчислення значення функції */
  /* Перший проміжок. ОДЗ не порушено */
  if ((x>=1) && (x<3) && (a!=0))
    { y=9/(a*x); ind=1; }
  /* Перший проміжок. ОДЗ порушено */
  if ((x>=1) && (x<3) && (a==0))
    printf("Ділення на нуль");
  /* Другий проміжок. Особливостей нема */
  if (x==3)
    { y=fabs(a*x*x+x+b); ind=1; }
  /* Значення x задано поза проміжком */
  if ((x<1) || (x>3))
    printf("Функція не визначена");
  /* Друк результату обчислення */
  if (ind)
    printf("f(%4.2f)=%6.3f", x, y);
  /* Затримка вікна виведення */
  printf("\n\n"); system("pause");
  return 0;
}
```

```
f(x)=9/(ax)          при x [1,3)
     abs(ax^2+x+b)   при x=3

Введіть значення a 2
Введіть значення b -4
Введіть значення x 3
f(3.00)=17.000
```

Завдання. Написати програму обчислення площі трикутника за формулою Герона з перевіркою, чи існує трикутник.

Практичне заняття № 5 (*комп'ютерний практикум № 3*)
Програмування найпростіших обчислювальних алгоритмів лінійної структури

Мета заняття: набути практичних навичок складати й програмувати мовою С найпростіші обчислювальні алгоритми лінійної структури, а також визначати константи, використовувати функції стандартної математичної бібліотеки math.h (див. Додаток В, п. 5), програмувати арифметичні вирази здійснювати форматоване введення-виведення даних за допомогою функцій printf і scanf з бібліотеки stdio.h.

Завдання. Розробити алгоритм, відповідно до якого треба послідовно виконати такі дії:

— вивести на екран вигляд заданих функцій f_1, f_2, f_3;

— ввести значення параметра *a* і змінної *x*;

— трьом змінним *y*, *z*, *t* дійсного типу послідовно присвоїти значення трьох заданих функцій; при цьому константи для першої функції f_1 визначити в декларативній частині, функцію f_2 записати з мінімумом операцій (наприклад, число $\frac{1}{4}$ подати як 0.25), ОДЗ не перевіряти:

— вивести обчислені значення на екран (f1=значення_y) у форматі з фіксованою точкою;

— виконати переприсвоєння значень змінних ($y \leftarrow z$, $z \leftarrow t$, $t \leftarrow y$);

— знову вивести на екран значення змінних (y=значення_y) у форматі з фіксованою точкою.

Розробити блок-схему алгоритму і запрограмувати його мовою С. Текст програми подати в структурованому вигляді (з відступами 2 пробіли). Коментарі в програмі обов'язкові (17-25% від усього коду програми).

За алгоритмом провести розрахунки не менш ніж з трьома різними наборами значень параметра *a* і змінної *x* (значення *a* і *x* підбирати так, щоб вони задовольняли ОДЗ). Правильність обчислень перевірити в Excel.

Варіанти завдань

1. $\frac{2x-a}{3-\sqrt{x-1}}-3\cos|x|,\ \frac{3}{2}-\frac{3}{10}x+7e^{-2tgx},\ \frac{4}{5}x^{6,8}.$

2. $\sqrt{\frac{x+2}{2-ax}}+4|\log_2 x-1|,\ \frac{5}{4}+\frac{3}{10}x-2e^{3tgx+5},\ \frac{2}{5}x^{2,6}.$

3. $\frac{2x+a}{\sqrt{x}-3}-\log_3|x-4|,\ \frac{1}{2}+\frac{3}{10}x-e^{-tgx+2},\ \frac{3}{4}x^{3,1}.$

4. $\frac{\sqrt{\ln x+2}}{4x-a}+\sin|x+4|\ ,\ \frac{3}{4}-\frac{4}{5}x+e^{ctg(x+2)},\ \frac{3}{5}x^{-0,2}.$

5. $\sqrt{\frac{x+a}{5x-3}}+|\sin x|,\ \frac{1}{4}+\frac{7}{5}x+4e^{tg(x-6)},\ \frac{3}{5}x^{3,4}.$

6. $\frac{5x-a}{1-\sqrt{x+1}}-ctg|x-5|,\ \frac{3}{4}x+\frac{6}{5}-7e^{x-8},\ \frac{1}{4}x^{4,2}.$

7. $\frac{\sqrt{\ln x+5}}{3x+a}+\log_3|x+5|,\ \frac{1}{8}-\frac{8}{10}x+e^{tg3x},\ \frac{1}{8}x^{-4,4}.$

8. $\frac{7x-a}{3+\log_2 x}+\sqrt{|x+3|},\ \frac{1}{8}x-\frac{3}{10}-e^{ctgx+1},\ \frac{5}{8}x^{-4,1}.$

9. $\frac{\sqrt{\ln(x+1)}}{5x-a}+\sin|x+5|,\ \frac{7}{2}x-\frac{2}{5}+2e^{tgx},\ \frac{3}{8}x^{-3,8}.$

10. $\sqrt{\frac{5x-a}{x^2-1}}+\cos|x-1|,\ \frac{5}{2}x+\frac{7}{5}-e^{4tgx},\ \frac{3}{4}x^{1,9}.$

11. $\sqrt{\frac{x-1}{7x+a}}+|x-tgx|,\ \frac{1}{2}x+\frac{7}{5}+2e^{-x+5},\ \frac{1}{2}x^{-0,4}.$

12. $\frac{\sqrt{3x+5}}{x-a}+\cos|x+5|,\ \frac{1}{10}x+\frac{7}{4}+7e^{ctg(x-3)},\ \frac{3}{4}x^{-3,11}.$

13. $\sqrt{\frac{x-3}{3x+a}}-5\sin|x|,\ \frac{1}{4}x-\frac{1}{2}-e^{tgx+7},\ \frac{5}{8}x^{6,2}.$

14. $\frac{x-3}{1-\sqrt{x+a}}-5\sin|x|\ ,\ \frac{1}{2}+\frac{8}{5}x-5e^{ctg(x+1)},\ \frac{1}{4}x^{4,6}.$

15. $\sqrt{\frac{x-a}{3x+1}}+|\sin x-2|,\ \frac{7}{2}-\frac{3}{10}x+e^{ctgx},\ \frac{2}{5}x^{3,8}.$

16. $\frac{\sqrt{x+2}}{2-ax}+|\cos x+5|,\ \frac{3}{4}+\frac{7}{10}x+7e^{2-tgx},\ \frac{7}{8}x^{-0,26}$

17. $\sqrt{\frac{x-a}{2x+1}}+|\sin x-3|,\ \frac{1}{4}x+\frac{11}{5}+e^{tgx-5},\ \frac{1}{5}x^{3,9}$

18. $\sqrt{\frac{5x+a}{x-3}}+\sin|x-5|,\ \frac{3}{8}-\frac{7}{5}x-6e^{ctgx-1},\ \frac{2}{5}x^{4,9}$

19. $\frac{\sqrt{x-2}}{3-ax}+|\cos x-4|,\ \frac{1}{4}+\frac{3}{10}x+7e^{1-tgx},\ \frac{7}{8}x^{-0,12}$

20. $\sqrt{\frac{x+1}{2-ax}}-\log_5|x-1|,\ \frac{3}{4}-\frac{7}{10}x+2e^{-tgx},\ \frac{3}{5}x^{-5,3}$

21. $\sqrt{\frac{x+a}{2x+5}}+|\sin x-2|,\ \frac{3}{2}-\frac{7}{10}x+e^{ctgx},\ \frac{5}{8}x^{4,8}$

22. $\sqrt{\frac{\ln x+1}{5x+a}}+|x+5tgx|,\ \frac{1}{4}-\frac{4}{10}x-e^{3x+5},\ \frac{4}{5}x^{-0,3}$

23. $\frac{3x-2a}{\sqrt{x}+5}-\ln|x+2|,\ \frac{5}{2}-\frac{7}{10}x-e^{1-ctgx},\ \frac{4}{5}x^{-2,1}$

24. $\frac{x-3a}{2+\sqrt{x-3}}-tg|x-2|,\ \frac{1}{4}x+\frac{7}{5}-2e^{x-3},\ \frac{7}{10}x^{4,2}$

25. $\sqrt{\frac{2x+a}{4x-3}}+|\cos x|,\ \frac{3}{4}+\frac{9}{5}x+4e^{tg(x-6)},\ \frac{3}{5}x^{3,3}$

26. $\frac{\sqrt{x+2}}{a-3x}+tg|\cos x+3|,\ \frac{1}{10}x+\frac{5}{4}-e^{2+x},\ \frac{3}{8}x^{-5,4}$

27. $\sqrt{\frac{4x-a}{x-2}}+\log|x-2|,\ \frac{5}{4}-\frac{4}{20}x-2e^{\sin x},\ \frac{4}{5}x^{5,4}$

28. $\frac{\sqrt{\ln x+3}}{3x+a-4}+ctg|x-4|,\ \frac{7}{4}-\frac{4}{5}x+xe^{-3tgx},\ \frac{7}{8}x^{0,7}$

29. $\frac{\sqrt{x+3}}{2a-x}+\log_3|\cos x+3|,\ \frac{8}{10}x+\frac{7}{4}-e^{1+x},\ \frac{3}{8}x^{-4,4}$

30. $\frac{x-a}{1-7\sqrt{x}}-\log_5|x-1|,\ \frac{9}{2}x+\frac{3}{10}+e^{ctg\,2x},\ \frac{1}{2}x^{6,3}$

Практичне заняття № 6 (*комп'ютерний практикум № 4*)

Розробка і реалізація алгоритмів розгалужених процесів з використанням послідовних умовних операторів

Мета заняття: набути практичних навичок складати алгоритми з послідовною перевіркою умов і програмувати їх засобами мови С, використовуючи неповну форму умовного оператора if.

Завдання. Розробити алгоритм з послідовною перевіркою умов, намалювати блок-схему й написати програму обчислення значення заданої логічною залежністю функції $f(x)$ при довільних значеннях параметрів a і b і незалежної змінної x.

В алгоритмі передбачити перевірку, чи немає ділення на нуль, чи підкореневий вираз невід'ємний, чи аргумент функції логарифма набуває додатних значень тощо. Вважається, що функція визначена на заданому проміжку, а поза ним не визначена. Видачу повідомлень про помилки здійснювати зразу ж при їхньому виявленні, а видачу результату обчислення значення функції $f(x)$ — у кінці програми. Коментарі в програмі обов'язкові (17-25%).

За алгоритмом провести розрахунки не менш ніж з п'ятьма різними наборами вхідних даних, які передбачають виконання різних гілок алгоритму.

Варіанти завдань

1. $f(x)=\begin{cases}\sin\ 3x & \text{при } x\in[0,5),\\ 2e^{ax-1}+1 & \text{при } x=5,\\ (bx-a)^{-1} & \text{при } x\in(5,12].\end{cases}$

2. $f(x)=\begin{cases}\sqrt{ax+2} & \text{при } x\in(-3,2),\\ |\cos 2x| & \text{при } x=2,\\ 3x^2+bx+1 & \text{при } x\in(2,+\infty).\end{cases}$

3. $f(x)=\begin{cases}bx^2+x-6 & \text{при } x\in[0,1),\\ \left(\sqrt{ax+b}\right)^{-1} & \text{при } x=1,\\ \cos\ x & \text{при } x\in(1,9).\end{cases}$

4. $f(x)=\begin{cases}\sin 5x & \text{при } x\in[-0,7),\\ x^2-bx & \text{при } x=7,\\ \ln(ax-8) & \text{при } x\in(7,10).\end{cases}$

5. $f(x)=\begin{cases}|ax+7| & \text{при } x\in(-\infty,9),\\ 5\sqrt{bx+1} & \text{при } x\in[9,15),\\ \sin x & \text{при } x=15.\end{cases}$

6. $f(x)=\begin{cases}\sin\ x & \text{при } x\in(-\infty,-2),\\ 2e^{ax}-1 & \text{при } x=-2,\\ (bx+2a)^{-2} & \text{при } x\in(-2,8).\end{cases}$

7. $f(x)=\begin{cases}\sqrt{5x^2-a} & \text{при } x\in(-\infty,3),\\ bx+8 & \text{при } x=3,\\ \cos x & \text{при } x\in(3,11).\end{cases}$

8. $f(x)=\begin{cases}2x+7 & \text{при } x\in[0,5),\\ 3e^{ax} & \text{при } x=5,\\ ln(bx+a) & \text{при } x\in(5,8).\end{cases}$

9. $f(x)=\begin{cases}\sin x & \text{при } x\in[-1,6),\\ 3x^2+bx-3 & \text{при } x=6,\\ ln(bx+a) & \text{при } x\in(6,+\infty),\end{cases}$

10. $f(x)=\begin{cases}\sqrt{x^2-a} & \text{при } x\in(-3,3),\\ 5x^3-bx+3 & \text{при } x=3,\\ \cos x & \text{при } x\in(3,8).\end{cases}$

11. $f(x)=\begin{cases}\sin x & \text{при } x\in[0,7),\\ 3e^{ax+1}-1 & \text{при } x=7,\\ (bx-a)^{-1} & \text{при } x\in(7,11).\end{cases}$

12. $f(x)=\begin{cases}\sqrt{ax-3} & \text{при } x\in(-8,3),\\ |x^2-bx+7| & \text{при } x=3,\\ \cos(x+1) & \text{при } x\in(3,+\infty).\end{cases}$

13. $f(x)=\begin{cases}\cos x & \text{при } x\in[0,5),\\ \left(\sqrt{ax-b}\right)^{-1} & \text{при } x=5,\\ bx^2+3x-2 & \text{при } x\in(5,8).\end{cases}$

14. $f(x)=\begin{cases}\sin x & \text{при } x\in[-1,6),\\ 3x^2+bx & \text{при } x=6,\\ ln(ax-3b) & \text{при } x\in(6,8).\end{cases}$

15. $f(x)=\begin{cases}\sqrt{3x^2-a} & \text{при } x\in(-3,3),\\ -bx+3 & \text{при } x=3,\\ \cos(x-4) & \text{при } x\in(3,8).\end{cases}$

16. $f(x)=\begin{cases}\cos x & \text{при } x\in(-\infty,4),\\ x^3-bx+1 & \text{при } x=4,\\ \sqrt{ax} & \text{при } x\in(4,7).\end{cases}$

17. $f(x)=\begin{cases}\sin x & \text{при } x\in[-1,3),\\ \sqrt{ax-2} & \text{при } x=3,\\ bx+a & \text{при } x\in(3,5).\end{cases}$

18. $f(x)=\begin{cases}\cos x & \text{при } x\in[0,4),\\ 3x^2+bx & \text{при } x=4,\\ ln(ax+b) & \text{при } x\in(4,9).\end{cases}$

19. $f(x)=\begin{cases}\sin x & \text{при } x\in[0,2),\\ 2e^{ax} & \text{при } x=2,\\ (bx+a)^{-1} & \text{при } x\in(2,8).\end{cases}$

20. $f(x)=\begin{cases}|ax+x^2| & \text{при } x\in(-\infty,4),\\ \sqrt{bx-1}+3 & \text{при } x=4,\\ \cos x & \text{при } x\in(4,10].\end{cases}$

21. $f(x)=\begin{cases}\sqrt{x+b} & \text{при } x\in[0,11),\\ ax^2-3x+14 & \text{при } x\in[11,15),\\ e^{-ax} & \text{при } x=15.\end{cases}$

22. $f(x)=\begin{cases}\sqrt{x^3-a} & \text{при } x\in(-\infty,7),\\ 3x^2-x+b & \text{при } x=7,\\ \sin 2x & \text{при } x\in(7,8).\end{cases}$

23. $f(x)=\begin{cases}\sqrt{bx+3} & \text{при } x\in(-2,10),\\ |x^2-abx+7| & \text{при } x=10,\\ \cos(x-1) & \text{при } x\in(10,+\infty).\end{cases}$

24. $f(x)=\begin{cases}\ln(ax+3) & \text{при } x\in(-2,5),\\ x^2-bx+7 & \text{при } x=5,\\ \cos(x+1) & \text{при } x\in(5,+\infty).\end{cases}$

25. $f(x)=\begin{cases}\sqrt{x^3+a} & \text{при } x\in(-\infty,7),\\ -bx-3 & \text{при } x=7,\\ \sin x & \text{при } x\in(7,10).\end{cases}$

26. $f(x)=\begin{cases}\cos x & \text{при } x\in[0,9),\\ 3e^{ax}+1 & \text{при } x=9,\\ ln(bx-1) & \text{при } x\in(9,11).\end{cases}$

27. $f(x)=\begin{cases}\sin(x+1) & \text{при } x\in[0,1),\\ ax^2+2bx-4 & \text{при } x\in[1,7),\\ (ax+b)^{-1} & \text{при } x=7.\end{cases}$

28. $f(x)=\begin{cases}|ax+b| & \text{при } x\in(-\infty,8)\\ 5\sqrt{bx+3} & \text{при } x=8,\\ \cos x & \text{при } x\in(8,10].\end{cases}$

29. $f(x)=\begin{cases}\sin\ x & \text{при } x\in[0,5),\\ 3e^{ax}+1 & \text{при } x=5,\\ (bx-10)^{-1} & \text{при } x\in(5,11).\end{cases}$

30. $f(x)=\begin{cases}\cos x & \text{при } x\in(-2,4],\\ \sqrt{ax+b} & \text{при } x\in(4,9),\\ x^3-bx+3 & \text{при } x=9.\end{cases}$

Практичне заняття № 7
Умовні оператори. Проектування алгоритмів і програм циклічної структури: цикл з параметром, цикл з передумовою

Мета заняття: закріпити лекційний матеріал і вдосконалити навички щодо використання й програмування вкладених умовних операторів, операторів вибору, циклів з параметром і циклів з передумовою.

1. Вкладені умовні оператори. Використання вкладених умовних операторів дає можливість компактно програмувати складну логіку обчислень.

Приклад 1.1. Запрограмувати функцію

$$sign(x)=\begin{cases}-1, & \text{якщо } x<0;\\ 0, & \text{якщо } x=0;\\ 1, & \text{якщо } x>0,\end{cases}$$

використовуючи вкладені умови:

```
if (x<0) sign=-1;
else
  if (x) sign=1;
  else sign=0;
```

Приклад 1.2. За трьома заданими числами дослідити, чи може існувати трикутник з такими сторонами, і якщо так, то визначити, який він: різнобічний, рівнобедрений чи рівнобічний.

Подамо не деталізований словесний опис *алгоритму розв'язування задачі*:

К1. З'ясувати, чи існує трикутник.

К2. Перевірити, чи жодна зі сторін не дорівнює інший. Якщо це так, то трикутник різнобічний і програма закінчить роботу. Якщо це не так, то програма перейде до наступної перевірки.

К3. Перевірити, чи всі сторони рівні. Якщо це так, то трикутник рівнобічний. Якщо це не так, то залишається тільки один варіант — трикутник рівнобедрений.

Програмно алгоритм можна реалізувати так:

```
#include <stdio.h>
#include <stdlib.h>
int main() { // Визначення типу трикутника
  int a,b,c;
  system("chcp 1251 & cls");
  printf("Визначення типу трикутника\n");
  printf("Введіть a, b, c \n\t");
  scanf("%d %d %d",&a,&b,&c);
  if (a+b <= c || a+c <= b || b+c <= a)
    printf("Трикутник не існує");
  else
    if (a != b && a != c && b != c)
      printf("Різнобічний");
    else
      if (a == b && b == c)
        printf("Рівнобічний");
      else
        printf("Рівнобедрений");
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Визначення типу трикутника
Введіть a, b, c
        1 2 1
Трикутник не існує
```

```
Визначення типу трикутника
Введіть a, b, c
        3 4 5
Різнобічний
```

```
Визначення типу трикутника
Введіть a, b, c
        2 2 2
Рівнобічний
```

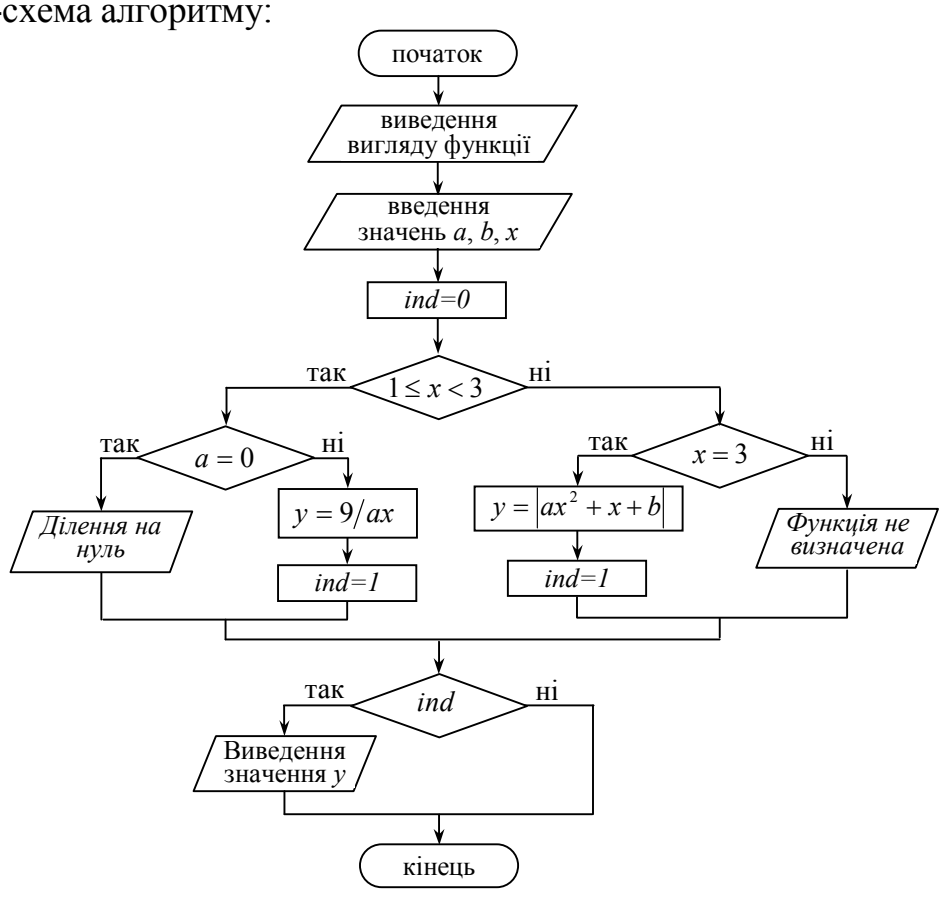
```
Визначення типу трикутника
Введіть a, b, c
        2 3 2
Рівнобедрений
```

Приклад 1.3. Розробити алгоритм з вкладеною перевіркою умов (порівняння в умовах не повторювати), подати його блок-схемою й написати програму обчислення значення заданої логічною залежністю функції

$$f(x)=\begin{cases}\dfrac{9}{ax} & \text{при } x\in[1,3),\\ \left|ax^2+x+b\right| & \text{при } x=3\end{cases}$$

при довільних значеннях параметрів a і b і незалежної змінної x. В алгоритмі передбачити перевірку, чи немає ділення на нуль. Вважається, що функція визначена на заданому проміжку, а поза ним не визначена. Видачу повідомлень про помилки здійснювати зразу ж при їхньому виявленні, а видачу результату обчислення значення функції $f(x)$ — у кінці програми (щоб реалізувати таку видачу повідомлень, треба використати робочу змінну-індикатор, яка вказує на те, чи значення функції було обчислене).

Блок-схема алгоритму:



Розроблений раніше алгоритм з використанням послідовних умовних операторів має простішу структуру і легший для сприйняття й реалізації, але він менш ефективний, ніж алгоритм з вкладеними умовами. Цей алго-

ритм дає можливість не повторювати порівняння і за рахунок цього виконується швидше. Крім того, при використанні вкладених умовних операторів у середньому (при багаторазових обчисленнях) виконується лише половина порівнянь.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
/* Обчислення значення функції, заданої логічною залеж-
 * ністю. Використання вкладених умовних операторів. */
  float x,   // значення аргумента
        y,   // значення функції
        a,b; // параметри функції
  int ind;   // індикатор друку значення функції
  system("chcp 1251 & cls");
  /* Виведення вигляду функції на екран */
  printf("f(x)=9/(ax)          при x [1,3)\n");
  printf("     abs(ax^2+x+b)   при x=3\n");
  /* Введення вхідних даних */
  printf ("\nВведіть значення a ");
  scanf("%f",&a);
  printf ("Введіть значення b ");
  scanf("%f",&b);
  printf ("Введіть значення x ");
  scanf("%f",&x);
  ind=0;
  /* Обчислення значення функції */
  if ((x>=1) && (x<3))   // Перший проміжок
    if (a==0) printf("Ділення на нуль");   // ОДЗ порушено
    else { y=9/(a*x); ind=1; }   // ОДЗ не порушено
  else
    if (x==3)   // Другий проміжок - особливостей нема
      { y=fabs(a*x*x+x+b); ind=1; }
    else   // Значення x задано поза проміжком
      printf("Функція не визначена");
```

```
  /* Друк результату обчислення */
  if (ind)
    printf("f(%4.2f)=%6.3f", x, y);
  /* Затримка вікна виведення */
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
f(x)=9/(ax)          при x [1,3)
     abs(ax^2+x+b)   при x=3

Введіть значення a 0
Введіть значення b 1
Введіть значення x 2
Ділення на нуль
```

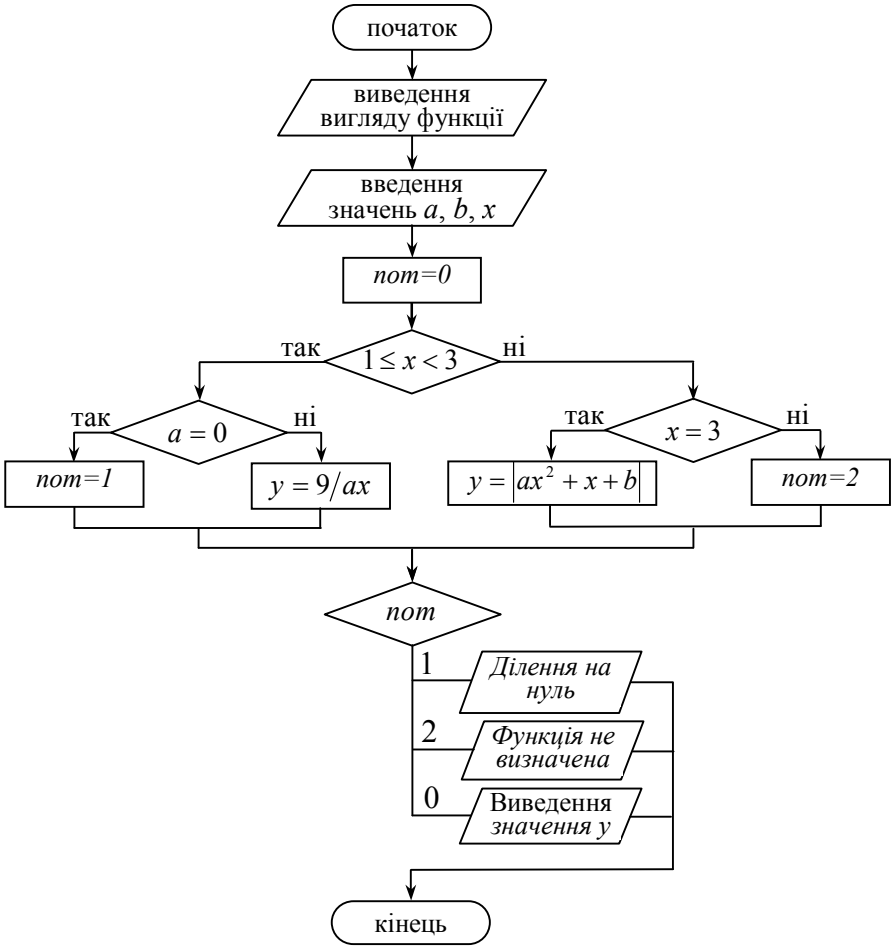
Завдання. Написати програму генерації двох випадкових чисел з проміжка -50...50 і визначення того, чи числа рівні, чи перше менше, чи друге менше. Для генерації випадкових чисел використати функцію `rand()` (бібліотека stdlib.h), для ініціалізації генерації — функцію `srand(1)` (бібліотека stdlib.h) чи `srand(time(0))` (бібліотеки stdlib.h і time.h). У Visual Studio максимальне згенероване число визначається константою RAND_MAX = 32767, мінімальне — 0. Генерація цілого числа від min до max включно — `rand()%(max-min+1)+min`, від 0 до 10 — `rand()%11` ; чисел –1, 0 чи 1 — `rand()%3-1`. Для генерації дійсних чисел застосовують формулу `(float)rand()/RAND_MAX*(max-min)+min`.

2. Оператор вибору. Умовний оператор при виконанні програми дає можливість вибрати одну із двох дій залежно від значення виразу. Узагальненням умовного оператора є оператор вибору (багатоваріантного розгалуження). Використання оператора вибору дає можливість реалізувати вибір однієї з кількох альтернатив.

Приклад 2.1. За умовою поданого в попередньому пункті прикладу 1.3 розробити алгоритм, відповідно до якого програма структуризується на блоки: блок введення даних, блок обробки і блок виведення результатів (вся видача повідомлень про помилки і результат обчислення здійснюється в кінці програми; щоб реалізувати таку видачу, треба використати робочу змінну, яка вказує номер повідомлення).

Відокремлення блоку обчислень від блоку видачі результатів можна ефективно реалізувати, якщо використати оператор вибору.

Блок-схема алгоритму:



Текст програми:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
/* Обчислення значення функції, заданої логічною залеж-
 * ністю. Використання вкладених умовних операторів. */
  float x,   // значення аргумента
        y,   // значення функції
        a,b; // параметри функції
  int nom;   // номер повідомлення (0 – обчислено значення)
  system("chcp 1251 & cls");
  /* Виведення вигляду функції на екран */
  printf("f(x)=9/(ax)          при x [1,3)\n");
  printf("     abs(ax^2+x+b)   при x=3\n");
  /* Введення вхідних даних */
  printf ("\nВведіть значення a ");
```

```
  scanf("%f",&a);
  printf ("Введіть значення b ");
  scanf("%f",&b);
  printf ("Введіть значення x ");
  scanf("%f",&x);
  nom=0;
  /* Обчислення значення функції */
  if ((x>=1) && (x<3))   // Перший проміжок
    if (a==0) nom=0;    // ОДЗ порушено
    else y=9/(a*x);     // ОДЗ не порушено
  else
    if (x==3)   // Другий проміжок - особливостей нема
      y=fabs(a*x*x+x+b);
    else   // Значення x задано поза проміжком
      nom=2;
  /* Друк результату обчислення */
  switch (nom) {
    case 1: printf("Ділення на нуль"); break;
    case 2: printf("Функція не визначена"); break;
    default: printf("f(%4.2f)=%6.3f", x, y); }
  /* Затримка вікна виведення */
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
f(x)=9/(ax)          при x [1,3)
     abs(ax^2+x+b)   при x=3

Введіть значення a 1.5
Введіть значення b 0
Введіть значення x 2
f(2.00)= 3.000
```

Завдання. Написати програму, яка при введенні двох дат у форматі *день.місяць.рік* (дні, місяці і роки — цілочисельні дані) визначає і виводить на екран інформацію про те, яка дата раніша, а яка пізніша, чи дати однакові.

3. Оператор циклу for з параметром. Якщо треба виконати якийсь оператор чи групу операторів кілька разів і кількість повторень не залежить від результату роботи оператора, то найкраще скористатися оператором циклу з параметром. Параметром циклу може бути змінна будь-якого типу. Цикл з параметром може не виконуватися жодного разу.

Приклад 3.1. Обчислити суму квадратів усіх цілих чисел, які потрапляють в інтервал $(\ln x, e^x)$ при $x \geq 1$.

```
/* Знаходження суми квадратів усіх цілих чисел з інтервалу
 * від логарифма до експоненти */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
  int s,       // сума
      a,b,    // проміжок
      i;      // ціле число з проміжка
  double x;    // введене значення
  system("chcp 1251 & cls");
  printf("Введіть начення x (x>=1):  ");
  scanf("%lf",&x);
  s=0;
  a=(int)log(x)+1;
  b=(int)(exp(x));
  printf("Проміжок [%d,%d]\n",a,b);
  // Обчислення суми квадратів
  for (i=a; i<=b; i++)
    s=s+i*i;
  printf("Сума %5d",s);
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Введіть начення x (x>=1):  1
Проміжок [1,2]
Сума     5
```

```
Введіть начення x (x>=1):  3
Проміжок [2,20]
Сума  2869
```

Приклад 3.2. Вивести таблицю синусів і косинусів для кутів від 0° до 90° з кроком 15°.

Оскільки тригонометричні функції з бібліотеки math.h працюють з радіанами, то градусну міру кута треба перевести в радіанну за формулою $\varphi_r = \varphi_\circ \cdot \pi / 180$. Значення числа π (константу M_PI) треба взяти з бібліотеки math.h, вказавши перед її підмиканням директиву `#define _USE_MATH_DEFINES`. Параметром циклу буде змінна дійсного типу.

```
/* Таблиця синусів і косинусів для кутів
 * від 0° до 90° з кроком 15° */
#include <stdio.h>
#include <stdlib.h>
#define _USE_MATH_DEFINES
#include <math.h>
int main() {
  double g,   // градусна міра кута
         r;   // радіанна міра кута
  printf("    x       sin(x)       cos(x)\n");
  for (g=0.0; g<=90.0; g+=5.0) {
    r=g*M_PI/180.0;
    printf("%5.0f°%13.7lf%13.7lf\n",g,sin(r),cos(r));
  }
  printf("\n\n");
  system("pause");
  return 0;
}
```

x	sin(x)	cos(x)
0	0.00	1.00
15	0.26	0.97
30	0.50	0.87
45	0.71	0.71
60	0.87	0.50
75	0.97	0.26
90	1.00	0.00

Завдання. Написати програми, використовуючи цикли з параметром.

1. Вивести коди символів від '0' до '9' (параметром циклу взяти змінну символьного типу).

2. Обчислити за схемою Горнера значення полінома в точці x:

$$y = 11x^{10} + 10x^9 + 9x^8 + \ldots + 2x + 1$$

3. У скриньці є кульки з числами від 2 до 8 (генерувати випадкові числа) і є 5 гравців. Спочатку перший гравець дістає зі скриньки кульки, доки не витягне кульку з числом 5; йому приписується сума чисел всіх витягнутих кульок. Потім другий робить те саме і т.д. Вказати номер гравця, у якого сума чисел буде найбільшою.

4. Знайти максимальне значення факторіала, яке можна обчислити для заданого типу даних (використати тип unsigned long long int і константу ULLONG_MAX з бібліотеки limits.h).

4. Оператор циклу while з передумовою. Умова в операторі циклу з передумовою перевіряється перед кожним його виконанням. Доки вираз

набуває істинного (ненульового) значення, цикл виконується. Якщо вираз має значення хибність (нульове), то цикл закінчує роботу і керування передається оператору, наступному за оператором циклу. Цикл з передумовою може не виконуватися жодного разу.

Приклад 4.1. Написати програму для обчислення значень простих арифметичних виразів: *число знак_операції число*. Знаком операції може бути: +, -, *, /, ^ (піднесення до степеня). Обчислення виразів виконувати доти, поки не буде введено будь-яку букву. Якщо вираз буде введено з помилкою, то в буфері вводу залишиться інформація — її треба очистити. При реалізації алгоритму використаємо ту властивість, що функція scanf повертає кількість реально введених і присвоєних змінним елементів даних (полів). Для очищення буфера вводу можна скористатися оператором `while (getchar()!='\n');`.

Текст програми:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
/* Простий калькулятор. Працює в циклі до введення букви */
  double a,b,   // числа
         r;     // результат
  char oper;    // символ операції +, -, *, /, ^
  int ind;      // індикатор друку
  system("chcp 1251 & cls");
  printf("Введіть вираз: число операція число\n\
  \toперація + - * / ^\
  \nВвід букви з нового рядка закінчує роботу\n\n");
  while (scanf("%lf%c%lf",&a,&oper,&b)) {
    ind=1;
    switch (oper) {
      case '+': r=a+b;
                break;
      case '-': r=a-b;
```

```
                break;
      case '*': r=a*b;
                break;
      case '/': if (b==0.) {
                  ind=0;
                  printf("\tZerodivide");
                }
                else r=a/b;
                break;
      case '^': r=pow(a,b);
                  break;
      default: ind=0;
               printf("\tНевідома операція");
    }
    if (ind)
      printf("\t%g%c%g= %g  ",a,oper,b,r);
    // очистка буфера вводу (позначимо символом #)
    while (getchar()!='\n')
      printf("#");
    printf("\n");
  }
  printf("\n\n");
  return 0;
}
```

```
Введіть вираз: число операці
        операція + - * / ^
Ввід букви з нового рядка за

3++5
        3+5= 8
3  +5
        Невідома операція
3+   5
        3+5= 8
3+-5
        3+-5= -2
3/0
        Zerodivide
3-7+2
        3-7= -4  ##
5^3 кінець
        5^3= 125  #######
5*100000000
        5*1e+008= 5e+008
кінець


Press any key to continue .
```

Приклад 4.2. Вивести таблицю символів і відповідних їм кодів ASCII від 32 до 255. У програмі використати одну однобайтну змінну (інші змінні не використовувати). Алгоритм побудувати так, щоб програма не зациклювалася (можливість зациклювання пов’язана з тим, що якщо однобайтній змінній присвоїти значення 256, то буде переповнення пам’яті і ця змінна набуде значення 0).

```
#include <stdio.h>
#include <stdlib.h>
int main() { // Виведення символів з кодами від 32 до 255
  unsigned char sym;   // символи
```

```
  sym=31;
  printf("\t\t ASCII\n");
  while (sym<255) {
    sym++;
    printf("\t%c - %d\t",sym,sym);
  }
  printf("\n\n");
  system("pause");
  return 0;
}
```

ASCII

- 32	! - 33	" - 34	# - 35	$ - 36
% - 37	& - 38	' - 39	(- 40	) - 41
* - 42	+ - 43	, - 44	- - 45	. - 46
/ - 47	0 - 48	1 - 49	2 - 50	3 - 51
4 - 52	5 - 53	6 - 54	7 - 55	8 - 56
9 - 57	: - 58	; - 59	< - 60	= - 61
> - 62	? - 63	@ - 64	A - 65	B - 66
C - 67	D - 68	E - 69	F - 70	G - 71
H - 72	I - 73	J - 74	K - 75	L - 76
M - 77	N - 78	O - 79	P - 80	Q - 81
R - 82	S - 83	T - 84	U - 85	V - 86
W - 87	X - 88	Y - 89	Z - 90	[- 91
\ - 92	] - 93	^ - 94	_ - 95	` - 96
a - 97	b - 98	c - 99	d - 100	e - 101
f - 102	g - 103	h - 104	i - 105	j - 106
k - 107	l - 108	m - 109	n - 110	o - 111
p - 112	q - 113	r - 114	s - 115	t - 116
u - 117	v - 118	w - 119	x - 120	y - 121
z - 122	{ - 123	| - 124	} - 125	~ - 126
⌂ - 127	А - 128	Б - 129	В - 130	Г - 131
Д - 132	Е - 133	Ж - 134	З - 135	И - 136
Й - 137	К - 138	Л - 139	М - 140	Н - 141
О - 142	П - 143	Р - 144	С - 145	Т - 146
У - 147	Ф - 148	Х - 149	Ц - 150	Ч - 151
Ш - 152	Щ - 153	Ъ - 154	Ы - 155	Ь - 156
Э - 157	Ю - 158	Я - 159	а - 160	б - 161
в - 162	г - 163	д - 164	е - 165	ж - 166
з - 167	и - 168	й - 169	к - 170	л - 171
м - 172	н - 173	о - 174	п - 175	- 176
▒ - 177	▓ - 178	| - 179	┤ - 180	╡ - 181
╢ - 182	╖ - 183	╣ - 184	╕ - 185	╗ - 186
╝ - 187	╜ - 188	┘ - 189	┐ - 190	└ - 191
┴ - 192	┬ - 193	├ - 194	─ - 195	┼ - 196
╞ - 197	╟ - 198	╚ - 199	╔ - 200	╩ - 201
╦ - 202	╠ - 203	═ - 204	╬ - 205	╧ - 206
╨ - 207	╤ - 208	╥ - 209	╙ - 210	╘ - 211
╒ - 212	╓ - 213	╫ - 214	╪ - 215	┘ - 216
┌ - 217	■ - 218	▒ - 219	▄ - 220	▌ - 221
▐ - 222	▀ - 223	р - 224	С - 225	т - 226
у - 227	ф - 228	х - 229	ц - 230	ч - 231
ш - 232	щ - 233	ъ - 234	ы - 235	ь - 236
э - 237	ю - 238	я - 239	Ё - 240	ё - 241
Є - 242	є - 243	Ї - 244	ї - 245	ў - 246
ў - 247	° - 248	∙ - 249	· - 250	√ - 251
№ - 252	¤ - 253	■ - 254	- 255	

Якщо в програмі перед виведенням таблиці використати оператор зміни розкладки клавіатури `system("chcp 1251 & cls");`, то буде побудована таблиця кодів ANSI, а не ASCII.

Приклад 4.3. Знайти “машинне епсилон” — мінімальне, відмінне від 0 дійсне число, яке після додавання до 1.0 дає ще результат, відмінний від 1.0 Стандартне бібліотечне (бібліотека `float.h`) значення “машинного епсилон” типу double `DBL_EPSILON = 2.2204460492503131e-016` (див. Додаток В, п. 5).

```
#include <stdio.h>
#include <stdlib.h>
#include <float.h>
int main() {
  double eps=1.;
  system("chcp 1251 & cls");
  while (eps/2+1.0>1.0) eps=eps/2;
  printf("\nОбчислене машинне епсилон типу double = %.18e",eps);
  printf("\nБібліотечне епсилон типу double\
  DBL_EPSILON = %.18e", DBL_EPSILON);
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Обчислене машинне епсилон типу double = 2.22044604925031310e-016
Бібліотечне епсилон типу double  DBL_EPSILON = 2.22044604925031310e-016
```

Оскільки треба знайти останнє значення eps, яке впливає на результат, то в умові закінчення циклу використати вираз eps+1.0>1.0 не можна, бо це буде перше eps, яке не впливає на результат. Тому умова має бути eps/2+1.0>1.0. Крім того, можна подати такий фрагмент програми, скориставшись робочою змінною eps1:

```
  double eps=1,eps1;
  while (eps+1.0>1.0) {eps1=eps; eps=eps/2;}
  printf("\nОбчислене машинне епсилон типу double = %.18e",eps1);
```

Завдання. Написати програми, використовуючи цикли з передумовою.

1. За введеним цілим числом утворити число з цифрами, поданими у зворотному порядку (реверс цілого числа: 3210 — 0123). При виведенні результату для відображення попередніх нулів скористатися модифікатором формату *.

2. Порахувати кількість речень у введеному тексті. Речення закінчується крапкою, знаком оклику, знаком питання.

3. Знайти найбільший спільний дільник (НСД) двох чисел, використовуючи алгоритм Евкліда: від більшого числа віднімаємо менше, доки числа не стануть рівними — це і є НСД.

Практичне заняття № 8 (*комп'ютерний практикум № 5*)
Розробка і реалізація алгоритмів розгалужених процесів з використанням вкладених умовних операторів і оператора вибору

Мета заняття: набути практичних навичок складати алгоритми і програмувати засобами мови С розгалужені процеси, використовуючи вкладені умовні оператори if, а також оператор вибору switch.

Завдання. Розробити два алгоритми з вкладеною перевіркою умов. У першому алгоритмі передбачити видачу повідомлень про помилки зразу ж при їхньому виявленні, а видачу результату обчислення значення функції $f(x)$ — у кінці програми. Відповідно до другого алгоритму видача повідомлень про помилки і видача результату обчислення значення функції $f(x)$ мають виконуватися в кінці програми.

Намалювати блок-схеми й написати програми обчислення значень заданої логічною залежністю функції $f(x)$ при довільних значеннях параметрів a і b і незалежної змінної x. В алгоритмах реалізувати перевірку ОДЗ. Вважається, що функція визначена на заданому проміжку, а поза ним *не визначена*. Порівняння в умовах не повторювати. За алгоритмами провести розрахунки не менш ніж з п'ятьма різними наборами вхідних даних, які пе-

редбачають виконання різних гілок алгоритму. Коментарі в програмі обов'язкові (17-25%).

Варіанти завдань

1. $f(x)=\begin{cases}\sin\ 3x & \text{при } x\in[0,5),\\ 2e^{ax-1}+1 & \text{при } x=5,\\ (bx-a)^{-1} & \text{при } x\in(5,12].\end{cases}$

2. $f(x)=\begin{cases}\sqrt{ax+2} & \text{при } x\in(-3,2),\\ |\cos 2x| & \text{при } x=2,\\ 3x^2+bx+1 & \text{при } x\in(2,+\infty).\end{cases}$

3. $f(x)=\begin{cases}bx^2+x-6 & \text{при } x\in[0,1),\\ \left(\sqrt{ax+b}\right)^{-1} & \text{при } x=1,\\ \cos\ x & \text{при } x\in(1,9).\end{cases}$

4. $f(x)=\begin{cases}\sin 5x & \text{при } x\in[-0,7),\\ x^2-bx & \text{при } x=7,\\ \ln(ax-8) & \text{при } x\in(7,10).\end{cases}$

5. $f(x)=\begin{cases}|ax+7| & \text{при } x\in(-\infty,9),\\ 5\sqrt{bx+1} & \text{при } x\in[9,15),\\ \sin x & \text{при } x=15.\end{cases}$

6. $f(x)=\begin{cases}\sin\ x & \text{при } x\in(-\infty,-2),\\ 2e^{ax}-1 & \text{при } x=-2,\\ (bx+2a)^{-2} & \text{при } x\in(-2,8).\end{cases}$

7. $f(x)=\begin{cases}\sqrt{5x^2-a} & \text{при } x\in(-\infty,3),\\ bx+8 & \text{при } x=3,\\ \cos x & \text{при } x\in(3,11).\end{cases}$

8. $f(x)=\begin{cases}2x+7 & \text{при } x\in[0,5),\\ 3e^{ax} & \text{при } x=5,\\ \ln(bx+a) & \text{при } x\in(5,8).\end{cases}$

9. $f(x)=\begin{cases}\sin\ x & \text{при } x\in[-1,6),\\ 3x^2+bx-3 & \text{при } x=6,\\ \ln(bx+a) & \text{при } x\in(6,+\infty),\end{cases}$

10. $f(x)=\begin{cases}\sqrt{x^2-a} & \text{при } x\in(-3,3),\\ 5x^3-bx+3 & \text{при } x=3,\\ \cos x & \text{при } x\in(3,8).\end{cases}$

11. $f(x)=\begin{cases}\sin\ x & \text{при } x\in[0,7),\\ 3e^{ax+1}-1 & \text{при } x=7,\\ (bx-a)^{-1} & \text{при } x\in(7,11).\end{cases}$

12. $f(x)=\begin{cases}\sqrt{ax-3} & \text{при } x\in(-8,3),\\ |x^2-bx+7| & \text{при } x=3,\\ \cos(x+1) & \text{при } x\in(3,+\infty).\end{cases}$

13. $f(x)=\begin{cases}\cos\ x & \text{при } x\in[0,5),\\ \left(\sqrt{ax-b}\right)^{-1} & \text{при } x=5,\\ bx^2+3x-2 & \text{при } x\in(5,8).\end{cases}$

14. $f(x)=\begin{cases}\sin\ x & \text{при } x\in[-1,6),\\ 3x^2+bx & \text{при } x=6,\\ \ln(ax-3b) & \text{при } x\in(6,8).\end{cases}$

15. $f(x)=\begin{cases}\sqrt{3x^2-a} & \text{при } x\in(-3,3),\\ -bx+3 & \text{при } x=3,\\ \cos(x-4) & \text{при } x\in(3,8).\end{cases}$

16. $f(x)=\begin{cases}\cos x & \text{при } x\in(-\infty,4),\\ x^3-bx+1 & \text{при } x=4,\\ \sqrt{ax} & \text{при } x\in(4,7).\end{cases}$

17. $f(x)=\begin{cases}\sin\ x & \text{при } x\in[-1,3),\\ \sqrt{ax-2} & \text{при } x=3,\\ bx+a & \text{при } x\in(3,5).\end{cases}$

18. $f(x)=\begin{cases}\cos\ x & \text{при } x\in[0,4),\\ 3x^2+bx & \text{при } x=4,\\ \ln(ax+b) & \text{при } x\in(4,9).\end{cases}$

19. $f(x)=\begin{cases}\sin\ x & \text{при } x\in[0,2),\\ 2e^{ax} & \text{при } x=2,\\ (bx+a)^{-1} & \text{при } x\in(2,8).\end{cases}$

20. $f(x)=\begin{cases}|ax+x^2| & \text{при } x\in(-\infty,4),\\ \sqrt{bx-1}+3 & \text{при } x=4,\\ \cos x & \text{при } x\in(4,10].\end{cases}$

21. $f(x)=\begin{cases}\sqrt{x+b} & \text{при } x\in[0,11),\\ ax^2-3x+14 & \text{при } x\in[11,15),\\ e^{-ax} & \text{при } x=15.\end{cases}$

22. $f(x)=\begin{cases}\sqrt{x^3-a} & \text{при } x\in(-\infty,7),\\ 3x^2-x+b & \text{при } x=7,\\ \sin 2x & \text{при } x\in(7,8).\end{cases}$

23. $f(x)=\begin{cases}\sqrt{bx+3} & \text{при } x\in(-2,10),\\ |x^2-abx+7| & \text{при } x=10,\\ \cos(x-1) & \text{при } x\in(10,+\infty).\end{cases}$

24. $f(x)=\begin{cases}\ln(ax+3) & \text{при } x\in(-2,5),\\ x^2-bx+7 & \text{при } x=5,\\ \cos(x+1) & \text{при } x\in(5,+\infty).\end{cases}$

25. $f(x)=\begin{cases}\sqrt{x^3+a} & \text{при } x\in(-\infty,7),\\ -bx-3 & \text{при } x=7,\\ \sin x & \text{при } x\in(7,10).\end{cases}$

26. $f(x)=\begin{cases}\cos\ x & \text{при } x\in[0,9),\\ 3e^{ax}+1 & \text{при } x=9,\\ \ln(bx-1) & \text{при } x\in(9,11).\end{cases}$

27. $f(x)=\begin{cases}\sin(x+1) & \text{при } x\in[0,1),\\ ax^2+2bx-4 & \text{при } x\in[1,7),\\ (ax+b)^{-1} & \text{при } x=7.\end{cases}$

28. $f(x)=\begin{cases}|ax+b| & \text{при } x\in(-\infty,8)\\ 5\sqrt{bx+3} & \text{при } x=8,\\ \cos x & \text{при } x\in(8,10].\end{cases}$

29. $f(x)=\begin{cases}\sin\ x & \text{при } x\in[0,5),\\ 3e^{ax}+1 & \text{при } x=5,\\ (bx-10)^{-1} & \text{при } x\in(5,11).\end{cases}$

30. $f(x)=\begin{cases}\cos x & \text{при } x\in(-2,4],\\ \sqrt{ax+b} & \text{при } x\in(4,9),\\ x^3-bx+3 & \text{при } x=9.\end{cases}$

Практичне заняття № 9 (*комп'ютерний практикум № 6*)
Проектування алгоритмів і програм циклічної структури (оператор циклу з параметром)

Мета заняття: набути практичних навичок розробляти алгоритми і програмувати засобами мови С циклічні процеси, використовуючи оператор циклу з параметром, і оформляти результати у вигляді таблиць, у яких інформація подається по рядках. Навчитися знаходити суми з найбільшою точністю.

Завдання 1. Розробити алгоритм, подати узагальнену блок-схему (без пооператорної деталізації, але в ній зобразити цикл, у якому обчислюються значення членів послідовності) і написати програму розв'язування поданої задачі.

Знайти перші m ($m \geq 1$) членів послідовності $\{x_n\}$ ($n = 1, 2, 3, \ldots$). Результат видати на екран у таблиці по рядках з k ($1 \leq k \leq 6$) колонками; верхня і нижня границі таблиці мають узгоджуватися з її розмірами. Визначити умову переходу на новий рядок в таблиці при будь-якому k. Врахувати, що може бути введено $m < k$. Якщо результати повністю не заповнюють останній рядок таблиці, то доповнити таблицю порожніми клітинками. За різних умов блоки дій не дублювати. В алгоритмі передбачити перевірку правильності введення даних до першої помилки. Коментарі в програмі обов'язкові (17-25%). За алгоритмом провести розрахунки не менш ніж з чотирма різними значеннями m і k.

Загальна схема програми має бути такою:

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <windows.h>   // робота з кирилицею
int main() {
/* Обчислення значень членів послідовності і
 * виведення значень в таблиці */
  int m,   // кількість ченів плслідовності (m>=1)
      i, // номер члена послідовності
      k;   // k – кількість колонок (від 1 до 6)
  double xn;   // значення члена послідовності
  char r[]="-----------";   // обмежувальна лінія таблиці
  SetConsoleOutputCP(1251);   /* установка кодової сторінки
                                 win-cp 1251 кодування
                                 ANSI) в потік виводу */
  printf("Загальний член послідовності: ");
  printf("\txn=cos(5n-2)/(3n-1)\n\n");
  printf("Вкажіть кількість членів послідовності\
    (не менше 1) = ");
  scanf("%d",&m);
  /* Перевірити, чи m>=1, якщо – ні, то видати відповідне
     повідомлення про помилку і закінчити роботу програми
     (оператори return і break не використовувати) */
...
```

```
    printf("Вкажіть кількість колонок (від 1 до 6) = ");
    scanf("%d",&k);
    /* Перевірити, чи 1<=k<=6, якщо - ні, то видати
       відповідне повідомлення про помилку і закінчити
       роботу програми
       (оператори return і break не використовувати) */
...
      /* видача таблиці */
      /* Передбачити варіант, коли m<k */
      printf("%s",r); printf("-\n");
      for (i=1; i<=m; i++) { /* виведення рядків таблиці */
        xn=cos(5.*i-2.)/(3.*i-1.);
        /* Перехід у таблиці на новий рядок.
           Перевірку виконати так, щоб перехід на новий
           рядок працював для 1<=k<=6. ОПТИМІЗУВАТИ
           цю перевірку (можливо, змінивши порядок
           операторів) */
        if (i%k==1 && i>k)
          printf("|\n");
        printf("| %8.5f ",xn);
      }
      /* якщо треба, то доповнити таблицю
         порожніми клітинками */
      /* закриття таблиці */
      printf("%s",r); printf("-\n");
      printf("Результат видано по рядках");
...
  printf("\n\n");
  system("pause");
  return 0;
}
```

При роботі програми на екран має виводитися така інформація:

```
Загальний член послідовності:   xn=cos(5n-2)/(3n-1)

Вкажіть кількість членів послідовності (не менше 1) = 0
m має бути не меншим від 1
```

```
Загальний член послідовності:   xn=cos(5n-2)/(3n-1)

Вкажіть кількість членів послідовності (не менше 1) = 9
Вкажіть кількість колонок (від 1 до 6) = 7
k має бути в межах від 1 до 6
```

```
Загальний член послідовності:   xn=cos(5n-2)/(3n-1)

Вкажіть кількість членів послідовності (не менше 1) = 7
Вкажіть кількість колонок (від 1 до 6) = 3
-----------------------------------
| -0.49500 | -0.02910 |  0.11343 |
|  0.06003 | -0.03806 | -0.05662 |
| -0.00066 |          |          |
-----------------------------------
Результат видано по рядках
```

Варіанти завдань

1. $x_n = \dfrac{\sin(n^2)+n}{3n^2-1}$;
2. $x_n = \dfrac{\ln(3n)-1}{2n^2+3}$;
3. $x_n = \dfrac{\cos(n-1)}{n^2+4n}$;
4. $x_n = \dfrac{\ln(n+1)}{3n^2-1}$;
5. $x_n = \dfrac{\sin(5n^2)+1}{3+n}$;
6. $x_n = \dfrac{\cos(7n)+4}{2n-3}$;
7. $x_n = \dfrac{\sin(3n-2)}{n^2+n}$;
8. $x_n = \dfrac{\ln(5+n)}{3n-2}$;
9. $x_n = \dfrac{\cos(2n)-4}{2n+3}$;
10. $x_n = \dfrac{\sin(5n-3)+n}{2n+3}$;
11. $x_n = \dfrac{\ln(n^2+1)}{3n^2}$;
12. $x_n = \dfrac{\cos(n+3)-2}{2n+1}$;
13. $x_n = \dfrac{\cos(2n+1)-4}{5n+1}$
14. $x_n = \dfrac{\sin(2n+3)+4}{n}$;
15. $x_n = \dfrac{\ln(n+4)}{3n+1}$;
16. $x_n = \dfrac{\cos(3n-1)}{n+2}$;
17. $x_n = \dfrac{\ln(3n)+5}{2n+5}$;
18. $x_n = \dfrac{\sin(2n+3)-2}{n+1}$;
19. $x_n = \dfrac{\sin(n-1)}{n^2+2}$;
20. $x_n = \dfrac{\cos(2n-5)+n}{4n+1}$;
21. $x_n = \dfrac{\cos(n-1)}{n^2+1}$;
22. $x_n = \dfrac{\sin(3n-2)}{n+3}$;
23. $x_n = \dfrac{\sin n^2-5}{3n^2+n}$;
24. $x_n = \dfrac{\cos(n^2-5)+4}{3n^2+1}$;
25. $x_n = \dfrac{\ln(3n)+5}{n^2}$;
26. $x_n = \dfrac{\sin(n-1)+4}{n^2+n}$;
27. $x_n = \dfrac{\ln(5n)+1}{2n+3}$;
28. $x_n = \dfrac{\cos(2n)+5}{3n^2+1}$;
29. $x_n = \dfrac{\ln(n^2+4)}{5n^2-n}$;
30. $x_n = \dfrac{\cos(n+2)}{n^2+3}$;

Завдання 2. Виконати подані нижче програми. У звіті подати скриншоти результатів. Вказати розрядність і тип процесора комп'ютера, на якому виконувалася програма. Пояснити (детально письмово у звіті), чому одержано різні значення при обчисленнях тих самих сум. Тип дійсних змінних залишити float.

1) Знаходження суми в прямому й зворотному порядку:

```
/* УВАГА: програма може працювати кілька хвилин –
 * не зависає */
#include <stdlib.h>
#include <stdio.h>
int main() {
  float s1,s2;
  long long int n;
  /* знаходження суми від більшого значення до меншого */
  s1=0.;
  for (n=1; n<= 100000000; n++)
    s1+=1./(n*n);
  /* знаходження суми від меншого значення до більшого */
  s2=0.;
  for (n=100000000; n>=1; n--)
    s2+=1./(n*n);
  printf("s1=%.17lf\n",s1);
  printf("s2=%.17lf\n\n",s2);
  system("pause");
  return 0;
}
```

2) Додавання до суми великого числа:

```
#include <stdlib.h>
#include <stdio.h>
int main() {
  float s1,s2;
  long long int n;
```

```
  /* знаходження суми - велике число додаємо на початку */
  s1=1000000;
  for (n=1; n<= 100000000; n++)
    s1+=1./(n*n);
  /* знаходження суми - велике число додаємо в кінці */
  s2=0;
  for (n=1; n<= 100000000; n++)
    s2+=1./(n*n);
  s2=s2+1000000;
  printf("s1=%.17f\n",s1);
  printf("s2=%.17f\n\n",s2);
  system("pause");
  return 0;
}
```

Подані програми демонструють, що *від перестановки* (певного групування) *доданків сума змінюється*.

Практичне заняття № 10
Циклічні структури з післяумовою, ітераційні цикли. Вкладені цикли

Мета заняття: закріпити лекційний матеріал і вдосконалити навички щодо використання й програмування циклів з післяумовою, їхнього використання при програмуванні ітераційних процесів, програмування вкладених циклів.

1. Оператор циклу do-while з післяумовою. Умова в операторі циклу з післяумовою перевіряється після кожного його виконанням. Якщо вираз має значення хибність (нульове), то цикл закінчує роботу і керування передається оператору, наступному за оператором циклу. Цикл з післяумовою виконується принаймні один раз.

Приклад 1.1. Вгадування числа — програма генерує випадкове число, а користувач вгадує його. Результатом є кількість спроб вгадати число.

```
#include <stdio.h>
#include <stdlib.h>   // Для одержання випадкових чисел
#include <time.h>   // для srand(time(0));)
int main () {
/*Генерація випадкових чисел від 1 до 100 і вгадування */
  int r, x;   // згенероване і введене числа
  int n;   // кількість спроб при вгадуванні
  system("chcp 1251 & cls ");   // зміна кодової сторінки
  srand(time(0));
  n=0; r=rand()%100+1;   /* число з проміжка від 1 до 100
*/
  do {
    printf("\nВведіть число від 1 до 100   ");
    scanf("%d",&x);
    if(r<x) printf("\tведено більше число");
    if(r>x) printf("\tведено менше число");
    n++;
  } while (r-x);
  printf("Ви вгадали число за %d спроб",n);
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Введіть число від 1 до 100   33
        введено більше число
Введіть число від 1 до 100   17
        введено менше число
Введіть число від 1 до 100   25
Ви вгадали число за 3 спроб
```

Приклад 1.2. Вивести на екран всі цілі числа від *n* до 1 по 10 у рядку.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int n,   // введене число
      i;   // лічильник кількості кроків виконання циклу
  system("chcp 1251 & cls");
  printf("Введіть значення N ");
```

```
  scanf("%d",&n);
  if (n<1) printf("Число менше від 1");
  else {
    i=1;
    do {
      printf("%5d",n);
      if (!(i%10)) printf("\n");
      i++;
    } while(--n);
  }
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Введіть значення N 0
Число менше від 1
```

```
Введіть значення N 1
    1
```

```
Введіть значення N 45
   45   44   43   42   41   40   39   38   37   36
   35   34   33   32   31   30   29   28   27   26
   25   24   23   22   21   20   19   18   17   16
   15   14   13   12   11   10    9    8    7    6
    5    4    3    2    1
```

Завдання. Написати програми, використовуючи цикл з післяумовою.

1. Порахувати кількість цифр натурального числа.

2. Порахувати кількість речень у введеному тексті. Речення закінчується крапкою, знаком оклику, знаком питання.

2. Ітераційні цикли. У програмуванні цикл do-while з післяумовою використовується набагато рідше, ніж while чи for. Але при програмуванні ітераційних процесів йому надається перевага.

Приклад 2.1. Обчислення з точністю до “машинного нуля” значення суми числового ряду $\frac{1}{1\cdot 2\cdot 3}+\frac{1}{2\cdot 3\cdot 4}+\frac{1}{3\cdot 4\cdot 5}+\frac{1}{4\cdot 5\cdot 6}+\ldots$

```
#include <stdio.h>
#include <stdlib.h>
int main() {
// обчислення суми ряду з точністю до "машинного нуля"
  double sum1, sum2;   // попередня й поточна суми
  int n;   // номер ітерації
  system("chcp 1251 & cls");
  n=0;
  sum2=0.;
```

```
  do {
    n++;
    sum1=sum2;
    sum2 +=1.0/(n*(n+1)*(n+2));
  } while (sum1<sum2);
  printf("Сума %d членів ряду\
    \n\t1/(1*2*3)+1/(2*3*4)+... = %lg",n,sum2);
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Сума 1290 членів ряду
        1/(1*2*3)+1/(2*3*4)+... = 0.25
```

У наведеній програмі сума обчислюється як значення змінної sum2. Її попереднє значення зберігається в змінній sum1. Оскільки наближене значення з додаванням невід'ємних доданків не зменшується, умовою продовження циклу є відношення sum1<sum2. Коли при додаванні поточного доданка значення суми залишається незмінним (за рахунок скінченності розрядної сітки для подання дійсних чисел), порушується умова sum1<sum2 і цикл припиняє виконуватися. Скінченність розрядної сітки подання дійсних чисел у комп'ютері визначає "машинний нуль".

Приклад 2.2. Обчислити квадратний корінь з будь-якого дійсного числа *a*.

Треба розглянути такі випадки:

— якщо *a*<0, то квадратний корінь не існує;

— якщо *a*=0, то корінь дорівнює 0;

— якщо *a*>0, то для обчислення квадратного кореня застосувати метод Ньютона (метод дотичних для знаходження нулів функції з заданим початковим наближенням; цей метод збігається дуже швидко):

$$b_0 = 1; \quad b_{n+1} = \frac{\frac{a}{b_n} + b_n}{2},$$

де b_n — наближене значення $\sqrt{a}$.

Для виходу із циклу ітерацій (послідовних наближень) можна застосувати абсолютну похибку

$$\left|a - b_n^2\right| < 10^{-6},$$

Але цей критерій не спрацює, якщо число буде меншим від 10^{-6}. Тому тут треба скористатися відносною похибкою:

$$\left|\frac{a}{b_n^2} - 1\right| < 10^{-6}.$$

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
/* Обчислення квадратного кореня за методом дотичних Ньютона */
  const double eps=1.e-6;
  double a,   // введене число
         bn;  // наближене значення кореня
  system("chcp 1251 & cls");
  printf("Введіть дійсне число ");
  scanf("%lg",&a);
  if (a<0) printf("\tКорінь не існує");
  else
    if (a==0) printf("\t0");
    else {   // число >0
       bn=1;
       do
         bn=(a/bn+bn)/2;
       while (fabs(a/(bn*bn)-1)>=eps);
       printf("\t%lg",bn);
     }
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Введіть дійсне число -8
        Корінь не існує
```

```
Введіть дійсне число 25
        5
```

```
Введіть дійсне число 2
        1.41421
```

При програмувані ітераційних циклів для підстраховки від зациклювання, яке може виникнути через некоректні вхідні дані, можна встановлювати обмеження щодо кількості кроків ітерацій.

Приклад 2.3. Написати програму обчислення наближеного значення функції:

$$\frac{x}{3}+\frac{x^2}{6}+\frac{x^3}{9}+\frac{1}{3}(1-x^3)\ln(1-x)=\sum_{n=1}^{\infty}\frac{x^{n+3}}{n(n+3)} \quad \text{при } x\in[-1,1).$$

При введенні даних перевіряти їхню коректність, очищати буфер вводу від зайвої інформації (скористатися функцією fseek з бібліотеки stdio.h). Встановити обмеження щодо кількості кроків ітерацій. Ітераційно обчислене наближене значення функції порівняти з точним значенням.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
/* Обчислення наближеного значення за степеневим рядом */
  double x,eps,   // аргумент функції і точність обчислень
         xn,   // степінь аргумента
         an,   // значення загального члена ряду
         s,f;   // наближене і точне значення функції
  int ki,   // обмеження кількості ітерацій
      n,   // кількість ітерацій при обчисленні
      ind;   // індикатор помилки при вводі
  system("chcp 1251 & cls");
  printf("Функція f=x/3+x^2/6+x^3/9+(1-x*x*x)*ln(1-x)/3\n");
  printf("ряд E(n=1..+00)(x^(n+3))/(n*(n+3))  при x[-1,1)\n");
  /* введення вхідних даних */
  ind=1;
  while (ind) {
    printf("  введіть x ");
    scanf("%lf",&x);
    if (x<-1 || x>=1)
      printf("\tневірне значення x\n");
    else ind=0;
    fseek(stdin,0,SEEK_END);   // очистка буфера вводу
  }
```

```
ind=1;
while (ind) {
  printf("  введіть eps ");
  scanf("%lf",&eps);
  if (eps<=0)
    printf("\tневірне значення eps\n");
  else ind=0;
  fseek(stdin,0,SEEK_END);
}
ind=1;
while (ind) {
  printf("  введіть ki ");
  scanf("%d",&ki);
  if (ki<1)
    printf("\tневірне значення ki\n");
  else ind=0;
  fseek(stdin,0,SEEK_END);
}
/* обчислення значення степеневого ряду */
n=0;
xn=x*x*x;
s=0;
do {
  n++;
  xn=xn*x;
  an=xn/(n*(n+3));
  s=s+an;
} while (fabs(an)>=eps && n<ki);
/* виведення результатів */
if (n==ki)
  printf("досягнуто ліміту кількості ітерацій\n");
printf("результат:\n");
printf("  кількість ітерацій n= %d\n",n);
printf("  наближене значення функції s= %0.9f\n",s);
```

```
   f=x/3+x*x/6+x*x*x/9+(1-x*x*x)*log(1-x)/3;
   printf("  точне значення функції f= %0.9f\n",f);
   printf("  похибка |f-s|= %0.9f\n",fabs(f-s));
   printf("\n\n");
   system("pause");
   return 0;
}
```

```
Функція f=x/3+x^2/6+x^3/9+(1-x*x*x)*ln(1-x)/3
ряд E(n=1..+00)(x^(n+3))/(n*(n+3))  при x[-1,1)
  введіть x 5
        невірне значення x
  введіть x 0.5
  введіть eps 0
        невірне значення eps
  введіть eps 1e-10
  введіть ki 0
        невірне значення ki
  введіть ki 1000
результат:
  кількість ітерацій n= 22
  наближене значення функції s= 0.020054295
  точне значення функції f= 0.020054295
  похибка |f-s|= 0.000000000
```

```
Функція f=x/3+x^2/6+x^3/9+(1-x*x*x)*ln(1-x)/3
ряд E(n=1..+00)(x^(n+3))/(n*(n+3))  при x[-1,1)
  введіть x 0.5
  введіть eps 1e-10
  введіть ki 15
досягнуто ліміту кількості ітерацій
результат:
  кількість ітерацій n= 15
  наближене значення функції s= 0.020054283
  точне значення функції f= 0.020054295
  похибка |f-s|= 0.000000011
```

```
Функція f=x/3+x^2/6+x^3/9+(1-x*x*x)*ln(1-x)/3
ряд E(n=1..+00)(x^(n+3))/(n*(n+3))  при x[-1,1)
  введіть x -0.1  є [-1;1]
  введіть eps 0,0001
        невірне значення eps
  введіть eps 0.0001
  введіть ki 10 >0
результат:
  кількість ітерацій n= 1
  наближене значення функції s= 0.000025000
  точне значення функції f= 0.000024052
  похибка |f-s|= 0.000000948
```

Завдання. Запрограмувати ітераційні процеси, використовуючи цикл з післяумовою.

1. Обчислити корінь степеня k з будь-якого дійсного числа a за методом дотичних Ньютона. Для врахування можливих варіантів значення степеня k і значення числа a запрограмувати такі вкладені умови:

якщо k=0, то корінь знайти не можливо;

інакше,

якщо a<0 і k — парне або a=0 і k — від'ємне, то корінь не існує

інакше,

якщо k=1 або a=0, то результатом є a;

інакше,

якщо k=-1, то результатом є $\frac{1}{a}$;

інакше, якщо k<0, то шукати $\frac{1}{\sqrt[k]{a}}$.

Для обчислення кореня степеня k застосувати метод Ньютона (метод дотичних для знаходження нулів функції з заданим початковим наближенням; цей метод збігається дуже швидко):

$$b_0 = 1; \quad b_{n+1} = \frac{\frac{a}{b_n^{k-1}} + (k-1)b_n}{k}$$

де b_n — наближене значення $\sqrt[k]{a}$.

Для виходу із циклу ітерацій (послідовних наближень) скористатися відносною похибкою:

$$\left| \frac{a}{b_n^k} - 1 \right| < 10^{-6}.$$

Програма має обчислювати корені (вводити числа і виводити значення кореня з заданою точністю), доки не буде введено замість числа будь-який символ, відмінний від цифри, наприклад, букву.

2. Обчислити значення числа π за двома формулами:

$$\pi = 3 + 4\left(\frac{1}{2 \cdot 3 \cdot 4} - \frac{1}{4 \cdot 5 \cdot 6} + \frac{1}{6 \cdot 7 \cdot 8} - \ldots \right) = 3 + \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n(2n+1)(n+1)};$$

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots \right) = 4 \cdot \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{2n-1}.$$

Порівняти швидкість збіжності — кількість ітерацій (доданків) для досягнення заданої точності ε порявняно з бібліотечним числом π (бібліотека math.h).

3. Вкладені цикли. Як і умовні оператори, цикли можна вкладати один в оден. Це можуть бути цикли різних типів. Цикли можуть бути як залежними (вкдадений цикл при обчисленні значень своїх параметрів використовує значення, змінювані в зовнішньому циклі), так і незалежними.

Приклад 3.1. Сформувати таблицю Келі для операції множення.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
/* Формування таблиці Келі для операції множення */
  int i,j;   // параметри циклів-множники
  system("chcp 1251 & cls");
  printf("\tТаблиця множення\n");
```

```
  for (i=1; i<=10; i++) {
    for (j=1; j<=10; j++)
      printf("%4d",i*j);
    printf("\n");
  }
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
                Таблиця множення
   1   2   3   4   5   6   7   8   9  10
   2   4   6   8  10  12  14  16  18  20
   3   6   9  12  15  18  21  24  27  30
   4   8  12  16  20  24  28  32  36  40
   5  10  15  20  25  30  35  40  45  50
   6  12  18  24  30  36  42  48  54  60
   7  14  21  28  35  42  49  56  63  70
   8  16  24  32  40  48  56  64  72  80
   9  18  27  36  45  54  63  72  81  90
  10  20  30  40  50  60  70  80  90 100
```

Приклад 3.2. Визначити за введеною датою номер дня в році (шляхом додавання кількостей днів за попередні місяці). Дати вводити доти, поки не буде введено будь-який символ, відмінний від цифри. При введенні неповної інформації дані брати з попереднього кроку (для цього також ініціалізувати відповідні змінні).

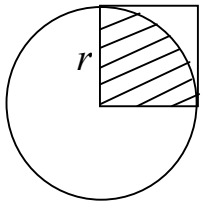
```
#include <stdio.h>
#include <stdlib.h>
int main() {
/* Визначення номера дня в році.
 * У циклі вводяться дати в форматі дд.мм.рррр */
  int dd, mm=1, rr=2019;   // введена інформація
  int k,   //  номер дня в році
      mr;   // робоча змінна - номер попереднього місяця
  system("chcp 1251 & cls");
  printf("Вводьте дати за форматом дд.мм.рррр\
    \nДля закінчення – букву\n");
  while (scanf("%d.%d.%d", &dd,&mm,&rr)){
    /* кількість днів за попередні місяці */
    k=0; mr=1;
    while (mr<mm) {
      switch (mr) {
        case 4: case 6: case 9: case 11:
                k=k+30; break;
        case 2: if (rr%4==0 && (rr%100!=0 || rr%400==0))
```

```
                    k=k+29;
                else k=k+28;
                break;
          default: k=k+31;
        }
        mr++;
      }
      /* одержання результату */
      k=k+dd;
     printf("   Номер дня в році - %3d (%d.%d.%d)\n",k,dd,mm,rr);
      while (getchar()!='\n');
   }
   printf("\n\n");
   system("pause");
   return 0;
}
```

```
Вводьте дати за форматом дд.мм.рррр
Для закінчення - букву
21
   Номер дня в році -  21 (21.1.2019)
21.03
   Номер дня в році -  80 (21.3.2019)
21.03.2000
   Номер дня в році -  81 (21.3.2000)
30ю03ю2020
   Номер дня в році -  90 (30.3.2000)
03.10.1932
   Номер дня в році - 277 (3.10.1932)
30.05.1959
   Номер дня в році - 150 (30.5.1959)
06.11.1996
   Номер дня в році - 311 (6.11.1996)
29.10.2001
   Номер дня в році - 302 (29.10.2001)
кінець
```

Прикалад 3.3. Обчислити наближено число π, застосовуючи метод Монте-Карло. Щоб працювати тільки з цілими додатними числами, треба розглянути четвертину круга в квадраті. При цьому сторона квадрата і радіус кола буде дорівнювати RAND_MAX=32767. Точки, які потрапляють на границі, треба враховувати. Обчислення провести кілька разів, вказуючи різну кількість точок, які генеруються. Обчислення виконувати до введення числа 0.

Виведемо формулу для розрахунків:



$$S_{круга} = \pi r^2;\ S_{сектора} = \frac{\pi r^2}{4};\ S_{квадрата} = r^2.$$

Тоді, ймовірність потрапляння точки в сектор:

$$p = \frac{S_{сектора}}{S_{квадрата}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4} \approx \frac{n}{N}.$$

Звідки

$$\pi \approx 4n/N,$$

де N — кількість усіх випробувань; n — кількість точок, які потрапили в сектор.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
// Обчислення числа Pi за методом Монте-Карло
  int x, y;  // згенеровані координати точки
  long long int N,   // кількість усіх згенерованих точок у квадраті
       n;   // кількість точок, які потрапили в четвертину круга
  double Pi;   // обчислене значення
  int i;   // лічильник згенерованих точок
  system("chcp 1251 & cls");
  srand(time(0));
  printf("    Наближене* обчислення значення числа Pi\n");
  printf("Вкажіть кількість генерацій випадкових точок");
  while (printf("\n  N="), scanf("%lld",&N), N) {
    n=0;
    for (i=0; i<N; i++)  {
      x=rand(); y=rand();  // згенеровань координати точки
      if (x*x+y*y<=RAND_MAX *RAND_MAX) n++;
    }
    Pi=4.*(double)n/(double)N;
    printf ("    число Pi=%.4f\n", Pi);
  }
  printf(" *Використано метод Монте-Карло");
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
   Наближене* обчислення
Вкажіть кількість генерації
  N=100
    число Pi=3.0000

  N=100
    число Pi=2.8800

  N=1000000
    число Pi=3.1432

  N=100000000
    число Pi=3.1414

  N=100000000
    число Pi=3.1415

  N=0
*Використано метод Монте-
```

Завдання. Написати програми, в яких використовуються вкладені цикли різних типів.

1. Побудувати:

— піраміду заввишки *n*;

```
    #
   ###
  #####
 #######
#########
```

```
#########
 #######
  #####
   ###
    #
```

```
#####
 ###
  #
 ###
#####
```

— перевернуту піраміду заввишки n;

— пісочний годинник заввишки $2n-1$.

2. Виловити з тексту цілі числа і надрукувати їх в стовпчик Число має бути останнім перед натисканням клавіші Enter.

Практичне заняття № 11 (*комп'ютерний практикум № 7*) **Проектування алгоритмів і програм циклічної структури (оператори циклів з передумовою і післяумовою)**

Мета заняття: набути практичних навичок розробляти алгоритми і програмувати засобами мови С циклічні процеси, використовуючи оператори циклів з передумовою чи післяумовою, працювати з дійсними числами ("плаваюча арифметика") і оформляти результати у вигляді таблиць, у яких інформація подається по рядках. Також ознайомитися з наслідками переповнення пам'яті при роботі з цілочисельними даними.

Завдання 1. Розробити алгоритм, подати узагальнену блок-схему (зобразивши в ній цикли while і do-while) і скласти програму розв'язування такої задачі: обчислити значення функції $f(x)$, де a і b можуть набувати довільних значень, аргумент x змінюється на довільно заданому проміжку $[x_1, x_2]$ із заданим кроком h.

При перевірці ОДЗ, якщо треба для врахування похибки роботи з "плаваючою арифметикою", використовувати досить мале число eps (при роботі з "плаваючою арифметикою" в загальному випадку точної рівності нулю чи іншому числу не може бути!). Результат видати на екран у таблиці (по рядках) з трьома парами колонок $x, f(x)$. Якщо функція при якомусь значенні аргумента не визначена (порушено ОДЗ), то в таблиці на місці результату подати "*не визн*" (зверніть увагу — кожна функція має по два випадки порушення ОДЗ). Якщо знайдене значення функції дуже велике (абсолютна величина перевищує певне значення), то в таблиці на місці результату подати "*переповн*" чи "*".

В алгоритмі передбачити перевірку правильності введення даних (границь проміжка і величини кроку) і виконання програми, доки дані не будуть введені правильно. Коментарі в програмі обов'язкові (17-25%). За алгоритмом провести розрахунки не менш ніж з трьома різними наборами вхідних даних.

Загальна схема програми має бути такою (нехай табулюється функція $f(x)=\frac{\sqrt{3x+a}}{\sin(2x-b)}+6e^{|x|}$):

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main() {
  double x1,x2,h,   // границі проміжка і крок
         a,b,x,     // параметри a і b і змінна x
         y;         // обчислене значення функції
  double eps=1e-14;   // точність обчислень
  int ind,   // чи правильно задано вхідні дані
      i,     // номер обчисленого значення
      k=3;   // кількість пар колонок
...
  printf("Табулювання функції:\n");
  printf("\tf(x)=sqrt(3x+a)/sin(2x-b)+6e^|x|\n");
     /* Введення вхідних даних */
  ind=1;
  do {   // доки не буде правильно введено дані
...
  } while (ind);
  printf("\tзначення параметрів a і b: ");
  scanf("%lf%lf",&a,&b);
  /* Проведення обчислень і формування таблиці */
...
  x=x1; i=0;
  // величина eps враховує похибку "плаваючої арифметики"
  while (x<=x2+eps) {
    i++;
```

```
    printf("| %6.2f",x);
    if (3*x+a<0 || fabs(sin(2*x-b))<eps)
      printf(" | не визн  ");
    else {   // обчислення значення і вивід у таблицю
      y=sqrt(3*x+a)/sin(2*x-b)+6*exp(fabs(x));
      // вивід значення функції або повідомлення "переповн"
...
    }
    if (i%k==0) printf("|\n");
    x+=h;
  }
     /* Доповнення таблиці порожніми клітинками */
...
  printf("\n\n");
  system("pause");
  return 0;
}
```

При роботі програми на екран має виводитися така інформація:

```
Табулювання функції:
        f(x)=sqrt(3x+a)/sin(2x-b)+6e^|x|
Вкажіть
        проміжок x1, x2: 3 -1
***Помилка: x1>x2
Вкажіть
        проміжок x1, x2: -1 3
        крок h: 10
***Помилка: h>x2-x1 або h<=0
Вкажіть
        проміжок x1, x2: -1 3
        крок h: 0.1
        значення параметрів a і b: 2.8 2
-------------------------------------------------------------
|    x   |   f(x)   |    x   |   f(x)   |    x   |   f(x)   |
-------------------------------------------------------------
|  -1.00 | не визн  |  -0.90 |    15.27 |  -0.80 |    14.78 |
|  -0.70 |    15.36 |  -0.60 |    28.06 |  -0.50 |     1.81 |
|  -0.40 |     5.17 |  -0.30 |     5.43 |  -0.20 |     5.13 |
|  -0.10 |     4.68 |  -0.00 |     4.16 |   0.10 |     4.82 |
|   0.20 |     5.48 |   0.30 |     6.15 |   0.40 |     6.81 |
|   0.50 |     7.43 |   0.60 |     7.94 |   0.70 |     8.16 |
|   0.80 |     7.50 |   0.90 |     2.95 |   1.00 | не визн  |
|   1.10 |    30.46 |   1.20 |    26.42 |   1.30 |    26.60 |
|   1.40 |    28.02 |   1.50 |    30.10 |   1.60 |    32.68 |
|   1.70 |    35.70 |   1.80 |    39.16 |   1.90 |    43.11 |
|   2.00 |    47.60 |   2.10 |    52.73 |   2.20 |    58.69 |
|   2.30 |    65.89 |   2.40 |    75.58 |   2.50 |    95.84 |
|   2.60 |    25.01 |   2.70 |    76.36 |   2.80 |    91.11 |
|   2.90 | переповн |   3.00 | переповн |        |          |
-------------------------------------------------------------
```

Варіанти завдань

1. $f(x)=\dfrac{\sqrt{ax}}{\cos(|b+x|)-1}+4\sin x$

2. $f(x)=\dfrac{\ln(|a-x|)}{\sin(x+b)}-9\cos x$

3. $f(x)=\dfrac{\ln(ax-1)}{\sin x-2b}+e^{-0,2|x|}$

4. $f(x)=\dfrac{\sqrt{4x+a}}{\sin(b+3x)}+\sin(|x-1|)$

5. $f(x)=\dfrac{\sqrt{3x+a}}{\sin(bx-1)}+\sin(|x|)$

6. $f(x)=\dfrac{\ln(ax+2)}{\cos(bx)-1}-3\sin(|x-1|)$

7. $f(x)=\dfrac{\ln(|ax+3|)}{\sin x+b}-\sin x$

8. $f(x)=\dfrac{\sqrt{x+a}}{\sin(x-b)}+\cos(|x|)$

9. $f(x)=\dfrac{\ln(bx+3)}{\cos(ax)-1}-7\sin(|x+2|)$

10. $f(x)=\dfrac{\sqrt{ax+9}}{\sin(b+x)}+\cos(|x|)$

11. $f(x)=\dfrac{\ln(a+x)}{\sin(|b+x|)}+\sin x$

12. $f(x)=\dfrac{\ln(|ax|)}{\sin(x-b)}+e^{-0,01|x+3|}$

13. $f(x)=\dfrac{\sqrt{5x+1}}{\sin(|bx|)}+4(x-a)$

14. $f(x)=\dfrac{\sqrt{x^2-a}}{\sin(bx)}+4\cos(|x|)$

15. $f(x)=\dfrac{\ln(|a+x|)}{\sin(bx)}-5\cos x$

16. $f(x)=\dfrac{\sqrt{2ax-1}}{\sin(x-b)}+e^{-0,02|x|+1}$

17. $f(x)=\dfrac{\ln(ax+4)}{\sin(3x)-b}-5e^{-0,02|x|}$

18. $f(x)=\dfrac{\ln(|ax|)}{\sin x-b}+2e^{-0,3|x|}$

19. $f(x)=\dfrac{\sqrt{a+x}}{\sin(|bx|)}-5\sin x$

20. $f(x)=\dfrac{\ln(ax+2)}{\sin x-2b}+e^{-0,3|x|}$

21. $f(x)=\dfrac{\sqrt{ax-2}}{\sin(x-b)}+\cos(|x-1|)$

22. $f(x)=\dfrac{\sqrt{3-ax}}{\sin(1+bx)}+\cos(|x|)$

23. $f(x)=\dfrac{\ln(ax-1)}{\sin(3x+b)}+e^{-0,03|x-1|}$

24. $f(x)=\dfrac{\sqrt{x+a}}{\sin(b-2x)}+\sin(|3+x|)$

25. $f(x)=\dfrac{\ln(ax+4)}{\sin(5x)-b}-3\cos(|x|-1)$

26. $f(x)=\dfrac{\sqrt{3x-a}}{\sin(b-2x)}+\sin(|x|)$

27. $f(x)=\dfrac{\ln(|a+x|)}{\sin(bx+1)}-2e^{-0,1|x+1|}$

28. $f(x)=\dfrac{\ln(ax-2)}{\sin(3x)+b}-3\sin(|x+1|)$

29. $f(x)=\dfrac{\sqrt{ax-1}}{\sin(3x-b)}+7\cos(|x|)$

30. $f(x)=\dfrac{\ln(ax+3)}{\cos(bx)-1}-4\sin(|x|)$

Завдання 2. Виконати програму обчислення значення факторіала числа *n*, послідовно задаючи тип char, unsigned char, short int, unsigned short int , int, unsigned int, long long int, unsigned long long int змінної factorial:

```
#include <stdio.h>
#include <stdlib.h>
int main() {
/* Обчислення значення факторіала */
  int n,i;
  char factorial;
  printf("Vvedit znachennia n ");
  scanf("%d",&n);
  factorial=1;
  for (i=2; i<=n; i++) {
    factorial*= i;
    printf("\t i=%2d - %20d\n",i,factorial);
  }
  printf("\n\n");
  system("pause");
  return 0;
}
```

Діапазони можливих значень змінних вказаних типів подано в таблиці:

Тип	Діапазон значень	Пам'ять (байтів)
char	–128 … 127 (-2^7 … 2^7-1)	1
unsigned char	0 … 255 (2^8-1)	1
short int	–32768 … 32767 (-2^{15} … $2^{15}-1$)	2
unsigned short int	0 … 65535 ($2^{16}-1$)	2
int	–2147483648 … 2147483647 (-2^{31} … $2^{31}-1$)	4
unsigned int	0 ... 4294967295 ($2^{32}-1$)	4
long long int	–9223372036854775808 … 9223372036854775807 (-2^{63} … $2^{63}-1$); *стандарт С99*	8
unsigned long long int	0 ... 18446744073709551615 (0 … $2^{64}-1$); *стандарт С99*	8

Дослідити, при якому найменшому значенні *n* буде одержано від'ємний чи неправильний результат (правильність перевірити на калькуляторі; правильно підбирати формати виведення). Результати досліджень подати в таблиці:

Тип змінної	При якому *n* одержано останнє правильне (чи перше неправильне) значення	Формат виведення
char	одержано останнє правильне значення *n*! при *n*=5	%d
. . .		

У звіті пояснити, чому так відбувається.

Практичне заняття № 12 (*комп'ютерний практикум № 8*)
Проектування алгоритмів і програм з вкладеними циклами

Мета заняття: набути практичних навичок розробляти алгоритми і програмувати засобами мови С циклічні процеси з вкладеними циклами, закріпити навички роботи з дійсними числами ("плаваюча арифметика") і оформляти результати у вигляді таблиць.

Завдання. Розробити алгоритм, намалювати узагальнену блок-схему і написати програму обчислення значення функції $f(x)$, де аргумент x змінюється на довільно заданому проміжку $[x_1, x_2]$ з кроком h. При знаходженні степеня числа a^n скористатися додатковою змінною, в якій буде накопичуватися результат (див. приклад):

Приклад. Обчислити суму $\sum_{n=2}^{5} \frac{3^{n+1} - x}{n}$:

```
...
s=0;
an=9;
for (n=2; n<=5; n++) {
  an*=3;
  s+=(an-x)/n
}
...
```

Результат видати на екран у таблиці з трьома парами колонок $x, f(x)$. Якщо функція при якомусь значенні аргумента не визначена, то в таблиці на місці результату подати "*не визн*" (при цьому суму не рахувати). Якщо функція набуває значення більшого від 10^6 (або іншого числа), то на місці результату подати "*переповн*" або "*".

В алгоритмі передбачити перевірку правильності введення даних (границь проміжку і величини кроку) і виконання програми, доки дані не будуть введені правильно. Коментарі в програмі обов'язкові (17-25%). За алгоритмом провести розрахунки не менш ніж з трьома різними наборами вхідних даних.

Варіанти завдань

1. $f(x)=\sqrt{\frac{x+2}{x-1}}\sum_{n=1}^{5}\frac{2^n-x}{n^2+1}+\ln(x+5)$

2. $f(x)=\frac{3+\ln(x+3)}{3-x}\sum_{n=1}^{7}\frac{2^{n+1}+x}{n^2}+\sqrt{x^2-1}$

3. $f(x)=\sqrt{\frac{x+3}{x^2-1}}\sum_{n=0}^{6}\frac{2^n+3x}{n+2}-\ln|x-3|$

4. $f(x)=\frac{\ln x-4}{x-2}\sum_{n=1}^{3}\frac{4^n-x}{n+5}-\sqrt{\ln|x-6|}$

5. $f(x)=\frac{2-x}{1-\sqrt{x+3}}\sum_{n=2}^{5}\frac{4^n+x}{n^2}-\ln|x-1|$

6. $f(x)=\frac{\ln x+3}{x-1}\sum_{n=3}^{5}\frac{2^n+x}{n-2}+\sqrt{20-x}$

7. $f(x)=\sqrt{\frac{x+1}{x-3}}\sum_{n=2}^{5}\frac{3^n+x}{n-1}+\ln(x+4)$

8. $f(x)=\frac{\ln(x+2)+4}{x-3}\sum_{n=1}^{4}\frac{3^n+x}{4n-1}+\sqrt{7-x}$

9. $f(x)=\sqrt{\frac{x-3}{x^2-4}}\sum_{n=0}^{3}\frac{6^n-x}{n+1}+\ln|10-x|$

10. $f(x)=\frac{\ln(x+1)+3}{2-x}\sum_{n=2}^{7}\frac{2^n+x}{3n-1}+\sqrt{15-x}$

11. $f(x)=\frac{x+1}{3-\sqrt{x^2-1}}\sum_{n=3}^{5}\frac{4^n+x}{n^2+1}-4\ln(x+5)$

12. $f(x)=\frac{\ln(x+2)}{x-3}\sum_{n=1}^{5}\frac{3^n+x}{n}+\sqrt{x^4-1}$

13. $f(x)=\frac{1-2x}{2-\sqrt{x+3}}\sum_{n=0}^{5}\frac{2^n+x}{n+2}-6\ln|x-3|$

14. $f(x)=\frac{\ln(x-5)+2}{x-1}\sum_{n=2}^{4}\frac{3^n-x}{n+5}+2\sqrt{\ln|x-4|}$

15. $f(x)=\sqrt{\frac{x^2-2}{x-3}}\sum_{n=2}^{5}\frac{4^n+x}{4n+3}+7\ln|x-7|$

16. $f(x)=\frac{\ln x-2}{x-1}\sum_{n=1}^{4}\frac{5^n+x}{n+3}+\sqrt{\ln|x-7|}$

17. $f(x)=\sqrt{\frac{x-1}{x-2}}\sum_{n=2}^{7}\frac{3^{n+1}+x}{n-1}+\ln|x|$

18. $f(x)=\frac{\ln(x+5)}{x+2}\sum_{n=0}^{6}\frac{2^n+x}{n+3}+\sqrt{\ln|x-3|}$

19. $f(x)=\dfrac{1-5x}{\sqrt{x+1}-2}\sum\limits_{n=2}^{4}\dfrac{4^n+x}{(n-1)^2}-\ln(x^2)$

20. $f(x)=\dfrac{\ln(x+4)}{x-3}\sum\limits_{n=2}^{5}\dfrac{4^n+x}{4n-1}+\sqrt{|x|-1}$

21. $f(x)=\dfrac{2x-1}{1-\sqrt{x+1}}\sum\limits_{n=1}^{4}\dfrac{5^n+x}{(n+1)^2}-\ln((x-1)^2)$

22. $f(x)=\dfrac{\ln(x+2)}{5-x}\sum\limits_{n=1}^{4}\dfrac{3^n+x}{2n-1}+\sqrt{\ln|x|}$

23. $f(x)=\sqrt{\dfrac{x-2}{x+1}}\sum\limits_{n=0}^{4}\dfrac{3^n+x}{n+1}-\ln(x+4)$

24. $f(x)=\dfrac{\ln(x+3)}{x-7}\sum\limits_{n=2}^{5}\dfrac{4^n+x}{4n-1}+\sqrt{x^2-2}$

25. $f(x)=\sqrt{\dfrac{x+2}{x^2-1}}\sum\limits_{n=2}^{6}\dfrac{3^{n+1}+x}{n}-x\ln|x-5|$

26. $f(x)=\dfrac{\ln(x-2)+2}{x-3}\sum\limits_{n=2}^{3}\dfrac{5^n+x}{4n+3}+\sqrt{\ln|x-9|}$

27. $f(x)=\dfrac{1-x}{1-\sqrt{x+2}}\sum\limits_{n=0}^{8}\dfrac{2^n+x}{2n+1}-5\ln|x-4|$

28. $f(x)=\dfrac{2-\ln(x+4)}{x+2}\sum\limits_{n=2}^{4}\dfrac{3^n+x}{3n-1}+4\sqrt{\ln|x|}$

29. $f(x)=\dfrac{\sqrt{x+2}}{x-3}\sum\limits_{n=1}^{5}\dfrac{4^n-5x}{n^2+n+1}+\ln(x^2-1)$

30. $f(x)=\dfrac{\ln(x-1)}{4-x}\sum\limits_{n=0}^{6}\dfrac{3^n-2x}{n+2}+\sqrt{8-x}$

Практичне заняття № 13 (*комп'ютерний практикум № 9*)

Проектування ітераційних алгоритмів

Мета заняття: набути практичних навичок проектування і програмування засобами мови С ітераційних алгоритмів з використанням циклів з післяумовою; набути навичок оптимально програмувати вирази, які передбачають обчислення факторіалів.

Завдання. Розробити алгоритм і написати програму обчислення значення функції $f(x)$, розкладеної в степеневий ряд. Призначення кожної змінної пояснити в коментарях. Обчислення суми членів ряду проводити доти, доки абсолютна величина члена ряду не стане меншою від ε (наприклад, $\varepsilon=10^{-6}$, $\varepsilon=10^{-9}$, …). При цьому порахувати кількість виконаних кроків ітерації (скільки членів ряду ввійшло в суму). Крім того, для підстраховки від зациклювання, яке може виникнути через некоректні вхідні дані, встановити ліміт кількості кроків. Якщо вихід із циклу відбувся через вичерпання ліміту, то видати про це повідомлення.

При *обчисленні наступного члена ряду* використовувати попередній член чи його частину, а не організовувати додатковий цикл для повного його обчислення.

Порівняти (знайти абсолютне значення різниці) обчислене з використанням ряду наближене значення функції із значенням, обчисленим за формулою функції.

Як результат роботи видати: обчислене наближене значення функції, кількість кроків ітерації, обчислене за формулою значення функції, абсолютну різницю наближеного і "точного" значень функції. В алгоритмі передбачити перевірку правильності введення даних — програму виконувати доти, поки дані не будуть введені правильно. Коментарі в програмі обов'язкові (17-25%). За алгоритмом провести розрахунки не менш ніж з трьома різними наборами вхідних даних: при різних значеннях x, ε і ліміту кількості кроків.

Варіанти завдань

Зауваження. Хоч ряди, для яких не вказано проміжок збіжності, збігаються на всій числовій прямій, при розрахунках брати проміжок $[-10,10]$.

1. $\dfrac{1-\sqrt{1-4x}}{2}=\sum\limits_{n=0}^{\infty}\dfrac{(2n)!x^{n+1}}{n!(n+1)!}$ при $x\in(-0{,}25;0{,}25)$

2. $1+(x-1)e^x=\sum\limits_{n=0}^{\infty}\dfrac{(n+1)x^{n+2}}{(n+2)!}$

3. $\cos x-1=\sum\limits_{n=1}^{\infty}\dfrac{(-1)^n x^{2n}}{(2n)!}$

4. $\sin x=\sum\limits_{n=0}^{\infty}\dfrac{(-1)^n x^{2n+1}}{(2n+1)!}$

5. $\dfrac{e^x+e^{-x}+2\cos x}{4}=\sum\limits_{n=0}^{\infty}\dfrac{x^{4n}}{(4n)!}$

6. $chx-1=\dfrac{e^x+e^{-x}}{2}-1=\sum\limits_{n=1}^{\infty}\dfrac{x^{2n}}{(2n)!}$

7. $chx=\dfrac{e^x+e^{-x}}{2}=\sum\limits_{n=0}^{\infty}\dfrac{x^{2n}}{(2n)!}$

8. $\cos x=\sum\limits_{n=0}^{\infty}\dfrac{(-1)^n x^{2n}}{(2n)!}$

9. $\dfrac{1}{2}-x-\dfrac{\sqrt{1-4x}}{2}=\sum\limits_{n=1}^{\infty}\dfrac{(2n)!x^{n+1}}{(n!)^2(n+1)}$ при $x\in[-0{,}25;0{,}25)$

10. $x\cos x=\sum\limits_{n=0}^{\infty}\dfrac{(-1)^n x^{2n+1}}{(2n)!}$

11. $(x-1)e^x+1=\sum\limits_{n=0}^{\infty}\dfrac{x^{n+2}}{n!(n+2)}$

12. $shx - x = \dfrac{e^x - e^{-x}}{2} - x = \sum\limits_{n=1}^{\infty} \dfrac{x^{2n+1}}{(2n+1)!}$

13. $\dfrac{1}{(1-4x)\sqrt{1-4x}} - 1 - 6x = \sum\limits_{n=2}^{\infty} \dfrac{(2n+1)!x^n}{(n!)^2}$ при $x \in [-0{,}25; 0{,}25)$

14. $1 + (x-1)e^x - \dfrac{x^2}{2} = \sum\limits_{n=1}^{\infty} \dfrac{(n+1)x^{n+2}}{(n+2)!}$

15. $\dfrac{1}{2} - \dfrac{\sqrt{1-4x}}{2} = \sum\limits_{n=0}^{\infty} \dfrac{(2n)!x^{n+1}}{(n!)^2(n+1)}$ при $x \in [-0{,}25; 0{,}25)$

16. $\sin x - x = \sum\limits_{n=1}^{\infty} \dfrac{(-1)^n x^{2n+1}}{(2n+1)!}$

17. $\dfrac{e^x - e^{-x} - 2\sin x}{4} = \sum\limits_{n=0}^{\infty} \dfrac{x^{4n+3}}{(4n+3)!}$

18. $\dfrac{1-2x-\sqrt{1-4x}}{4} = \sum\limits_{n=0}^{\infty} \dfrac{(2n+1)!x^{n+2}}{n!(n+2)!}$ при $x \in [-0{,}25; 0{,}25]$

19. $shx = \dfrac{e^x - e^{-x}}{2} = \sum\limits_{n=0}^{\infty} \dfrac{x^{2n+1}}{(2n+1)!}$

20. $x\cos 3x = \sum\limits_{n=0}^{\infty} \dfrac{(-1)^n 3^{2n} x^{2n+1}}{(2n)!}$

21. $\dfrac{1}{(1-4x)\sqrt{1-4x}} = \sum\limits_{n=0}^{\infty} \dfrac{(2n+1)!x^n}{(n!)^2}$ при $x \in [-0{,}25; 0{,}25)$

22. $\dfrac{e^x - e^{-x} + 2\sin x}{4} = \sum\limits_{n=0}^{\infty} \dfrac{x^{4n+1}}{(4n+1)!}$

23. $\dfrac{(1-4x)\sqrt{1-4x} + 6x(1-x) - 1}{12} = \sum\limits_{n=0}^{\infty} \dfrac{(2n+2)!x^{n+3}}{(n+1)!(n+3)!}$ при $x \in [-0{,}25; 0{,}25]$

24. $\ln 2 - \ln(1+\sqrt{1-4x}) = \sum\limits_{n=0}^{\infty} \dfrac{(2n+1)!x^{n+1}}{\left((n+1)!\right)^2}$ при $x \in [-0{,}25; 0{,}25]$

25. $x - \sin x = \sum\limits_{n=2}^{\infty} \dfrac{(-1)^n x^{2n-1}}{(2n-1)!}$

26. $\dfrac{1}{\sqrt{1-4x}} = \sum\limits_{n=0}^{\infty} \dfrac{(2n)!x^n}{(n!)^2}$ при $x \in [-0{,}25; 0{,}25)$

26. $\dfrac{(1-4x)\sqrt{1-4x} + 6x - 1}{12} = \sum\limits_{n=0}^{\infty} \dfrac{(2n)!x^{n+2}}{n!(n+2)!}$ при $x \in [-0{,}25; 0{,}25]$

28. $\ln \dfrac{2}{1+\sqrt{1-4x}} = \sum\limits_{n=1}^{\infty} \dfrac{(2n-1)!x^n}{\left(n!\right)^2}$ при $x \in [-0{,}25; 0{,}25]$

29. $a^x = \sum\limits_{n=0}^{\infty} \dfrac{(\ln a)^n x^n}{n!}$ $(a > 0,\ a \neq 1)$

30. $e^x - 1 = \sum\limits_{n=0}^{\infty} \dfrac{x^{n+1}}{(n+1)!}$

При знаходженні деяких поданих в завданні сум точність залежить від того, як записати вираз обчислення значення загального члену ряду; наприклад, запис `xn/((4*n+3)*(4*n+2)*(4*n+1)*4*n)` дає меншу точність, ніж запис `xn/(4*n+3)/(4*n+2)/(4*n+1)/4/n`.

Практичне заняття № 14
Розробка програм модульної структури: функції без параметрів і з параметрами. Передача параметрів. Рекурсія

Мета заняття: закріпити лекційний матеріал і поглибити знання щодо розробки програм модульної структури, програмування функцій без параметрів і з параметрами; розібратися в принципах роботи локальних, глобальних і статичних змінних; розібратися в механізмах передачі параметрів за значенням і за адресою; розглянути рекурсивні функції.

1. Програми модульної структури. Використання модулів дає можливість писати добре структуровані програми, в яких легко відслідковується основний алгоритм, які неважко зрозуміти тим, хто їх читає, які легко налагоджувати і які менш чутливі до помилок програмування. Крім того, окремі модулі можна багаторазово використовувати в тій самій програмі і в різних програмах. У мові програмування С модулі оформляються як функції, всі функції є зовнішніми. Зв'язок між функціями здійснюється через параметри, повернені значення і глобальні змінні. При цьому головна функція описує алгоритм без деталізації, а решта функцій деталізують алгоритм.

Приклад 1.1. Пошук у тексті цілих чисел і виведення їх на екран. Натискання клавіші Enter означає кінець вводу. Останнім перед кінцем вводу має бути число (якщо останнім перед Enter буде не число, то ввід і обробка тексту продовжаться далі). Замість клавіші Enter можна використати будь-який символ, наприклад, * чи #.

При розробці алгоритму модульної структури для розв'язання цієї задачі можна виділити дві основні функції — пропуск символів, відмінних від цифр, і формування цілого числа.

```
/* Пошук у тексті цілих чисел */
#include <stdio.h>
#include <stdlib.h>
void ignoresymb(void);   // прототип функції ignoresymb
int number (void);   // прототип функції number
unsigned char sym;   // символ з введеного тексту (глобальна змінна)
int main() {    // Головна функція
  int n;   // ціле число
  system("chcp 1251 & cls");
  printf("Введіть текст з числами\n");
  sym=' ';
  while (sym!='\n') {
    ignoresymb();
    n=number();
    printf("%-8d",n);
  }
  printf("\n\n");
  system("pause");
  return 0;
}
void ignoresymb(void) {   // пропуск символів, відмінних від цифр
  while ((sym=getchar()) <'0' || sym>'9');
}
int number (void) {   // формування числа
  int ch;
  ch=0;
  while (sym>='0' && sym<='9') {
    ch=ch*10+sym-'0';
    sym=getchar();
  }
  return ch;
}
```

```
Введіть текст з числами
Найбільша кількість псевдонімів була в О. Конисього - 141, в І. Франка -99,
О. Маковея - 56, у Лесі Українки - 1
141     99      56      1

Press any key to continue . . .
```

```
Введіть текст з числами
   Мовознавець Іван Ющук у статті "12 фактів про давність української мови"
(20.07.2006) пише, що українська мова вже в 6-7 ст. мала окреслено сучасні
обриси (У 2017 І.П. Ющук видав "Словник української мови 6 століття")
12      20      7       2006    6       7       2017    6
   Україна близько 700 років була розчленована між різними державами, які
насаджували їй свої мови. У складі радянської імперії Україна перебувала з
1922 до 1991
700     1922    1991

Press any key to continue . . .
```

Обидві функції ignoresymb і number не мають параметрів. Функція ignoresymb не має локальних змінних; функція number має одну локальну змінну ch. Функція number повертає обчислене значення. Змінна sym є глобальною і використовується у всіх функціях (включаючи й головну). Проте хороший стиль програмування вимагає не використовувати глобальні змінні, якщо немає виняткової потреби (у даному випадку змінна використовується у всіх функціях).

Приклад 1.2. Впорядкування двох чисел за зростанням: вводяться два числа; якщо треба, то вони впорядковуються за зростанням, якщо ні, то виводиться повідомлення, що задано впорядковані числа.

```
/* Впорядкування двох цілих чисел за зростанням */
#include <stdio.h>
#include <stdlib.h>
int sort(int *a, int *b);   // прототип функції
int main() {   // Головна функція
  int n, m;   // вхідні значення
  system("chcp 1251 & cls");
  printf("\tВпорядкування за зростанням двох чисел\
    \nВведіть значення n і m\n");
  while (scanf("%d%d",&m,&n))
  if (sort(&m,&n))
    printf("   Результат впорядкування: %d %d\n",m,n);
```

```
  else
    printf("   Задані числа впорядковані\n");
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
       Впорядкування за зростанням двох чисел
Введіть значення n і m
2 5
   Задані числа впорядковані
3 -1
   Результат впорядкування: -1 3
кінець
```

```
int sort(int *a, int *b) { // Функція впорядкування двох чисел
  int c,   // робоча змінна
      r;   // індикатор результату
  if (*a<=*b) r=0;
  else { c=*a; *a=*b; *b=c; r=1;}
  return r;
}
```

Функція sort має два параметри a і b, вона повертає їхні змінені значення, а також своє значення (індикатор), функція має дві локальні змінні c і r.

Приклад 1.3. Написати програму скорочення дробів.

Для скорочення дробів треба знайти найбільший спільний дільник (НСД). Розглянемо інший (не Евклідів) алгоритм:

a%b → r1

b%r1 → r2

r1%r2 → r3 і т.д.

і, якщо остача від ділення дорівнює 0, то останній знаменник є НСД.

```
/* Скорочення дробів */
#include <stdio.h>
#include <stdlib.h>
int nsd(int ch, int zn);   // прототип функції nsd
// Головна функція
int main() { // Скорочення дробів
  int a,b,c;   // чисельник, знаменник і НСД
  system("chcp 1251 & cls");
  printf("Введіть дріб a/b: ");
  scanf("%d/%d",&a,&b);
  c=nsd(a,b);
```

```
  if (c>1) {
    a=a/c;
    b=b/c;
  }
  printf("НСД=%d. Скорочений дріб: %d/%d",c,a,b);
  printf("\n\n");
  system("pause");
  return 0;
}
// Функція знаходження НСД
int nsd(int ch, int zn) {
  int r;   // остача від ділення
  while (zn) {   // zn!=0
    r=ch%zn;
    ch=zn;
    zn=r;
  }
  return ch;
}
```

```
Введіть дріб a/b: 6/8
НСД=2. Скорочений дріб: 3/4
```

```
Введіть дріб a/b: 35/21
НСД=7. Скорочений дріб: 5/3
```

```
Введіть дріб a/b: 1/15
НСД=1. Скорочений дріб: 1/15
```

```
Введіть дріб a/b: 0/8
НСД=8. Скорочений дріб: 0/1
```

Функція nsd має два параметри ch і zn, значення яких не змінює; має одну локальну змінну r; вона повертає обчислене значення НСД.

Завдання. Розробити програми модульної структури

1. Впорядкувати три числа *a*, *b* і *c* за зростанням, використовуючи функцію впорядкування двох чисел *a* і *b* за зростанням. Програма, якщо було задано впорядковані числа, має видавати повідомлення “Задані числа впорядковані”, інакше виводити результат впорядкування. Глобальні змінні не використовувати.

2. Обчислити значення кількості сполук C_n^m (при введенні даних значення *n* повинно бути не більшим від 20).

2. Передача параметрів за значенням і за адресою. Статичні змінні. При передачі параметра за значенням відповідним фактичним параметром може бути вираз (у широкому розумінні) відповідного типу. Під час виконання викликаної функції ніякі зміни значення формального параметра не впливають на значення відповідного фактичного параметра. Після закінчення

роботи викликаної функції фактичний параметр буде мати те саме значення, яке мав до початку роботи функції, незалежно від того, що відбувалося з формальним параметром.

При передачі параметра за адресою (за посиланням) фактичним параметром має бути вказівник (покажчик) на змінну, тобто адреса змінної. Вказівники на змінні (крім масивів) записуються з операцією & взяття адреси. Формальний параметр повинен бути оголошений як вказівник: *тип *ім'я змінної вказівника*. Унарна операція * розіменування вказівника, застосована до вказівника, забезпечує доступ до вмісту комірки пам'яті, на яку вказує вказівник. Усі зміни формального параметра відображаються на фактичному параметрі. При цьому фактичним параметром має бути вказівник (покажчик) на змінну, тобто адреса змінної. Відповідним формальним параметром має бути вказівник, тип якого точно збігається з типом змінної, адреса якої передається.

Приклад 2.1. Знайти кути трикутника за відомими трьома сторонами. При цьому написати функціюм виведення інформації про програму, введення даних і перевірки існування трикутника, обчислення кута за сторонами (з викликом функції обчислення конкретного кута), виведення результатів. У програмі використати: формулу косинусів $a^2 = b^2 + c^2 - 2bc\cos A$; перехід від радіанів до градусів $\varphi_\circ = \varphi_r \cdot 180/\pi$. Набір на клавіатурі позначення градусів — Alt+0176; значка авторського права — Alt+0169.

```
/* Обчислення величин кутів трикутника */
#include <stdio.h>
#include <stdlib.h>
#define _USE_MATH_DEFINES
#include <math.h>
void inform();
void vvid(double *a, double *b, double *c);
void kutyABC(double a, double b, double c,
             double *A, double *B, double *C);
double kut(double a, double b, double c);
void vyvid(double A, double B, double C);
/* Головна функція - організовує обчислення
```

```
 * величин кутів для багатьох трикутників */
int main() {
  double a,b,c,A,B,C;
  int ind;
  char vidp;
  system("chcp 1251 & cls");
  inform();
  ind=1;
  while (ind) {
    fseek(stdin,0,SEEK_END);
    vvid(&a,&b,&c);
    kutyABC(a,b,c,&A,&B,&C);
    vyvid(A,B,C);
    printf("Продовжувати роботу (Y - так)? ");
    fseek(stdin,0,SEEK_END);
    vidp=getchar();
    fseek(stdin,0,SEEK_END);
    if (!(vidp=='Y'||vidp=='y')) ind=0;
  }
  printf("\n\n");
  system("pause");
  return 0;
}
/* Інформація про програму */
void inform() {
  printf("Програма обчислення кутів трикутника\
    за довжинами його сторін\n");
  printf("Для обчислення величини кута використано\
     формулу косинусів:\n");
  printf("\t\tcosA=(b^2+c^2-a^2)/2bc\n");
  printf("© Ющук-Кублій Л.І.\n\n");
}
/* введення даних, перевірка їхньої правильності */
void vvid(double *a, double *b, double *c) {
  int ind;
  ind=1;
```

```
  do {
    printf("Введіть a, b, c  ");
    /* Виловлювання механічних помилок вводу */
    while (scanf("%lf %lf %lf",a,b,c)<3) {
     printf("Невірний формат даних.\nПовторіть ввід a, b, c  ");
      fseek(stdin,0,SEEK_END);
    }
    /* Перевірка умови існування трикутника */
    if (*a<=0 || *b<=0 || *c<=0 || *a+*b <= *c
        || *a+*c <= *b || *b+*c <= *a)
      printf("   Трикутник не існує\n");
    else ind=0;
  } while (ind);
}
/* Знаходження кутів трикутника за вказаними сторонами */
void kutyABC(double a, double b, double c,
             double *A, double *B, double *C) {
  *A=kut(a,b,c);
  *B=kut(b,c,a);
  *C=kut(c,a,b);
}
/* Обчислення кута A навпроти сторони a
 * за вказаними сторонами трикутника a, b, c */
double kut(double a, double b, double c) {
  return acos((b*b+c*c-a*a)/(2*b*c))*180./M_PI;
}
/* Виведення значень кутів A, B, C,
 * які спираються відповідно на сторони a, b, c */
void vyvid(double A, double B, double C) {
  printf("Навпроти сторони\n");
  printf("   a кут %0.2f°\n", A);
  printf("   b кут %0.2f°\n", B);
  printf("   c кут %0.2f°\n", C);
  printf("Похибка при обчисленні кутів %g°\n\n", fabs(180.-A-B-C));
}
```

```
Програма обчислення кутів трикутника за довжинами його сторін
Для обчислення величини кута використано формулу косинусів:
                cosA=(b^2+c^2-a^2)/2bc
© Ющук-Кублій Л.І.

Введіть a, b, c  1, 2, 3
Невірний формат даних.
Повторіть ввід a, b, c  1 2 3
   Трикутник не існує
Введіть a, b, c  5 6 7
Навпроти сторони
   a кут 44.42°
   b кут 57.12°
   c кут 78.46°
Похибка при обчисленні кутів 0°

Продовжувати роботу (Y - так)? y
Введіть a, b, c  7 9 5
Навпроти сторони
   a кут 50.70°
   b кут 95.74°
   c кут 33.56°
Похибка при обчисленні кутів 7.10543e-015°

Продовжувати роботу (Y - так)? Y
Введіть a, b, c  3 4 5 прямокутний
Навпроти сторони
   a кут 36.87°
   b кут 53.13°
   c кут 90.00°
Похибка при обчисленні кутів 0°

Продовжувати роботу (Y - так)? Y
Введіть a, b, c  1 1 1 рівнобічний
Навпроти сторони
   a кут 60.00°
   b кут 60.00°
   c кут 60.00°
Похибка при обчисленні кутів 1.42109e-014°

Продовжувати роботу (Y - так)? ні


Press any key to continue . . .
```

Якщо в функції vvid замість блоку перевірки існування трикутника помістити звернення до функції, яка виконує таку перевірку, то в цю функцію можна передавати параметри або за значенням, або за адресою. При цьому програмні коди будуть відрізнятися. Розглянемо обидва випадки.

а). Передача параметрів за значенням. Звернення до функції має вигляд:

```
ind=prav_zn(*a,*b,*c);    // передаються значення
```

Код функції буде таким:

```
/* перевірка умови існування трикутника */
int prav_zn(double a, double b, double c) { // параметри-значення
  int ind;
  ind=1;
```

```
  if (a<=0 || b<=0 || c<=0 ||a+b <= c || a+c <= b || b+c <= a)
    printf("   Трикутник не існує\n");
  else ind=0;
  return ind;
 }
```

б). Передача параметрів за адресою. Звернення до функції має вигляд:

```
ind=prav_ad(a,b,c);   // передаються адреси
```

Код функції буде таким:

```
/* перевірка умови існування трикутника */
int prav_ad(double *a, double *b, double *c) { // параметри-адреси
  int ind;
  ind=1;
  if (*a<=0 || *b<=0 || *c<=0 ||*a+*b <= *c
      || *a+*c <= *b || *b+*c <= *a)
    printf("   Трикутник не існує\n");
  else ind=0;
  return ind;
 }
```

Змінні, описані з ключовим словом static — статичні. Для статичної змінної компілятор виділяє постійне місце в пам'яті і вона існує до закінчення роботи програми. Статична змінна доступна лише в тій функції, де вона визначена (тобто це локальна змінна, яка зберігає своє значення між викликами функції). Локальна статична змінна ініціалізується тільки один раз. Явно не ініціалізована статична змінна неявно ініціалізується нулем.

Приклад 2.2. Використання статичної змінної.

```
#include <stdio.h>
#include <stdlib.h>
int hello();
int main() {
  system("chcp 1251 & cls");
  printf(" - %d-й виклик\n", hello());
  printf(" - %d-й виклик\n", hello());
  printf(" - %d-й виклик\n", hello());
  printf("\n\n");
```

```
  system("pause");
  return 0;
}
int hello () {
  static int n = 1;
  printf(" Привіт!");
  return n++;
}
```

```
Привіт! - 1-й виклик
Привіт! - 2-й виклик
Привіт! - 3-й виклик
```

Завдання. 1. Розробити два варіанти програми модульної структури (з використанням глобальних змінних і з використанням фактичних-формальних параметрів), яка має функцію введення, функцію обчислення і функцію виведення, для обчислення значення

$$f(x)=\begin{cases}\dfrac{9}{ax} & \text{при } x\in[1,3),\\ \left|ax^2+x+b\right| & \text{при } x=3\end{cases}$$

при довільних значеннях параметрів a і b і незалежної змінної x.

2. Визначити, яке значення буде виведене після виконання кожної з програм (свою відповідь перевірте, виконавши програми на комп'ютері):

а)
```
#include <stdio.h>
#include <stdlib.h>
int x;
void f1();
int main() {
  x=0;
  f1();
  printf("%d",x);
  printf("\n\n");
  system("pause");
  return 0;
}
void f1() {
  x=1;
}
```

б)
```
#include <stdio.h>
#include <stdlib.h>
void f2();
int main() {
  int x;
  x=0;
  f2();
  printf("%d",x);
  printf("\n\n");
  system("pause");
  return 0;
}
void f2() {
  int x;
  x=1;
}
```

в)
```
#include <stdio.h>
#include <stdlib.h>
void f3(int *x);
int main() {
  int x;
  x=0;
  f3(&x);
  printf("%d",x);
  printf("\n\n");
  system("pause");
  return 0;
}
void f3(int *y) {
  *y=1;
}
```

г)
```
#include <stdio.h>
#include <stdlib.h>
void f4(int x);
int main() {
  int x;
  x=0;
  f4(x);
  printf("%d",x);
  printf("\n\n");
  system("pause");
  return 0;
}
void f4(int y) {
  y=1;
}
```

ґ)
```
#include <stdio.h>
#include <stdlib.h>
int x;
void f5(int *x);
int main() {
  x=0;
  f5(&x);
  printf("%d",x);
  printf("\n\n");
  system("pause");
  return 0;
}
void f5(int *y) {
  *y=1;
  x=x+1;
}
```

д)
```
#include <stdio.h>
#include <stdlib.h>
void f6(int *x);
int main() {
  int x;
  x=0;
  f6(&x);
  {int x; x=6;  }
  printf("%d",x);
  printf("\n\n");
  system("pause");
  return 0;
}
void f6(int *y) {
  *y=1;
 }
```

Пояснити, чому одержано саме таке значення.

Зауваження. При використанні вказівників у виразах для уникнення помилок, пов’язаних з пріоритетами операцій, треба розіменування вказівника *p брати в дужки: (*p). Наприклад, *p++ працює неправильно, хоч обидві

операції * і ++ унарні і мають однаковий пріоритет і тому повинні виконуватися зліва направо, тобто спочатку *p, а потім інкремент. При цьому правильно працює вираз (*p)++. Оператор *p =*p +1 працює правильно.

4. Рекурсивні функції. Рекурсивна функція — це функція, яка викликає сама себе. Застосування рекурсії дає можливість компактно подати алгоритм. Створення рекрсивної функції, як правило, передбачає зведення задачі до такої самої задачі, але меншої складності чи меншої розмірності. У рекурсивній функції обов'язково повинен бути оператор повернення без рекурсивного виклику.

Приклад 4.1. Перевірка правильності введення значення з використанням рекурсії.

```
#include <stdio.h>
#include <stdlib.h>
void vvid_n(int *n);
int main() {
  int n;
  system("chcp 1251 & cls");
  vvid_n(&n);
  printf("\nВведене значення %d",n);
  printf("\n\n");
  system("pause");
  return 0;
}
/* Ввід значення n */
void vvid_n(int *n) {
  printf("Введіть значення n в межах від 0 до 9:   ");
  scanf("%d",n);
  if (*n<0 || *n>9) vvid_n(n);
}
```

```
Введіть значення n в межах від 0 до 9:   15
Введіть значення n в межах від 0 до 9:   -2
Введіть значення n в межах від 0 до 9:   3

Введене значення 3
```

У функцію vvid_n параметр передається за адресою.

Завдання. Написати програму обчислення значення факторіала

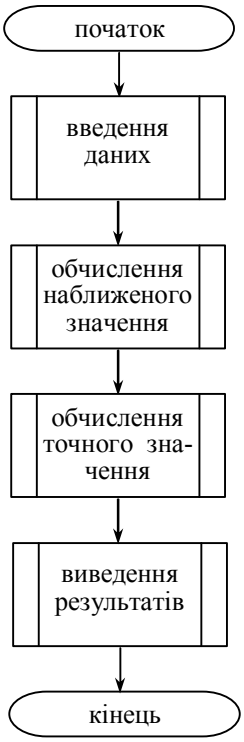
$$\begin{cases} n! = n \cdot (n-1) \cdot \ldots \cdot 3 \cdot 2 \cdot 1, \\ 0! = 1, \end{cases}$$

використовуючи рекурсивну функцію.

Практичні заняття № 15-16 (*комп'ютерний практикум № 10-11*)
Розробка програм модульної структури з використанням функцій. Налагодження програм

Мета занять: набути практичних навичок розробляти програми модульної структури і програмувати їх засобами мови С з використанням функцій; виконувати налагодження програм модульної структури.

Завдання. На основі розробленого в комп'ютерному практикумі № 9 алгоритму обчислення значення функції $f(x)$ розробити і написати дві програми модульної структури, використовуючи функції введення даних, обчислення наближеного значення заданої функції, обчислення точного значення заданої функції, виведення результатів. Схема головної функції має бути такою:



При цьому в першій програмі використати функції без параметрів — обмін інформацією здійснювати через глобальні змінні; у другій програмі обмін інформацією здійснювати через параметри без використання глобальних змінних.

У кожній функції після її заголовка в коментарях вказати для всіх змінних і параметрів, які в ній використовуються, змінні локальні чи глобальні,

параметри передаються за значенням чи за адресою. Коментарі в програмах обов'язкові (17-25%). Провести розрахунки не менш ніж з трьома різними наборами вхідних даних.

Варіанти завдань

Зауваження. Хоч ряди, для яких не вказано проміжок збіжності, збігаються на всій числовій прямій, при розрахунках брати проміжок $[-10,10]$.

1. $\dfrac{1-\sqrt{1-4x}}{2}=\sum\limits_{n=0}^{\infty}\dfrac{(2n)!x^{n+1}}{n!(n+1)!}$ при $x\in(-0{,}25;0{,}25)$

2. $1+(x-1)e^x=\sum\limits_{n=0}^{\infty}\dfrac{(n+1)x^{n+2}}{(n+2)!}$

3. $\cos x-1=\sum\limits_{n=1}^{\infty}\dfrac{(-1)^n x^{2n}}{(2n)!}$

4. $\sin x=\sum\limits_{n=0}^{\infty}\dfrac{(-1)^n x^{2n+1}}{(2n+1)!}$

5. $\dfrac{e^x+e^{-x}+2\cos x}{4}=\sum\limits_{n=0}^{\infty}\dfrac{x^{4n}}{(4n)!}$

6. $chx-1=\dfrac{e^x+e^{-x}}{2}-1=\sum\limits_{n=1}^{\infty}\dfrac{x^{2n}}{(2n)!}$

7. $chx=\dfrac{e^x+e^{-x}}{2}=\sum\limits_{n=0}^{\infty}\dfrac{x^{2n}}{(2n)!}$

8. $\cos x=\sum\limits_{n=0}^{\infty}\dfrac{(-1)^n x^{2n}}{(2n)!}$

9. $\dfrac{1}{2}-x-\dfrac{\sqrt{1-4x}}{2}=\sum\limits_{n=1}^{\infty}\dfrac{(2n)!x^{n+1}}{(n!)^2(n+1)}$ при $x\in[-0{,}25;0{,}25)$

10. $x\cos x=\sum\limits_{n=0}^{\infty}\dfrac{(-1)^n x^{2n+1}}{(2n)!}$

11. $(x-1)e^x+1=\sum\limits_{n=0}^{\infty}\dfrac{x^{n+2}}{n!(n+2)}$

12. $shx-x=\dfrac{e^x-e^{-x}}{2}-x=\sum\limits_{n=1}^{\infty}\dfrac{x^{2n+1}}{(2n+1)!}$

13. $\dfrac{1}{(1-4x)\sqrt{1-4x}}-1-6x=\sum\limits_{n=2}^{\infty}\dfrac{(2n+1)!x^n}{(n!)^2}$ при $x\in[-0{,}25;0{,}25)$

14. $1+(x-1)e^x-\dfrac{x^2}{2}=\sum\limits_{n=1}^{\infty}\dfrac{(n+1)x^{n+2}}{(n+2)!}$

15. $\dfrac{1}{2}-\dfrac{\sqrt{1-4x}}{2}=\sum\limits_{n=0}^{\infty}\dfrac{(2n)!x^{n+1}}{(n!)^2(n+1)}$ при $x\in[-0{,}25;0{,}25)$

16. $\sin x-x=\sum\limits_{n=1}^{\infty}\dfrac{(-1)^n x^{2n+1}}{(2n+1)!}$

17. $\dfrac{e^x-e^{-x}-2\sin x}{4}=\sum\limits_{n=0}^{\infty}\dfrac{x^{4n+3}}{(4n+3)!}$

18. $\dfrac{1-2x-\sqrt{1-4x}}{4}=\sum\limits_{n=0}^{\infty}\dfrac{(2n+1)!x^{n+2}}{n!(n+2)!}$ при $x\in[-0{,}25;0{,}25]$

19. $shx = \dfrac{e^x - e^{-x}}{2} = \sum\limits_{n=0}^{\infty} \dfrac{x^{2n+1}}{(2n+1)!}$ 20. $x\cos 3x = \sum\limits_{n=0}^{\infty} \dfrac{(-1)^n 3^{2n} x^{2n+1}}{(2n)!}$

21. $\dfrac{1}{(1-4x)\sqrt{1-4x}} = \sum\limits_{n=0}^{\infty} \dfrac{(2n+1)!x^n}{(n!)^2}$ при $x \in [-0{,}25; 0{,}25)$

22. $\dfrac{e^x - e^{-x} + 2\sin x}{4} = \sum\limits_{n=0}^{\infty} \dfrac{x^{4n+1}}{(4n+1)!}$

23. $\dfrac{(1-4x)\sqrt{1-4x} + 6x(1-x) - 1}{12} = \sum\limits_{n=0}^{\infty} \dfrac{(2n+2)!x^{n+3}}{(n+1)!(n+3)!}$ при $x \in [-0{,}25; 0{,}25]$

24. $\ln 2 - \ln(1+\sqrt{1-4x}) = \sum\limits_{n=0}^{\infty} \dfrac{(2n+1)!x^{n+1}}{\left((n+1)!\right)^2}$ при $x \in [-0{,}25; 0{,}25]$

25. $x - \sin x = \sum\limits_{n=2}^{\infty} \dfrac{(-1)^n x^{2n-1}}{(2n-1)!}$ 26. $\dfrac{1}{\sqrt{1-4x}} = \sum\limits_{n=0}^{\infty} \dfrac{(2n)!x^n}{(n!)^2}$ при $x \in [-0{,}25; 0{,}25)$

26. $\dfrac{(1-4x)\sqrt{1-4x} + 6x - 1}{12} = \sum\limits_{n=0}^{\infty} \dfrac{(2n)!x^{n+2}}{n!(n+2)!}$ при $x \in [-0{,}25; 0{,}25]$

28. $\ln \dfrac{2}{1+\sqrt{1-4x}} = \sum\limits_{n=1}^{\infty} \dfrac{(2n-1)!x^n}{\left(n!\right)^2}$ при $x \in [-0{,}25; 0{,}25]$

29. $a^x = \sum\limits_{n=0}^{\infty} \dfrac{(\ln a)^n x^n}{n!}$ $(a>0,\ a \neq 1)$ 30. $e^x - 1 = \sum\limits_{n=0}^{\infty} \dfrac{x^{n+1}}{(n+1)!}$

Налагодження програми. Якщо одержаний результат виконання програми відрізняється від очікуваного, то в логіці програми є помилки. Для виявлення і виправлення цих помилок можна перейти в режим налагодження, в якому для перевірки стану локальних змінних використовуються точки призупинення виконання програми і стає доступною інформація про поточні значення змінних.

Для встановлення точки призупинення виконання програми треба клацнути мишкою на лівому полі біля границі вікна або встановити курсор на рядку програми і натиснуть F9 — з'явиться червоний кружечок-позначка. Повторне виконання дії знімає позначку.

Для виконання програми в режимі налагодження треба натиснути клавішу F5, або вибрати команду Debug – Start Debugging, або на стандартній панелі інструментів натиснути кнопку Start Debugging.

Коли програма досягне точки призупинення, виконання програми тимчасово припиняється. Жовта стрілка на лівому полі вказує на рядок, який буде виконано наступним. Значення змінних в даний момент виконання програми можна побачити у спливаючих підказках, які з'являються при підведенні курсора на змінну, або у вікні Watch. Щоб додати змінну у вікно Watch, треба клацнути правою кнопкою мишки по імені змінної і вибрати команду Add Watch — змінна додасться у вікно Watch (якщо це вікно перекриває текст програми, то його, взявши за рядок заголовку, можна перетягнути вниз екрана).

Значення змінних у вікні Watch при виконанні програми змінюються.

Для виконання чергового рядка коду без заходу всередину функції треба натиснути F10 або вибрати команду Debug – Step Over.

Для виходу з поточної функції треба вибрати команду Debug – Step Out.

Щоб виконати частину програми до певного місця, треба встановити курсор у цьому місці і вибрати команду Debug – Run to cursor.

Для дострокового завершення роботи в режимі налагодження треба натиснути комбінацію клавіш Shift + F5 або вибрати команду Debug – Stop Debugging.

Практичне заняття № 17
Обробка одновимірних масивів.
Алгоритми сортування в одновимірних масивах

Мета заняття: закріпити лекційний матеріал і виробити навички застосування типових алгоритмів при обробці одновимірних масивів; розглянути прості алгоритми сортування масивів.

1. Робота з одновимірними масивами. Масив — це структура даних, яка складається з елементів того самого типу. Одновимірні масиви можна інтерпретувати як вектори. Доступ до окремого елемента масиву здійснюється за допомогою імені масиву й індекса. Ім'я масиву — це вказівник на діля-

нку пам’яті, в якій розміщується перший елемент масиву. Індексом елемента може бути будь-який вираз цілого типу int, типу char чи типу enum. У всіх масивів індексація елементів починається з індекса 0. Введення й виведення масивів здійснюється поелементно. У мові С перевірка виходу за границі масиву покладається на програміста.

Якщо параметром функції є одновимірний масив, то її формальний параметр можна описати трьома способами:

1) як вказівник — `void func(int *mas);`

2) як масив фіксованого розміру — `void func(int mas[5]);`

3) як масив невизначеного розміру — `void func(int mas[]);`

Звернення до функції у всіх трьох випадках буде однаковим: `func(mas)`.

Приклад 1.1. Знайти кут між двома векторами. Використати формулу $\cos\alpha = \frac{(\vec{a},\vec{b})}{|\vec{a}|\cdot|\vec{b}|}$; перехід від радіанів до градусів здійснити за формулою $\varphi_{\circ} = \varphi_{r}\cdot 180/\pi$.

```
#include <stdio.h>
#include <stdlib.h>
#define _USE_MATH_DEFINES
#include <math.h>
#define N 10   // максимальна розмірність вектора
void vvid(int *a, int n);
int dobutok(int *a, int *b, int n);
int main() {
  int a[N],b[N];
  int n;   // реальна довжина вектора
  double alfa;
  system("chcp 1251 & cls");
  printf("Вкажіть розмірність векторів ");
  scanf("%d",&n);
  printf("Введіть координати вектора a "); vvid(a,n);
  printf("Введіть координати вектора b "); vvid(b,n);
```

```
  /* обчислення кута */
  alfa=acos(dobutok(a,b,n)/
       sqrt((double)dobutok(a,a,n)*dobutok(b,b,n)))/M_PI*180.;
  printf("Кут між векторами %0.2f°",alfa);
  printf("\n\n");
  system("pause");
  return 0;
}
/* Ввід координат вектора */
void vvid(int *a, int n) {
  int i;
  for (i=0; i<n; i++) scanf("%d",&a[i]);
}
/* Знаходження скалярного добутку двох векторів */
int dobutok(int *a, int *b, int n) {
  int i;
  int s=0;
  for (i=0; i<n; i++)
    s+=a[i]*b[i];
  return s;
}
```

```
Вкажіть розмірність векторів 5
Введіть координати вектора a 1 2 3 4 5
Введіть координати вектора b 5 4 3 2 1
Кут між векторами 50.48°
```

```
Вкажіть розмірність векторів 3
Введіть координати вектора a 0 1 0
Введіть координати вектора b 0 0 1
Кут між векторами 90.00°
```

Завдання. 1. Визначити, чи є вектори порівнюваними (тобто, $\vec{a} \le \vec{b}$, чи $\vec{a} = \vec{b}$, чи $\vec{a} \ge \vec{b}$), чи непорівнюваними.

2. Написати програму для обчислення факторіала великого числа. Для зберігання цифр результату скористатися елементами цілочисельного масиву. Максимальне значення числа типу int дорівнює 2 147 483 647. Для економії місця в пам'яті вигідніше розміщувати в одному елементі масиву якомога більше цифр. При цьому треба стежити, щоб результати всіх проміжних операцій не перевищували максимального значення для вибраного типу даних (щоб не було переповнення пам'яті). Нехай є, наприклад, 14-значне число і подамо його в системі числення за основою d=1000000:

$$12345678901234 = 12 \cdot (10^6)^2 + 345678 \cdot (10^6)^1 + 901234 \cdot (10^6)^0.$$

Для його зберігання треба всього 3 елементи масиву A[2], A[1] і A[0].

Розглянемо множення числа

$$A = a_n d^n + a_{n-1} d^{n-1} + ... + a_2 d^2 + a_1 d^1 + a_0$$

на число k, у результаті чого одержимо число

$$A \cdot k = \alpha_m d^m + \alpha_{m-1} d^{m-1} + ... + \alpha_2 d^2 + \alpha_1 d^1 + \alpha_0$$

При цьому

$\alpha_0 = (a_0 \cdot k)\%d$ і в наступний розряд додається перенос $r_1 = (a_0 \cdot k)/d$;

$\alpha_1 = (a_1 \cdot k + r_1)\%d$ і перенос $r_2 = (a_1 \cdot k + r_1)/d$

$\alpha_2 = (a_2 \cdot k + r_2)\%d$ і перенос $r_3 = (a_2 \cdot k + r_2)/d$ і т.д.

Обчислення закінчуються, коли одночасно виконаються 2 умови:

— усі цифри числа *A* опрацьовано;

— перенос у наступний розряд дорівнює 0.

2. Пошук елементів у масиві. Задача пошуку елементів масиву з певними властивостями чи пошуку елементів, рівних заданому, на практиці виникає досить часто.

Приклад 2.1. Вивести на екран перший мінімальний і перший максимальний елементи масиву. Цикл пошуку виконати з другого елемента, щоб елемент не порівнювати сам із собою.

```
#include <stdio.h>
#include <stdlib.h>
#define N 10
int main() { // пошук мінімального й максимального елементів
  int a[N]={2,4,1,8,9,1,3,9,1,9};
  int kmin, kmax;   // номери мінімального й максимального елементів
  int i;   // номер поточного елемнента
  int amin, amax;   // мінімальний і максимального елементи
  system("chcp 1251 && cls");
  amin=a[0]; kmin=0; amax=a[0]; kmax=0;
  for (i=1; i<N; i++) { // цикл виконується з другого елемента
    if (a[i]<amin) { amin=a[i]; kmin=i; }
```

```
    if (a[i]>amax) { amax=a[i]; kmax=i; }
  }
  printf("Мінімальний елемент  a[%d]=%d\n",kmin,amin);
  printf("Максимальний елемент a[%d]=%d",kmax,amax);
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Мінімальний елемент  a[2]=1
Максимальний елемент a[4]=9
```

Приклад 2.2. Алгоритм бінарного (дихотомічного) пошуку — ділення масиву навпіл. Цей алгоритм застосовують до відсортованого за зростанням масиву. При кожному пошукові інтервал між початковим і кінцевим індексами ділиться на дві рівні частини Максимальна кількість повторень циклу дорівнює $\log_2 n+1$. При цьому, якщо індекс елемента *k*, то його номер *k*+1.

```
#include <stdio.h>
#include <stdlib.h>
#define N 9
/* використання бінарного пошуку для визначення
   в масиві номера елемента, рівного x */
int main() {
  int a[N]={1,5,7,9,12,18,21,40,45};
  int x;   // задане значення
  int i,j,k;   // індекс елемнта
  system("chcp 1251 & cls");
  for (i=0; i<N; i++) printf("%4d",a[i]);
  printf("введіть значення x\n");
  scanf("%d",&x);
  i=0;   // почаковий індекс
  j=N-1;   // кінцевий індекс
  do {
    k=(i+j)/2;   // середина інтервалу індексів
    if (x>a[k]) i=k+1;
    else j=k-1;
  } while (a[k]!=x && i<=j);
  if(a[k]==x) printf("Номер елемента %d",k+1);
  else printf("Елемент не знайдено");
```

```
  printf("\n\n");
  system("pause");
  return 0;
}
```

Наведемо приклад покрокового виконання цього алгоритму.

1. Нехай масив має 9 елементів:

1 5 7 9 12 18 21 40 45

і x=7 (у масиві є елемент з таким значенням).

Початкові значення: i=0
j=8
І прохід циклу: k=(0+8)/2=4
(x=7)<(a[4]=12),
отже, j=4-1=3, і залишається=0
оскільки (a[4]=12)≠(x=7) і (i=0)<=(j=3), то виконується
ІІ прохід циклу: k=(0+3)/2=1
(x=7)>(a[1]=5),
отже, i=1+1=2, j залишається=3
оскільки (a[1]=5)≠(x=7) і (i=2)<=(j=3), то виконується
ІІІ прохід циклу: k=(2+3)/2=2
(x=7)!>(a[2]=7),
отже, j=2-1=1, і залишається=2
оскільки (a[2]=7)=(x=7), то цикл закінчує роботу.

Оскільки при k=2 (a[2]=7)=(x=7),
то буде виведено номер елемента 2+1=3.

```
   1   5   7   9  12  18  21  40  45
введіть значення x
7
Номер елемента 3
```

2. Нехай масив має ті самі 9 елементів і x=4 (у масиві немає елемента з таким значенням).

Початкові значення: i=0
j=8
І прохід циклу: k=(0+8)/2=4
(x=4)<(a[4]=12),
отже, j=4-1=3, і залишається=0
оскільки (a[4]=12)≠(x=4) і (i=0)<=(j=3), то виконується
ІІ прохід циклу: k=(0+3)/2=1
(x=4)<(a[1]=5),
отже, j=k-1=0, і залишається=0
оскільки (a[1]=5)≠(x=4) і (i=0)<=(j=0), то виконується
ІІІ прохід циклу: k=(0+0)/2=0
(x=4)>(a[0]=1),
отже, j=0-1=-1, і залишається=0
оскільки (a[2]=7)!=(x=4) і (i=0)>(j=-1), то цикл закінчує роботу.

Оскільки при k=0 (a[0]=1)≠(x=4),
то буде виведено повідомлення,
що елемент не знайдено.

```
   1   5   7   9  12  18  21  40  45
введіть значення x
4
Елемент не знайдено
```

Приклад 2.3. Алгоритм бінарного пошуку з використанням рекурсивної функції.

```
#include <stdio.h>
#include <stdlib.h>
#define N 10
int binary(int *a,int x,int i,int j);
/* використання бінарного рекурстивного пошуку
   для визначення в масиві номера елемента, рівного x */
int main() {
  int a[N]={1,5,7,9,12,18,21,40,45,100};
  int x;   // задане значення
  int k;   // індекс знайденого елемента
  int i;
  system("chcp 1251 & cls");
  for (i=0; i<N; i++) printf("%4d",a[i]);
  printf("\nВведіть значення x\n");
  scanf("%d",&x);
  k=binary(a,x,0,N-1);
  if (k==-1)  printf("Елемент не знайдено");
  else printf("Номер елемента %d",k+1);
  printf("\n\n");
  system("pause");
  return 0;
}
int binary(int *a,int x,int i,int j) {
  int k;
  if (i>j) k=-1;
  else {
    k=(i+j)/2;   // середина інтервалу індексів
    if (x<a[k]) k=binary(a,x,i,k-1);
    else
      if (x>a[k]) k=binary(a,x,k+1,j);
  }
  return k;
}
```

```
   1   5   7   9  12  18  21  40  45 100
Введіть значення x
5
Номер елемента 2
```

```
   1   5   7   9  12  18  21  40  45 100
Введіть значення x
41
Елемент не знайдено
```

Завдання. 1. Знайти номери всіх елементів масиву, рівних заданому числу *x*. Для збереження знайдених індексів елементів використати новий масив.

2. Порахувати кількість різних елементів масиву.

3. Написати програму, яка реалізує алгоритм — решето Ератосфена (бл. 275-194 до н. е.; давньогрецький науковець і письменник) пошуку простих чисел. Прості числа діляться лише самі на себе і на 1 (число 1 не є простим). Першими простими числами є: 2, 3, 5, 7, 11, 13, 17, 19, 23, 49... Алгоритм Ератосфена: беремо перше просте число — 2 і всі числа, кратні 2, вилучаємо — в масиві замінюємо їх на 0. Далі в масиві шукаємо наступне число, не рівне 0; це буде просте число; всі наступні числа, кратні йому, вилучаємо і т.д.

3. Перестановки в масивах. Перестановки в масивах використовують у випадках, коли треба зібрати елементи з певними властивостями на початку чи в кінці масиву, при сортуванні масивів.

Прикладт 3.1. Усі елементи, кратні 5, записати на початок масиву, зберігши їхній порядок, а всі решта перемістити в кінець, теж зберігши їхній порядок.

```
#include <stdio.h>
#include <stdlib.h>
#define N 10
int main() {
  int a[N]={1,9,100,21,40,5,12,18,7,45};
  int r;   // робоча змінна
  int i;   // індекс поточного елемента
  int k;   // індекс елемента, кратного 5
  int j;   // індекс елемента, який підсувається
  system("chcp 1251 & cls");
  printf("Початковий масив\n");
  for (i=0; i<N; i++) printf("%d  ",a[i]);
  /* Перестановка */
  k=-1;
  for (i=0; i<N; i++)
    if (a[i]%5==0) {
```

```
      k++; r=a[i];
      for (j=i-1; j>=k; j--) a[j+1]=a[j];
      a[k]=r;
    }
  printf("\nОброблений масив\n");
  for (i=0; i<N; i++) printf("%d  ",a[i]);
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Початковий масив
1  9  100  21  40  5  12  18  7  45
Оброблений масив
100  40  5  45  1  9  21  12  18  7
```

Приклад 3.2. Написати програму сортування елементів масиву за зростанням, використавши такий алгоритм. Перший елемент порівнюється з усіма наступними. Якщо наступний поточний елемент менший від першого, то елементи переставляються місцями. Потім береться поточний наступний елемент і знову, якщо треба, то переставляються місцями з першим і т.д. Після перегляду масиву до кінця на першому місці буде стояти мінімальний елемент. Далі другий елемент порівнюється з усіма наступними і т. д.

```
#include <stdio.h>
#include <stdlib.h>
#define N 10
/* сортування: перестановка поточного і меншого наступного
   елементів */
int main() {
  int a[N]={100,21,40,5,1,9,12,18,7,45};
  int r;   // робоча змінна
  int i;   // індекс поточного елемента
  int j;   // індекс наступного елемента
  system("chcp 1251 & cls");
  printf("\nПочатковий масив\n");
  for (i=0; i<N; i++)
    printf("  %d",a[i]);
 /* Сортування */
  for (i=0; i<N-1; i++)
    for (j=i+1; j<N; j++)
```

```
      if (a[j]<a[i]) {
        r=a[i];
        a[i]=a[j];
        a[j]=r;
      }
  printf("\nВідсортований масив\n");
  for (i=0; i<N; i++)
    printf("  %d",a[i]);
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Початковий масив
  100  21  40  5  1  9  12  18  7  45
Відсортований масив
  1  5  7  9  12  18  21  40  45  100
```

Приклад 3.3. Сортування масивів методом бульбашки (див. Додаток Г) і злиття двох відсортованих масивів. Алгоритм злиття двох відсортованих масивів починається з порівняння перших елементів відповідних масивів a і b. За лічильником i вибираються елементи з масиву a, за лічильником j — з масиву b, а за лічильником k —елементи заносяться до злитого масиву ab. У масив ab заноситься менший з елементів, що порівнюються, а далі в порівнянні бере участь наступний елемент того масиву, елемент якого вже записаний до масиву ab. Ця процедура повторюється, доки елементи одного з масивів не закінчаться. Наприкінці треба переписати до масиву ab всі елементи іншого масиву, які залишилися.

```
#include <stdio.h>
#include <stdlib.h>
#define N 5
#define M 4
void vvid(int *a, int n);
void vyvid(int *a, int n);
void sortbul(int *a, int n);
void zlyttia(int *a, int n, int *b, int m, int *ab);
/* Сортування масивів методом звичайної бульбашки
   і злиття двох відсортованих масивів */
```

```
int main() {
  int a[N], b[M];   // вхідні масиви
  int ab[N+M];   // масив, утворений злиттям
  // заповнення масивів
  system("chcp 1251 & cls");
  printf("\Введіть елементи масиву a[%d]:\n",N);
  vvid(a,N);
  printf("\Введіть елементи масиву b[%d]:\n",M);
  vvid(b,M);
  printf("\Перший масив:\n");
  vyvid(a,N);
  printf("\nДругий масив:\n");
  vyvid(b,M);
  sortbul(a,N); sortbul(b,M);   // сортування масивів
  printf("\nВідсортований перший масив:\n");
  vyvid(a,N);
  printf("\nВідсортований другий масив:\n");
  vyvid(b,M);
  zlyttia(a,N,b,M,ab);   // злиття відсортованих масивів
  printf("\nРезультуючий злитий масив:\n");
  vyvid(ab,N+M);   // вивід результату злиття
  printf("\n\n");
  system("pause");
  return 0;
}
/* Ввід елементів масиву */
void vvid(int *a, int n) {
  int i;
  for (i=0; i<n; i++)
    scanf("%d",&a[i]);
}
/* Вивід елементів масиву */
void vyvid(int *a, int n) {
  int i;
  for (i=0; i<n; i++) printf("  %d",a[i]);
}
```

```
/* Сортування масиву методом звичайної бульбашки */
void sortbul(int *a, int n) {
  int i, j;
  int r;
  for (i=0; i<n-1; i++)
    for (j=0; j<n-i-1; j++)
      if (a[j]>a[j+1]) {r=a[j]; a[j]=a[j+1]; a[j+1]=r;}
}
/* Злиття відсортованих масивів */
void zlyttia(int *a, int n, int *b, int m, int *ab) {
  int i, j, k;   // індекси масивів a, b, ab
  int l;   // індекс для дозапису
  /* злиття, доки не буде досягнуто кінця одного з масивів */
  for (i=0, j=0, k=0; i<n && j<m; k++)
    if (a[i]<b[j])
      ab[k]=a[i++];
    else
      ab[k]=b[j++];
  /* дозапис елементів з іншого масиву */
  if (i==n)
    for (l=k; l<n+m; l++)
      ab[l]=b[j++];
  else
    for (l=k; l<n+m; l++)
      ab[l]=a[i++];
}
```

```
Введіть елементи масиву a[5]:
5 0 3 8 4
Введіть елементи масиву b[4]:
6 2 9 1
Перший масив:
  5  0  3  8  4
Другий масив:
  6  2  9  1
Відсортований перший масив:
  0  3  4  5  8
Відсортований другий масив:
  1  2  6  9
Результуючий злитий масив:
  0  1  2  3  4  5  6  8  9
```

Завдання. Написати програми обробки одновимірних масивів.

1. Виконати циклічний зсув елементів масиву, щоб а) останній елемент став першим, наприклад, 1 2 3 4 5 → 5 1 2 3 4; б) перший елемент став останнім, наприклад, 1 2 3 4 5 → 2 3 4 5 1.

2. Зібрати на початку масиву всі елементи, відмінні від нуля, зберігши їхній порядок, а всі нулі помістити в кінці масиву.

3. Поділити цілочисельний масив на два — з парними й непарними значеннями.

Практичне заняття № 18 (*комп'ютерний практикум № 12*)

Обробка одновимірних масивів

Мета заняття: набути практичних навичок розробляти алгоритми обробки одновимірних масивів, а також створювати програми модульної структури, в яких використовуються ці алгоритми, і програмувати їх засобами мови С з використанням функцій.

Завдання. Розробити алгоритм обробки цілочисельного масиву відповідно до завдання варіанту (всю обробку виконувати в тому самому масиві — робочі масиви не використовувати), запрограмувати його у вигляді однієї чи двох функцій.

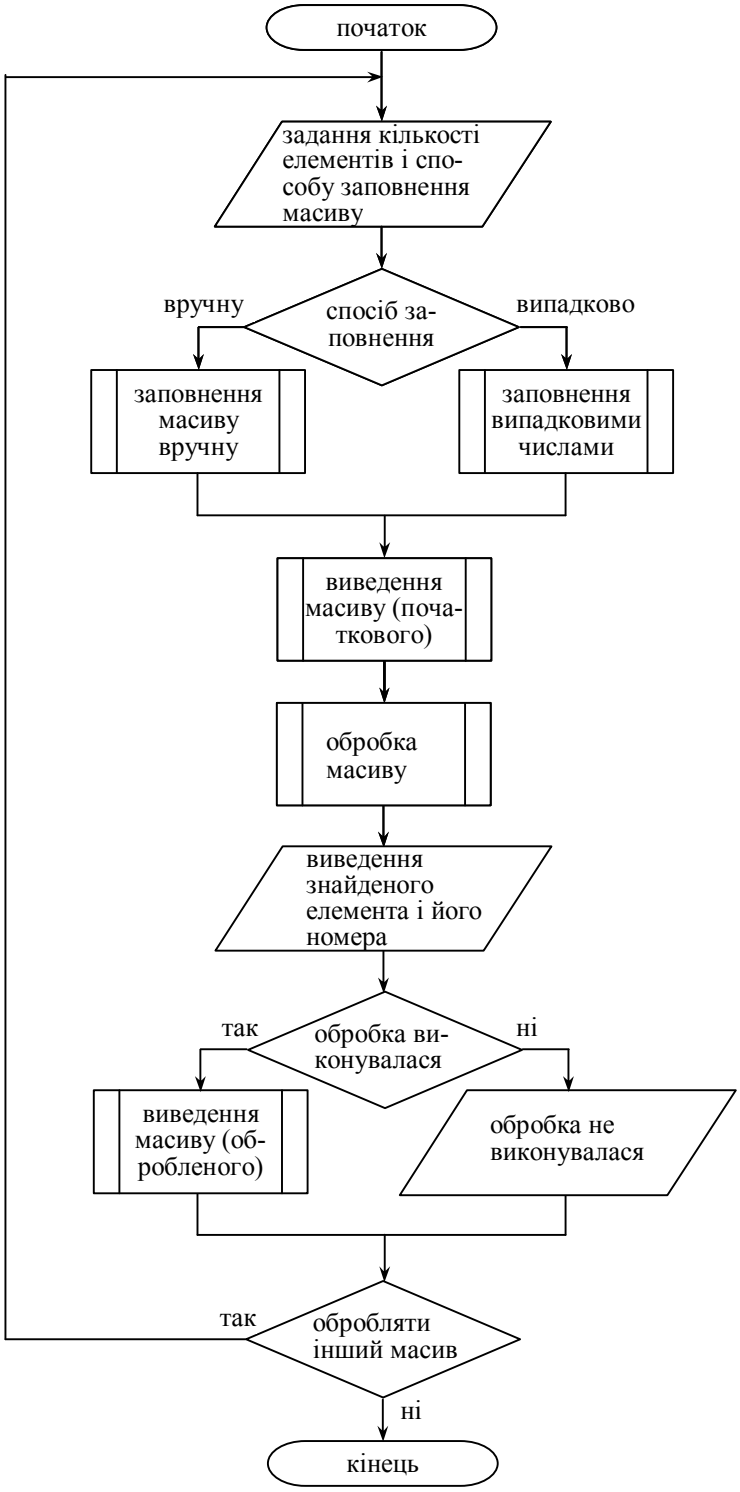
Розробити функцію введення в діалоговому режимі елементів масиву, функцію заповнення масиву випадковими числами, функцію виведення масиву (функцію виведення використовувати для виведення початкового та обробленого масивів).

У функціях глобальні змінні не використовувати. У функціях, призначених для обробки інформації, не здійснювати введення і виведення інформації.

У головній функції в діалоговому режимі задати реальну кількість елементів масиву (максимальну кількість елементів масиву визначити директивою препроцесору #define); передбачити вибір у діалоговому режимі способу заповнення масиву (вручну чи випадковими числами) і звернення до відповідної функції; звернення до функції виведення масиву для виведення вхідного масиву; звернення до функції обробки масиву; виведення знайденого мінімального чи максимального елемента і його номера; звернення до функції виведення масиву (тої самої, що й для виведення вхідного масиву) для виведення масиву після обробки; передбачити можливість багаторазового виконання алгоритму.

Коментарі в програмі обов'язкові (17-25%).

Блок-схему головної функції подано нижче (при написанні програми дотримуватися поданої схеми):



За алгоритмом провести розрахунки не менш ніж з трьома різними наборами вхідних даних. При цьому одним із масивів має бути 5-елементний масив, для якого подати протокол покрокового знаходження мінімального / максимального елемента.

Зауваження. При виконанні завдання треба врахувати, що індексація елементів масиву починається з 0, і, якщо індекс елемента дорівнює *n*, то номер цього елемнента дорівнює *n*+1. Перша і друга половини масиву у випадку непарної й парної кількості елементів визначаються так, як це показано на рисунку:

Незалежно від того, кількість елементів масиву є парним чи непарним числом, алгоритм обробки даних має бути тим самим при визначенні першої чи другої половини масиву, а також при організації циклів (треба використати ту властивість, що результатом ділення двох цілих чисел є ціле число — дробова частина відкидається).

Варіанти завдань

1. Якщо мінімальний елемент першої половини масиву має парний номер і дорівнює 0, то виконати таку обробку масиву: переставити місцями кожні два сусідні елементи (a_1 і a_2, a_3 і a_4 …), якщо вони відрізняються рівно в 2 рази.

2. Якщо максимальний елемент масиву має парний номер і кратний 3, то виконати таку обробку масиву: елементи другої половини масиву розмістити у зворотному порядку.

3. Якщо мінімальний елемент другої половини масиву має парний номер і є від'ємним числом, то виконати таку обробку масиву: елементи з парними значеннями зібрати на початок масиву, зберігши їхній порядок; решту елементів розмістити в кінці масиву, замінивши нулями.

4. Якщо максимальний елемент масиву має непарний номер і розміщується в другій половині масиву, то виконати таку обробку: елементи з парними значеннями зібрати в кінці масиву, зберігши їхній порядок; решту елементів розмістити на початку масиву, замінивши нулями.

5. Якщо мінімальний елемент масиву кратний 10 і розміщується в першій половині масиву, то виконати таку обробку масиву: переставити місцями кожні два сусідні елементи (a_1 і a_2, a_3 і a_4 …), якщо вони мають різні знаки.

6. Якщо максимальний елемент масиву кратний 5 і розміщується в другій половині масиву, то виконати таку обробку масиву: елементи, значення яких відмінні від 0, зібрати на початок масиву, зберігши їхній порядок, а нулі розмістити в кінці масиву.

7. Якщо мінімальний елемент масиву має парний номер і є парним числом, то виконати таку обробку масиву: елементи першої його половини розмістити в зворотному порядку.

8. Якщо максимальний елемент першої половини масиву має непарний номер і дорівнює 0, то виконати таку обробку масиву: елементи з парними номерами (a_2, a_4, a_6 ...) розмістити в зворотному порядку (... a_6, a_4, a_2), а елементи з непарними номерами залишити на попередньому місці.

9. Якщо мінімальний елемент другої половини масиву має парний номер і дорівнює 0, то виконати таку обробку масиву: впорядкувати за зростанням кожні два сусідні елементи (a_1 і a_2, a_3 і a_4 …).

10. Якщо максимальний елемент першої половини масиву кратний 10 і має парний номер, то виконати таку обробку масиву: елементи, значення яких кратні 3, зібрати в кінці масиву, зберігши їхній порядок; решту елементів розмістити на початку масиву, замінивши нулями.

11. Якщо мінімальний елемент масиву кратний 5 і розміщується в першій половині масиву, то виконати таку обробку: переставити місцями кожні два сусідні елементи (a_1 і a_2, a_3 і a_4 …), якщо вони мають різні знаки.

12. Якщо максимальний елемент другої половини масиву має непарний номер і є непарним числом, то виконати таку обробку масиву: елементи, значення яких відмінні від 0, зібрати на початку масиву, зберігши їхній порядок, а нулі розмістити в кінці масиву.

13. Якщо мінімальний елемент масиву має парний номер і розміщується в другій половині масиву, то виконати таку обробку масиву: елементи з непарни-

ми номерами зібрати у кінці масиву, зберігши їхній порядок; решту елементів розмістити на початку масиву, замінивши нулями.

14. Якщо максимальний елемент другої половини масиву має непарний номер і є додатним числом, то виконати таку обробку масиву: переставити місцями кожні два сусідні елементи (a_1 і a_2, a_3 і a_4 …), якщо їхні абсолютні значення відрізняються менш, ніж у 2 рази.

15. Якщо мінімальний елемент другої половини масиву має парний номер і є парним числом, то виконати таку обробку масиву: елементи з непарними номерами (a_1, a_3, a_5 ...) розмістити в зворотному порядку (... a_5, a_3, a_1), а елементи з парними номерами залишити на попередньому місці.

16. Якщо максимальний елемент масиву кратний 3 і розміщується в першій половині масиву, то виконати таку обробку масиву: елементи, значення яких відмінні від 0, зібрати на початок масиву, зберігши їхній порядок, а нулі розмістити в кінці масиву.

17. Якщо мінімальний елемент першої половини масиву має непарний номер і кратний 3, то виконати таку обробку масиву: переставити місцями кожні два сусідні елементи (a_1 і a_2, a_3 і a_4 …), якщо жоден із пари не дорівнює нулю.

18. Якщо максимальний елемент другої половини масиву кратний 4 і має парний номер, то виконати таку обробку масиву: елементи, значення яких кратні 3, зібрати на початок масиву, зберігши їхній порядок; решту елементів розмістити в кінці масиву, замінивши нулями.

19. Якщо мінімальний елемент другої половини масиву має парний номер і є від'ємним числом, то виконати таку обробку масиву: елементи з непарними значеннями зібрати в кінці масиву, зберігши їхній порядок; решту елементів розмістити на початку масиву, замінивши нулями.

20. Якщо максимальний елемент масиву має непарний номер і розміщується в першій половині масиву, то елементи першої його половини розмістити в зворотному порядку.

21. Якщо мінімальний елемент першої половини масиву має парний номер і є непарним числом, то виконати таку обробку масиву: переставити місцями кожні два сусідні елементи (a_1 і a_2, a_3 і a_4 …), якщо їхні абсолютні значення відрізняються більш, ніж у 2 рази.

22. Якщо максимальний елемент другої половини масиву має непарний номер і є парним числом, то переставити місцями першу і другу його половини, зберігши в них порядок елементів.

23. Якщо мінімальний елемент першої половини масиву має парний номер і є додатним числом, то елементи з парними номерами (a_2, a_4, a_6 ...) розмістити в зворотному порядку (... a_6, a_4, a_2), а елементи з непарними номерами залишити на попередньому місці.

24. Якщо максимальний елемент масиву має непарний номер і розміщується в другій половині масиву, то виконати таку обробку масиву: елементи другої його половини розмістити в зворотному порядку.

25. Якщо мінімальний елемент першої половини масиву кратний 7 і має непарний номер, то виконати таку обробку масиву: впорядкувати за зростанням кожні два сусідні елементи (a_1 і a_2, a_3 і a_4 …), якщо вони мають різну парність.

26. Якщо максимальний елемент масиву має непарний номер і розміщується в другій половині масиву, то елементи першої його половини розмістити в зворотному порядку.

27. Якщо мінімальний елемент першої половини масиву має парний номер і є додатним числом, то виконати таку обробку масиву: переставити місцями першу і другу його половини, зберігши в них порядок елементів.

28. Якщо максимальний елемент другої половини масиву має непарний номер і є від'ємним числом, то виконати таку обробку масиву: впорядкувати за спаданням кожні два сусідні елементи (a_1 і a_2, a_3 і a_4 …), якщо вони мають однакову парність.

29. Якщо мінімальний елемент другої половини масиву має парний номер і є додатним числом, то виконати таку обробку масиву: елементи з пар-

ними номерами зібрати на початку масиву, зберігши їхній порядок; решту елементів розмістити у кінці масиву, замінивши нулями.

30. Якщо максимальний елемент першої половини масиву має непарний номер і є парним числом, то виконати таку обробку масиву: елементи з непарними значеннями зібрати в кінці масиву, зберігши їхній порядок; решту елементів розмістити на початок масиву, замінивши нулями.

Практичне заняття № 19 (*комп'ютерний практикум № 13*)
Сортування в одновимірних масивах

Мета заняття: набути практичних навичок детально розробляти і програмувати алгоритми сортування одновимірних масивів, а також створювати програми модульної структури, в яких реалізуються ці алгоритми, і програмувати їх засобами мови С з використанням функцій.

Завдання. Розробити алгоритм сортування цілочисельного масиву, застосовуючи вказаний у варіанті метод (див. Додаток Г). Усі дії виконувати в тому самому масиві — робочі масиви не використовувати. Запрограмувати розроблений алгоритм у вигляді функції

Розробити функцію введення в діалоговому режимі елементів масиву, функцію заповнення масиву випадковими числами, функцію виведення масиву (функцію виведення використовувати для виведення початкового і відсортованого масивів).

У функціях глобальні змінні не використовувати. У функції сортування не здійснювати введення і виведення інформації.

У головній функції в діалоговому режимі задати реальну кількість елементів масиву (максимальну кількість елементів масиву визначити директивою #define); передбачити вибір у діалоговому режимі способу заповнення масиву і звернення до відповідної функції; звернення до функції виведення масиву для виведення початкового масиву; звернення до функції сортування масиву; звернення до функції виведення масиву для виведення масиву після

обробки; передбачити можливість перевірки помилок (до введення правильних даних), багаторазового виконання програми.

Коментарі в програмі обов'язкові (17-25%).

За алгоритмом провести розрахунки не менш ніж з трьома різними наборами вхідних даних.

Крім алгоритму сортування свого варіанту, треба знати всі решта алгоритми.

Варіанти завдань

1. Метод бульбашки з індикатором перестановки і запам'ятовуванням позиції останньої перестановки для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину на початку масиву.

2. Шейкер-сортування для впорядкування елементів масиву за зростанням.

3. Обмінне сортування простою вибіркою з пошуком максимуму для впорядкування елементів масиву за спаданням.

4. Метод бульбашки з індикатором перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

5. Сортування простими вставками для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину в кінці масиву.

6. Метод бульбашки з індикатором перестановки і запам'ятовуванням позиції останньої перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

7. Шейкер-сортування з індикатором перестановки для впорядкування елементів масиву за зростанням.

8. Обмінне сортування простою вибіркою з пошуком максимуму для впорядкування елементів масиву за зростанням.

9. Метод бульбашки з індикатором перестановки для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину в кінці масиву.

10. Сортування простими вставками для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

11. Метод бульбашки з індикатором перестановки і запам’ятовуванням позиції останньої перестановки для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину в кінці масиву.

12. Шейкер-сортування для впорядкування елементів масиву за спаданням.

13. Обмінне сортування простою вибіркою з пошуком мінімуму для впорядкування елементів масиву за спаданням.

14. Метод бульбашки з індикатором перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину в кінці масиву.

15. Сортування простими вставками для впорядкування елементів масиву за спаданням, розміщую чи вихідну множину в кінці масиву.

16. Метод бульбашки з індикатором перестановки і запам’ятовуванням позиції останньої перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину в кінці масиву

17. Шейкер-сортування з індикатором перестановки для впорядкування елементів масиву за спаданням.

18. Обмінне сортування простою вибіркою з пошуком мінімуму для впорядкування елементів масиву за зростанням.

19. Метод бульбашки з індикатором перестановки для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину на початку масиву.

20. Сортування простими вставками для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

21. Метод бульбашки з запам’ятовуванням позиції останньої перестановки й індикатором перестановки для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину на початку масиву.

22. Шейкер-сортування для впорядкування елементів масиву за зростанням.

23. Обмінне сортування простою вибіркою з пошуком мінімуму для впорядкування елементів масиву за спаданням.

24. Метод бульбашки з індикатором перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

25. Сортування простими вставками для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину в кінці масиву.

26. Метод бульбашки з запам’ятовуванням позиції останньої перестановки й індикатором перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

27. Шейкер-сортування з індикатором перестановки для впорядкування елементів масиву за зростанням.

28. Обмінне сортування простою вибіркою з пошуком мінімуму й максимуму для впорядкування елементів масиву за зростанням.

29. Обмінне сортування простою вибіркою з пошуком мінімуму й максимуму для впорядкування елементів масиву за спаданням.

30. Обмінне сортування простою вибіркою з пошуком максимуму для впорядкування елементів масиву за зростанням.

Практичне заняття № 20
Обробка двовимірних масивів

Мета заняття: закріпити лекційний матеріал і виробити навички застосування типових алгоритмів при обробці двовимірних масивів; навчитися працювати з секторами матриць, застосовуючи різні підходи.

1. Робота з двовимірними масивами. Двовимірний масив — це масив із масивів. Двовимірні масиви можна іетерпретувати як матриці, які є наборами вектор-рядків. Елементи двовимірного масиву розташовуються в пам’яті в порядку зростання другого індексу, тобто, по рядках. Доступ до окремого елемента масиву здійснюється за допомогою імені масиву й індексів. Індекси елемента масиву подаються кожен у своїх квадратних дужках після імені.

Індексом елемента може бути будь-який вираз цілого типу int, типу char чи типу enum. У всіх масивів індексація елементів починається з індекса 0. Введення й виведення масивів здійснюється поелементно.

Якщо параметром функції є двовимірний масив, то її формальний параметр можна описати двома способами:

1) як масив фіксованого розміру — `void func(int matr[3][5]);`

2) як масив невизначеного розміру — `void func(int matr[][5]);`

Звернення до функції в обох випадках буде однаковим: `func(matr)`.

У випадку багатовимірних масивів в оголошенні параметрів функції обов'язково треба вказати всі розміри вимірів, крім найлівішого. Ці розміри потрібні компілятору для того, щоб всередині функції правильно обчислити адресу елемента масиву. У функцію передається тільки вказівник на початковий елемент масиву (тобто елемент з нульовими індексами).

Приклад 1.1. Обхід матриці ланцюжком (змійкою) по рядках.

```
#include <stdio.h>
#include <stdlib.h>
#define N 7
void vyvid(int a[N][N], int n);
void zmiika(int a[N][N], int n);
/* Нумерація елементів масиву змійкою по рядках */
int main() {
  int a[N][N]={0};   // заповнення матриці нулями
  int n;   // заданий розмір квадратної матриці
  system("chcp 1251 & cls");
  printf("Нумерація елементів масиву змійкою\n");
  printf("Задайте розмір матриці n:  ");
  scanf("%d",&n);
  zmiika(a,n);
  printf("\nРезультат:\n");
  vyvid(a,n);
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
/* побудова витків змійки (спочатку зліва направо,
   потім справа наліво і т.д.) */
void zmiika(int a[N][N], int n) {
  int i, k,   // індекси елементів масиву
         m;   // номер елемента при обході
  m=1;   // початкове значення
  for (k=0; k<n; k++)    // Побудова витків змійки
    // Рядок змійки з непарним індексом
    if (k%2) for (i=n-1; i>=0; i--) a[k][i]=m++;
    // Рядок змійки з парним індексом
    else for (i=0; i<n; i++) a[k][i]=m++;
}
void vyvid(int a[N][N], int n) {
  int i, j;  // індекси елементів масиву
  //printf("\n");
  for (i=0; i<n; i++) {
    for (j=0; j<n; j++)
      printf("%5d",a[i][j]);
    printf("\n");
  }
}
```

```
Нумерація елементів масиву змійкою
Задайте розмір матриці n:  6

Результат:
    1    2    3    4    5    6
   12   11   10    9    8    7
   13   14   15   16   17   18
   24   23   22   21   20   19
   25   26   27   28   29   30
   36   35   34   33   32   31
```

Приклад 1.2. Обхід матриці по спіралі за часовою стрілкою

```
#include <stdio.h>
#include <stdlib.h>
#define N 7
void vyvid(int a[N][N], int n);
void spiral(int a[N][N], int n);
/* Нумерація елементів масиву по спіралі за часовою стрілкою */
int main() {
  int a[N][N]={0};   // заповнення матриці нулями
  int n;   // заданий розмір квадратної матриці
  system("chcp 1251 & cls");
  printf("Нумерація елементів масиву по спіралі\n");
  printf("Задайте розмір матриці n:  ");
  scanf("%d",&n);
```

```
  spiral(a,n);
  printf("\nРезультат:\n");
  vyvid(a,n);
  printf("\n\n");
  system("pause");
  return 0;
}
/* побудова витків спіралі за часовою стрілкою */
void spiral(int a[N][N], int n) {
  int i, j, k, p, m ;  // індекси елементів масиву
  p=n/2;   // кількість витків спіралі
  m=1;   // початкове значення
  for (k=0; k<p; k++) {   // Побудова витків спіралей
    // Верхній рядок витка (заповнення зліва направо)
    for (j=k; j<n-k; j++) a[k][j]=m++;
    // Правий стовпчик витка (заповнення згори вниз)
    for (i=k+1; i<n-k; i++) a[i][n-k-1]=m++;
    // Нижній рядок витка (заповнення справа наліво)
    for (j=n-k-2; j>=k; j--) a[n-k-1][j]=m++;
    // Лівий стовпчик витка (заповнення знизу вгору)
    for (i=n-k-2; i>k; i--) a[i][k]=m++;
  }
  if (n%2) a[p][p]=n*n;   // заповнення центру при непарному n
}
void vyvid(int a[N][N], int n) {
  int i, j;  // індекси елементів масиву
  for (i=0; i<n; i++) {
    for (j=0; j<n; j++)
      printf("%5d",a[i][j]);
    printf("\n");
  }
}
```

```
Нумерація елементів масиву по спіралі
Задайте розмір матриці n:  7

Результат:
    1    2    3    4    5    6    7
   24   25   26   27   28   29    8
   23   40   41   42   43   30    9
   22   39   48   49   44   31   10
   21   38   47   46   45   32   11
   20   37   36   35   34   33   12
   19   18   17   16   15   14   13
```

Приклад 1.3. Транспонування квадратної матриці.

```
#include <stdio.h>
#include <stdlib.h>
#define N 7
```

```
void vyvid(int a[N][N], int n, int m);
/* Транспонування матриці. Виконується шляхом перестановки
   відносно головної діагоналі нижньої і верхньої частин
   матриці симетрично: при цьому розглядається тільки
   верхня трикутна матриця без діагоналі */
int main() {
  int a[N][N];
  int i,j,r;
  int n;   // робочий розмір матриці
  system("chcp 1251 & cls");
  hrintf("Вкажіть розмір матриці  ");
  scanf("%d",&n);
  for (i=0, r=1; i<n; i++)
    for (j=0; j<n; j++)
      a[i][j]=r++; // заповнення матриці
  printf("\nПочаткова матриця:\n");
  vyvid(a,n,n);
  for (i=0; i<n-1;  i++)
    for (j=i; j<n; j++) {   // має бути симетрична перестановка
      r=a[i][j];
      a[i][j]=a[j][i];
      a[j][i]=r;
    }
  printf("\nОброблена матриця:\n");
  vyvid(a,n,n);
  printf("\n\n");
  system("pause");
  return 0;
}
void vyvid(int a[N][M], int n,int m) {
  int i, j;  // індекси елементів масиву
  for (i=0; i<n; i++) {
    for (j=0; j<m; j++)
      printf("%5d",a[i][j]);
    printf("\n");
  }
}
```

```
Вкажіть розмір матриці  5

Початкова матриця:
    1    2    3    4    5
    6    7    8    9   10
   11   12   13   14   15
   16   17   18   19   20
   21   22   23   24   25

Оброблена матриця:
    1    6   11   16   21
    2    7   12   17   22
    3    8   13   18   23
    4    9   14   19   24
    5   10   15   20   25
```

Завдання. Написати програми, в яких здійснюється обробка двовимірних масивів.

1. Здійснити обхід матриці ланцюжком (змійкою) паралельно до побічної діагоналі.

2. Побудувати магічний квадрат (сума елементів кожного рядка, кожного стовпчика, головної й побічної діагоналей рівні) непарного розміру, застосувавши поданий алгоритм.

У будь-якому магічному квадраті перше число, тобто 1, зберігається в клітинці (n/2, n-1). Нехай ця позиція буде (i, j). Наступне число зберігається в позиції (i-1, j+1), де при цьому кожен рядок і стовпчик розглядається як круговий масив.

При заповненні квадрата мають виконуватися три умови:

(1) позиція наступного номера розраховується шляхом зменшення номера рядка попереднього числа на 1 і збільшення номера стовпчика попереднього числа на 1. У будь-який час, якщо розрахована позиція рядка стає рівною -1, то вона буде обертатися до n-1 Так само, якщо розрахункова позиція стовпчика стає рівною n, то вона буде обертатися до 0;

(2) якщо магічний квадрат вже містить число у розрахунковій позиції, розрахункова позиція стовпчика зменшується на 2, а розрахована позиція рядка збільшується на 1;

(3) якщо розрахована позиція рядка дорівнює -1, і обчислена позиція стовпчика дорівнює n, то нова позиція буде: (0, n-2).

3. Знайти добуток двох прямокутних матриць $C_{m\times n} = A_{m\times k} B_{k\times n}$.

2. Сортування у двовимірних масивах. Методи сортування можна застосовувати не тільки до одновимірнмх масивів, а й до масивів більших вимірностей.

Приклад 2.1. Сортування елементів матриці методом бульбашки за зростанням.

Код сортування одновимірного масиву методом бульбашки (див. Додаток Г) такий:

```
void sortbul(int *a, int n) { // метод бульбашки
  int i, k;
  int r;
  for (i=0; i<n-1; i++)
    for (k=0; k<n-i-1; k++)
      if (a[k]>a[k+1]) {
        r=a[k];
        a[k]=a[k+1];
        a[k+1]=r;
    }
}
```

Для сортування двовимірного масиву з наскрізним індексом елементів $k = \overline{0, n \cdot m - 1}$ треба за наскрізним індексом k обчислити індекси (i, j) елементів у двовимірному масиві за формулами: $i = k / m$, $j = k\%m$. Правильність формул можна перевірити на прикладі поданої матриці:

$$\begin{pmatrix} \underset{0,0}{0} & \underset{0,1}{1} & \underset{0,2}{2} \\ \underset{1,0}{3} & \underset{1,1}{4} & \underset{1,2}{5} \end{pmatrix}_{2\times 3} \quad \begin{matrix} n = 2 \\ m = 3 \end{matrix}.$$

```
#include <stdio.h>
#include <stdlib.h>
#define N 3
#define M 6
void vyvid(int a[N][M], int n, int m);
void sortbulMatr(int a[N][M], int nm, int m);
/* Сортування елементів матриці методом бульбашки */
int main() {
  int a[N][M]={12,4,18,15,2,13,9,16,5,17,10,7,1,11,14,8,3,6};
  int i,j,k;
  int n,m;
  n=N; m=M;
  system("chcp 1251 & cls");
  printf("\nПочаткова матриця:\n");
  vyvid(a,n,m);
  sortbulMatr(a, n*m, m);
```

```
  printf("\nЗмінена матриця:\n");
  vyvid(a,n,m);
  printf("\n\n");
  system("pause");
  return 0;
}
/* Сортування масиву методом бульбашки — матриця розглядається
   як одновимірний масив з наскрізною нумерацією елементів
   (по рядках); для доступу до конкретного елемента його
   індекси перераховуються */
void sortbulMatr(int a[N][M], int nm, int m) {
// nm - кількість елементів матриці
// m - кількість елементів у рядку матриці
  int i, k;   // параметри циклів при наскрізному індексі
  int ir, jr;   // індекси (i,j) для наскрізного індекса k
  int ir1, jr1; // індекси (i,j) для наскрізного індекса k+1
  int r;   // робоча змінна для перестановки елементів
  for (i=0; i<nm-1; i++)
    for (k=0; k<nm-i-1; k++) {
      ir=k/m; jr=k%m;
      ir1=(k+1)/m; jr1=(k+1)%m;
      if (a[ir][jr]>a[ir1][jr1]) {
        r=a[ir][jr];
        a[ir][jr]=a[ir1][jr1];
        a[ir1][jr1]=r;}
    }
}
void vyvid(int a[N][M], int n,int m) {
  int i, j;  // індекси елементів масиву
  for (i=0; i<n; i++) {
    for (j=0; j<m; j++)
      printf("%5d",a[i][j]);
    printf("\n");
  }
}
```

```
Початкова матриця:
   12    4   18   15    2   13
    9   16    5   17   10    7
    1   11   14    8    3    6

Змінена матриця:
    1    2    3    4    5    6
    7    8    9   10   11   12
   13   14   15   16   17   18
```

Завдання. Написати програми сортування елементів частин матриці.

1. Відсортувати елементи головної діагоналі квадратної матриці, переставляючи при цьому відповідні стовпчики матриці.

2. Відсортувати елементи квадратної матриці під побічною діагоналлю.

3. *Обхід секторів матриці*. У Додатку Ґ розглядаються два підходи щодо вибору елементів з частин квадратної матриці, які розміщуються між головною і побічною діагоналями. Один підхід є оптимальним — проходи циклів виконуються тільки для тих елементів, які мають бути вибрані. Інший дуже простий підхід базується на тому, що суми й різниці індексів елементів матриці на діагоналях, паралельних головній і побічній діагоналям, постійні.

Приклад 3.1. Занумерувати елементи квадратної матриці, які розміщуються над головною й побічною діагоналями, включаючи частину побічної діагоналі. Реалізувати два підходи — оптимальний і з використанням суми й різниці індексів елементів.

```
#include <stdio.h>
#include <stdlib.h>
#define N 10
void vyvid(int a[N][N], int n);
void sektor1_o(int n);
void sektor1_p(int n);
int main() {
  int n;   // заданий розмір квадратної матриці
  system("chcp 1251 & cls");
  printf("Робота з секторами квадратної матриці\n");
  printf("Задайте порядок матриці n:  ");
  scanf("%d",&n);
  sektor1_o(n);
  sektor1_p(n);
  printf("\n\n");
  system("pause");
  return 0;
}
void vyvid(int a[N][N], int n) {
```

```
  int i, j;  // індекси елементів масиву
  for (i=0; i<n; i++) {
    for (j=0; j<n; j++)
      printf("%3d",a[i][j]);
    printf("\n");
  }
}
void sektor1_o(int n) {   // оптимальний обхід
  int a[N][N]={0};
  int k=1;   // номер елемента
  int i, j;  // індекси елементів масиву
for (i=0; i<(n+1)/2; i++)
    for (j=i+1; j<n-i; j++) a[i][j]=k++;
  printf("\nОптимальний алгоритм:\n");
  vyvid(a,n);
}
void sektor1_p(int n) {   // простий алгоритм
  int a[N][N]={0};
  int k=1;   // номер елемента
  int i, j;  // індекси елементів масиву
for (i=0; i<n; i++)
    for (j=0; j<n; j++)
      if (i+j<=n-1 && i-j<0) a[i][j]=k++;
  printf("\nПростий алгоритм:\n");
  vyvid(a,n);
}
```

```
Робота з секторами квадратної матриці
Задайте порядок матриці n:  6

Оптимальний алгоритм:
  0  1  2  3  4  5
  0  0  6  7  8  0
  0  0  0  9  0  0
  0  0  0  0  0  0
  0  0  0  0  0  0
  0  0  0  0  0  0

Простий алгоритм:
  0  1  2  3  4  5
  0  0  6  7  8  0
  0  0  0  9  0  0
  0  0  0  0  0  0
  0  0  0  0  0  0
  0  0  0  0  0  0
```

```
Робота з секторами квадратної матриці
Задайте порядок матриці n:  7

Оптимальний алгоритм:
  0  1  2  3  4  5  6
  0  0  7  8  9 10  0
  0  0  0 11 12  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0

Простий алгоритм:
  0  1  2  3  4  5  6
  0  0  7  8  9 10  0
  0  0  0 11 12  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
```

Приклад 3.2. Заповнити п’ятірками нижню частину квадратної матриці між діагоналями (діагоналі враховувати), проведеними через будь-який елемет a[k][m].

```
#include <stdio.h>
#include <stdlib.h>
#define N 7
#define M 7
void vyvid(int a[N][N], int n, int m);
/* Робота з нижньою частиною квадратної матриці між
   діагоналями (діагоналі враховувати),
   проведеними через будь-який елемент */
int main() {
  int a[N][N]={0};   // заповнення матриці нулями
  int i, j;  // індекси елементів масиву
  int jl, jp;   // індекси колонок зліва і справа
  int n,   // робочий розмір матриці
      k,m;   // індекси вказаного елемента
  n=N-2;
  system("chcp 1251 & cls");
  printf("\nВкажіть порядок матриці n ");
  scanf("%d",&n);
  printf("\nЗадайте індекси елемента m і k ");
  scanf("%d %d",&m,&k);
  a[k][m]=5;
  jl=m;jp=m;
  /* заповнюється п’ятірками частина матриці разом з діаго-
налями */
  for (i=k; i<n;  i++) {   // якщо i=k+1, то без діагоналей
    for (j=jl; j<=jp; j++)
      a[i][j]=5;
    if (jl>0) jl--;
    if (jp<n-1) jp++;  printf("\n");
  }
  printf("\nОброблена матриця:\n");
  for (i=0; i<n; i++) {
```

```
    for (j=0; j<m; j++)
      printf("%5d",a[i][j]);
    printf("\n");
  }
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Вкажіть порядок матриці n: 7
Задайте індекси елемента k i m: 0 2

Оброблена матриця:
  0  0  5  0  0  0  0
  0  5  5  5  0  0  0
  5  5  5  5  5  0  0
  5  5  5  5  5  5  0
  5  5  5  5  5  5  5
  5  5  5  5  5  5  5
  5  5  5  5  5  5  5
```

```
Вкажіть порядок матриці n: 7
Задайте індекси елемента k i m: 1 4

Оброблена матриця:
  0  0  0  0  0  0  0
  0  0  0  0  5  0  0
  0  0  0  5  5  5  0
  0  0  5  5  5  5  5
  0  5  5  5  5  5  5
  5  5  5  5  5  5  5
  5  5  5  5  5  5  5
```

```
Вкажіть порядок матриці n: 7
Задайте індекси елемента k i m: 4 5

Оброблена матриця:
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  5  0
  0  0  0  0  5  5  5
  0  0  0  5  5  5  5
```

```
Вкажіть порядок матриці n: 7
Задайте індекси елемента k i m: 3 3

Оброблена матриця:
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  0  0  0  0
  0  0  0  5  0  0  0
  0  0  5  5  5  0  0
  0  5  5  5  5  5  0
  5  5  5  5  5  5  5
```

Завдання. Написати програму заповнення п’ятірками верхньої частини квадратної матриці між діагоналями (діагоналі враховувати), проведеними через будь-який елемент a[k][m].

Практичне заняття № 21 (*комп’ютерний практикум № 14*)

Обробка двовимірних масивів

Мета заняття: набути практичних навичок детально розробляти і програмувати алгоритми обробки двовимірних масивів, а також створювати програми модульної структури, в яких реалізуються ці алгоритми, і програмувати їх засобами мови С з використанням функцій.

Завдання. Розробити алгоритми обробки двовимірного цілочисельного масиву (матриці $n \times n$; значення n користувач задає в діалоговому режимі; час-

тини матриць вибирати за поданими малюнками) відповідно до двох частин завдання варіанту, запрограмувати їх у вигляді функцій.

Розробити функцію введення в діалоговому режимі елементів двовимірного масиву, функцію заповнення масиву випадковими числами, функцію виведення масиву.

У головній функції передбачити вибір у діалоговому режимі способу заповнення масиву і звернення до відповідної функції; введення додаткової інформації; звернення до функції виведення масиву для виведення початкового масиву — елементи сектора, який відповідає першій частині завдання, виділити, наприклад, *5*; умовою вибирати суму і різницю індексів (див. Додаток Ґ); звернення до функцій обробки масиву відповідно до двох частин завдання, застосовуючи оптимальний перегляд секторів (див. Додаток Ґ); виведення результату першої частини завдання; виведення порахованої кількості в другій частині завдання; звернення до функції виведення масиву для виведення масиву після обробки.

Глобальні змінні не використовувати. Коментарі в програмі обов'язкові (17-25%).

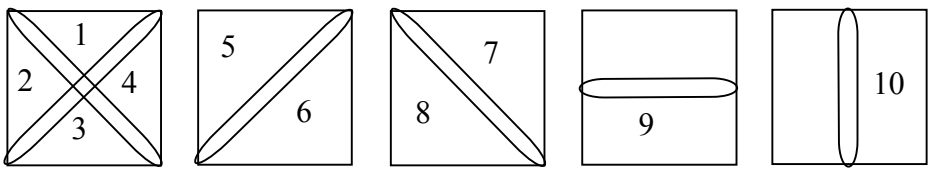
За алгоритмом провести розрахунки не менш ніж з двома різними наборами вхідних даних для парного значення n і не менш ніж з двома різними наборами вхідних даних для непарного значення n.

При захисті роботи треба вміти описати обхід будь-якого сектора з будь-якого варіанту.

Зауваження. Для елемента з індексами (i, j) в масиві його номер $nom = i \cdot n + j + 1$; звідси індекси $i = (nom - 1)/n$, $j = nom - i \cdot n - 1$

Варіанти завдань

Використано позначення: $\wedge 3$ — сектор 3 з ементами обох діагоналей; 3\ — з елементами головної діагоналі; 3 — без діагональних елемнентів тощо.



1. У секторі \7 знайти елемент (вказати його номер), значення якого найбільш відрізняється від заданого числа. У секторі 1/ порахувати кількість елементів і елементи замінити нулями.

2. У секторі 8\ порахувати кількість чисел, рівних за абсолютною величиною заданому числу. У секторі 2/ порахувати кількість елементів і елементи замінити нулями.

3. У секторі 2 порахувати кількість елементів, значення яких менші від заданого числа. У секторі 9 (якщо *n* непарне, то без —) порахувати кількість елементів і елементи замінити нулями.

4. У секторі 1 порахувати суму непарних чисел. У секторі 10 (якщо *n* непарне, то з |) порахувати кількість елементів і елементи замінити нулями.

5. У секторі 3 знайти номер першого елемента, значення якого ділиться на задане число без остачі. У секторі 8\ порахувати кількість елементів і елементи замінити нулями.

6. У секторі 4 порахувати кількість елементів, значеннями яких є парні числа. У секторі \7 порахувати кількість елементів і елементи замінити нулями.

7. У секторі ∨1 порахувати кількість елементів з від'ємними значеннями. У секторі /6 порахувати кількість елементів і елементи замінити нулями.

8. У секторі 2> порахувати кількість елементів, значення яких рівні нулю. У секторі 5 порахувати кількість елементів і елементи замінити нулями.

9. У секторі /4 знайти номер першого елемента, значення якого ділиться на задане число без остачі. У секторі 7 порахувати кількість елементів і елементи замінити нулями.

10. У секторі 3\ порахувати кількість елементів, значення яких менші від заданого числа. У секторі 9 (якщо *n* непарне, то з —) порахувати кількість елементів і елементи замінити нулями.

11. У секторі /3 порахувати кількість елементів, значення яких менші від заданого числа. У секторі 6 порахувати кількість елементів і елементи замінити нулями.

12. У секторі 4\ порахувати кількість елементів, значення яких кратні заданому числу. У секторі 8\ порахувати кількість елементів і елементи замінити нулями.

13. У секторі \4 порахувати кількість елементів, значеннями яких є непарні числа. У секторі 10 (якщо *n* непарне, то з |) порахувати кількість елементів і елементи замінити нулями.

14. У секторі 5/ знайти найменший додатний елемент (вказати його номер). У секторі 2> порахувати кількість елементів і елементи замінити нулями.

15. У секторі /6 знайти найбільший від'ємний елемент (вказати його номер). У секторі < 4 порахувати кількість елементів і елементи замінити нулями.

16. У секторі 5 порахувати суму елементів, значення яких більші від заданого числа. У секторі 3\ порахувати кількість елементів і елементи замінити нулями.

17. У секторі 6 порахувати суму значень елементів, які діляться на 3 без остачі. У секторі /4 порахувати кількість елементів і елементи замінити нулями.

18. У секторі 1/ порахувати кількість елементів, значення яких більші від заданого числа. У секторі 5/ порахувати кількість елементів і елементи замінити нулями.

19. У секторі 2/ знайти найменший елемент (вказати його номер). У секторі /6 порахувати кількість елементів і елементи замінити нулями.

20. У секторі 7 порахувати кількість елементів, значеннями яких є парні числа. У секторі $\vee 1$ порахувати кількість елементів і елементи замінити нулями.

21. У секторі $\wedge 3$ порахувати кількість елементів, значеннями яких є натуральні числа. У секторі 8 порахувати кількість елементів і елементи замінити нулями.

22. У секторі $5/$ знайти суму всіх елементів, значення яких менші від заданого числа. У секторі 1 порахувати кількість елементів і елементи замінити нулями.

23. У секторі 8 порахувати суму значень елементів, які діляться на 3 без остачі. У секторі 2 порахувати кількість елементів і елементи замінити нулями.

24. У секторі \1 порахувати суму непарних чисел. У секторі 9 (якщо *n* непарне, то без —) порахувати кількість елементів і елементи замінити нулями.

25. У секторі 2\ порахувати кількість елементів, значення яких менші від заданого числа. У секторі 10 (якщо *n* непарне, то з |) порахувати кількість елементів і елементи замінити нулями.

26. У секторі < 4 порахувати кількість елементів масиву, значення яких рівні заданому числу. У секторі \7 порахувати кількість елементів і елементи замінити нулями.

27. У секторі 9 (якщо *n* непарне, то з —) порахувати кількість непарних чисел. У секторі /4 порахувати кількість елементів і елементи замінити нулями.

28. У секторі 10 (якщо *n* непарне, то з |) порахувати кількість елементів з від'ємними значеннями. У секторі /3 порахувати кількість елементів і елементи замінити нулями.

29. У секторі 9 порахувати кількість елементів, значення яких рівні нулю. У секторі 2\ порахувати кількість елементів і елементи замінити нулями.

30. У секторі 10 порахувати кількість елементів, остача від ділення значень яких на задане число дорівнює 3. У секторі \1 порахувати кількість елементів і елементи замінити нулями.

Практичне заняття № 22
Робота з рядками символів. Побітові операції. Робота зі структурами

Мета заняття: закріпити лекційний матеріал і виробити навички застосування типових алгоритмів при роботі з рядками символів; розглянути типові задачі використання побітових операцій; виробити навички роботи зі структурами.

1. Робота з рядками символів. На відміну від інших мов мова С не має спеціального типу даних — рядки символів. У мові С рядок символів моде-

люється за допомогою одновимірного масиву символів. Найважливішою особливістю для масиву, який зберігає рядок символів, є те, що після всіх символів у кінці рядка записується спеціальний символ '\0' (нуль-символ) з числовим кодом 0, який відіграє роль ознаки кінця рядка.

При виведенні рядка на екран функцією printf використовується формат %s: `char a[5]="abcd"; printf("Текст %s", a);`. Виведення окремого символа рядка здійснюється за форматом %c: `printf("%c", a[i]);`.

Для введення рядків символів можна скористатися різними функціями бібліотеки stdio.h. Розглянемо найосновніші.

У функції scanf, як і в функції printf, використовується формат %s. При цьому зчитуються всі введені символи до першого пробільного символа (пробіл ' ', Tab '\t', Enter '\n'). Якщо для рядка відведено менше пам'яті, ніж вводиться символів, то, оскільки мова С не підтримує автоматичного контролю виходу за границю масиву, символи будуть записуватися не тільки в пам'ять, виділену під масив, а й у комірки пам'яті безпосередньо після масиву і, отже, можуть зіпсувати значення інших змінних або коди команд. Щоб запобігти цьому, треба в форматі вводу вказати граничну довжину рядка (на 1 меншу від розміру рядка або ще меншу).

```
char b[10];
/* вводяться символи до першого пробільного,
   але не більше 9 */
scanf("%9s",b);
```

решта набраних користувачем символів залишається в буфері вводу. Треба також враховувати, що scanf залишає в буфері пробільний символ.

Для введення текстів з пробілами і знаками табуляції можна скористатися функцією gets, яка вводить всі символи до натискання Enter і має один параметр — вказівник на масив символів:

```
char b[10];
gets(b);   // вводяться символи до першого Enter
```

Але функція gets не підтримує можливості задання граничної довжини тексту, що може призвести до виходу за межі пам'яті рядкового масиву.

Функція fgets вводить текст, який містить пробіли і знаки табуляції, до натискання Enter в межах заданої довжини. Функція має 3 параметри: вказівник на масив символів, цілочисельний вираз — гранична довжина тексту *n* (зчитуються перші *n*-1 символів, а решта залишаються в буфері), третій аргумент вказує файл введення (при введенні тексту з клавіатури — stdin):

```
char b[10];
int i=3;
fgets(b,2*i,stdin);   // буде зчитано 5 символів тексту
```

Функція getchar повертає наступний символ зі стандартного вводу (stdin), включаючи символи пробілу, Tab і Enter; аргументів не має:

```
#define N 10
char b[N];
i=0;
while (i<N-1) {
  b[i]=getchar(); i++;
}
b[N-1]=0;
```

Для обробки рядків символів можна скористатися функціями бібліотеки string.h. Розглянемо найосновніші.

Функцію strlen використовують для визначення реальної довжини рядка; має один аргумент — вказівник на рядок; повертає довжину рядка — кількість справжніх символів (без нуль-символа). Реально рядок може містити менше символів, ніж зарезервовано:

```
char b[10]="abc";   // зарезервовано пам'ять для
                    // 9 символів, зайнято 4 байти
n=strlen(b);   // значення n=3
```

Оскільки в мові С операція присвоювання для рядків не працює, то для копіювання рядків використовують функцію strcpy, яка має два аргументи —

вказівники на рядки s1 і s2. Дія функції аналогічна присвоюванню s1=s2. При цьому не здійснюється контроль виходу за межі масиву:

```
char a[10]="abc", b[10]="de";
strcpy(a,b);   // рядок a має значення "de"
strcpy(b,"копія");   // рядок b має значення "копія"
```

Функцію strcat використовують для зчеплення (конкатенації) двох рядків; має два аргументи — вказівники на рядки s1 і s2. Дія функції аналогічна дії s1=s1+s2. При цьому не здійснюється контроль виходу за межі масиву:

```
char a[10]="abc", b[10]="de";
strcat(a,b);   // рядок a набуває значення "abcde",
               // рядок b залишає значення "de"
```

У мові С не можна порівнювати рядки за допомогою операцій порівняння (==, <, >, !=). Для порівняння двох рядків (порівнюються послідовності символів за вказаними адресами) використовується функція strcmp, яка має два аргументи — вказівники на рядки s1 і s2; повертає результат порівняння: 0 — якщо рядки рівні; від'ємне значення -1 — якщо s1<s2; додатне значення 1 — якщо s1>s2:

```
char a[10]="abc", b[10]="de";
ind=strcmp(a,b);   // ind набуває значення -1
```

Функцію sprintf використовують для виведення значення в текстовий рядок. Першим аргументом функції є вказівник на рядок, а наступні параметри такі ж, як у функції printf:

```
char a[10]="abc", b[10]="de";
char str[80];
/* str набуває значення: "Порівняння abc і de дає -1" */
sprintf(str,"Порівняння %s і %s дає %d",a,b,strcmp(a,b));
```

У задачах обробки текстової інформації часто виникає потреба розпізнати, чи є символ літерою, великою чи малою, цифрою, знаком пунктуації тощо. Такі класифікаційні функції (правда, тільки для англійського алфавіту) містяться в бібліотеці ctype.h. Фактичним параметром такої функції є константа чи змінна символьного типу, а результатом значен-

ня 0 (якщо значення параметра не є символом певного типу) чи 1 (якщо є символом певного типу). Ці функції вказано в таблиці:

isalnum	символ є буквено-цифровим (мала чи велика буква англійського алфавіту чи цифра)
isalpha	символ є буквою (мала чи велика буква англійського алфавіту)
islower	символ є малою буквою англійського алфавіту
isupper	символ є великою буквою англійського алфавіту
iscntrl	символ є керуючим символом (має код в межах від 0 до 31 чи дорівнює 127 (Del))
isdigit	символ є десятковою цифрою (цифра від 0 до 9)
isgraph	символ має графічне подання (є друкованим символом, відмінним від пробілу; має код від 33 до 126)
isprint	символ для друку (є друкованим символом, включаючи пробіл; має код від 32 до 126)
ispunct	символ є символом пунктуації (є знаком пунктуації чи пробілом)
isspace	символ є пробільним (пробіл, повернення каретки, горизонтальна табуляція, вертикальна табуляція, переведення нового рядка)
isxdigit	символ є шістнадцятковою цифрою (від A до F або від a до f і від 0 до 9.)

Функції бібліотеки ctype.h не розраховані на роботу з кириличними символами. Крім того, наприклад, функція `isdigit(str[i])` при `char str[N]` не сприймає кирилицю — під час обробки видає повідомлення про помилку; якщо задати `isdigit(str[i]&255)`, а також при `unsigned char str[N]` функція `isdigit(str[i])` повідомлення про помилку не видає.

Також в бібліотеці ctype.h є функції, зміни регістру букви (працюють тільки для англійського алфавіту; при обробці кириличних букв повідомлення про помилку не видають, але й не змінюють регістру):

tolower	приводить символ до нижнього регістру
toupper	приводить символ до верхнього регістру

Приклад 1.1. Знайти позицію першого входження послідовності символів (слова) в текст. Кінець тексту позначає символ #.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define N 200
void vvidstr(char text[N], int *nt);
int substr(char text[N], int nt, char slovo[N], int ns);
int main() {
/* Визначення позиції початку входження послідовності
   символів (слова) у текст */
  int nt, ns,  // кількість символів у тексті й слові
      ind;   // позиція входження слова в текст
  char text[N];   // введений текст
  char slovo[N];   // введене слово
  system("chcp 1251 & cls");
  /* ввід тексту до граничної кількості символів чи символа #
     (після # можуть бути ще символи, але вони не вводяться) */
  printf(" Введіть текст (до  #)\n");
  vvidstr(text,&nt);
  /* ввід слова до граничної кількості символів чи символа # */
  printf("\n Введіть слово (до #)\n");
  vvidstr(slovo,&ns);
  printf("\n Введений текст:\n %s\n слово: %s\n",text,slovo);
  /* визначення входження */
  ind=substr(text,nt,slovo,ns);
  printf("\n Позиція входження слова в текст: %d",ind);
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
/* ввід текстової інформації до символа #, але не більше,
   ніж N символів */
void vvidstr(char text[N], int *nt) {
  fseek(stdin,0,SEEK_END);   // очистка буфера вводу
  *nt=-1;
  while (*nt<N-1 && text[*nt]!='#') {
    (*nt)++;   // кількість введених символів
    text[*nt]=getchar();
  }
  text[*nt]='\0';   // кінець рядка символів
}
/* визначення першого входження слова в текст */
int substr(char text[N], int nt, char slovo[N], int ns) {
  int i,j,
      nom;   // номер позиції першого входження
  for (i=0, j=0; i<nt-ns && j!=ns; i++)
    for (j=0; j<ns && text[i+j]==slovo[j]; j++);
    if (j==ns) nom=i; else nom=0;
  return nom;
}
```

```
 Введіть текст (до  #)
Життя прикрашають дві речі: можливість вивчати математику й можливість
викладати її (Сімеон-Дені Пуассон; 1781-1840) # французький фізик і математик

 Введіть слово (до #)
вість#

 Введений текст:
 Життя прикрашають дві речі: можливість вивчати математику й можливість
викладати її (Сімеон-Дені Пуассон; 1781-1840)
 слово: вість

 Позиція входження слова в текст: 34
```

```
 Введіть текст (до  #)
Моя любов - Україна й математика
   (М.П. Кравчук; 1892-1942) # український математик, репресований

 Введіть слово (до #)
край#

 Введений текст:
 Моя любов - Україна й математика
   (М.П. Кравчук; 1892-1942)
 слово: край

 Позиція входження слова в текст: 0
```

```
 Введіть текст (до  #)
Розв'язування математичних задач може викликати звичку думати
   # (з Інтернету)

 Введіть слово (до #)
я математи#к

 Введений текст:
 Розв'язування математичних задач може викликати звичку думати

 слово: я математи

 Позиція входження слова в текст: 13
```

Приклад 1.2. За введеним номером дня визначити його назву і його статус — робочий чи неробочий. Використати перераховувані дані типу enum і масив рядків символів.

```
#include <stdio.h>
#include <stdlib.h>
int main() { // Визначення дня тижня і його статусу за номером
  int dd;   // введена інформація - номер дня тижня
  enum {Mon,Tue,Wed,Thu,Fri,Sat,Sun};
  char name[][20]={"понеділок", "вівторок", "середа",
       "четвер", "п'ятниця", "субота", "неділя"};
  system("chcp 1251 & cls");
  printf("Введіть номер дня тижня (понеділок - 1)");
  while (printf("\n "), scanf("%d", &dd)) {
    dd--;
    switch (dd) {
      case Sat: case Sun:
                printf("\t%s - вихідний день",name[dd]);
                break;
      case Mon: case Tue: case Wed: case Thu: case Fri:
                printf("\t%s - робочий день",name[dd]);
                break;
      default:  printf("неправильно вказано номер дня");
    }
  }
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Введіть номер дня тижня (понеділок - 1)
 0
неправильно вказано номер дня
 8
неправильно вказано номер дня
 1
        понеділок - робочий день
 7
        неділя - вихідний день
 кінець
```

Приклад 1.3. Підрахувати кількість англійських слів у введеному тексті.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define N 80
int main() {
/* Підрахунок кількость англійських слів у введеному тексті.
   При char str[N] функція isalpha(str[i]) не сприймає кирилицю
   (при обробці видає помилку); якщо задати isalpha(str[i]&255),
   то сприймає. При unsigned char str[N] функція isalpha(str[i])
   сприймає кирилицю */
  int i,   // поточний символ
      n,   // кількість слів
      ind;   // індикатор слова: 1 - слово, 0 - роздільник
  char str[N+1];
  system("chcp 1251 & cls");
  printf(" Введіть текст до символа #\n");
  i=-1;
  /* ввід тексту до граничної кількості символів чи символа #
     (після # можуть бути ще символи, але вони не вводяться) */
  while (i<N-1 && str[i]!='#') {
    i++;
    str[i]=getchar();
  }
  while(getchar()!='\n');
  /* щоб однотипно обробляти текст, додаємо фіктивний пробіл */
  str[i]=' '; str[i+1]='\0';
  printf("\n Введений текст:\n%s\n",str);
  /* підрахунок кількості іноземних слів */
  n=0; i=0; ind=0;
  while (str[i]!='\0') {
    if (isalpha(str[i]&255) && !ind) ind=1;
    else
      if (!isalpha(str[i]&255) && ind) { ind=0; n++; };
    i++;
  }
  printf("\n Кількість іноземних слів у тексті: %d",n);
```

```
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
 Введіть текст до символа #
Visual Studio -середовище програмування.
Мови програмування C, C++, C#

 Введений текст:
Visual Studio -середовище програмування.
Мови програмування C, C++, C

 Кількість іноземних слів у тексті: 5
```

Програма вибирає тільки англійські слова (Visual, Studio, C, C, C)

```
 Введіть текст до символа #
Мова C, C++, C##

 Введений текст:
Мова C, C++, C

 Кількість іноземних слів у тексті: 0
```

Весь текст набрано на українській розкладці клавіатури

```
 Введіть текст до символа #
Плагіат не пройде!
Символи буде виявлено #

 Введений текст:
Плагіат не пройде!
Символи буде виявлено

 Кількість іноземних слів у тексті: 11
```

У тексті подібні кириличні символи замінено англійськими. Вони розпізнаються як окремі англійські слова (a, ia, e, o, e, C, o, y, e, e, o)

Завдання. Написати програми обробки текстової інформації.

1. Підрахувати кількість пробілів у введеному тексті.

2. Підрахувати кількість цілих чисел у введеному тексті.

3. Замінити в тексті всі подібні за зображенням англійські символи кириличними.

2. Побітові (порозрядні) операції. На відміну від багатьох мов програмування, в мові С, яка розроблялася як мова системного програмування, визначено повний набір побітових операцій. Це такі операції, як (див. Додаток Б, п. 3): порозрядне заперечення ~, зсув вліво << і вправо >>, порозрядне логічне “і” &, порозрядне виключне “або” (додавання за модулем 2) ^, порозрядне логічне “або” |.

Ці операції застосовують тільки до даних типу char або int (до даних типу float, double, void або інших складніших типів побітові операції застосовувати не можна).

Приклад 2.1. Обчислити степені числа 8 від 0 до 20.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
/* Одержання степенів числа 8 від 0 до 20 */
  long long int i=1;   //  число 8 у степені від 0 до 8
  int n=0;   // значення степеня
  system("chcp 1251 & cls");
  printf("Число 8 у степені\n\
  \bстепінь\tзначення\n%5d%21lld", n, i);
  while (n<20) {
    i=i<<3;
    n++;
    printf("\n%5d%21lld", n, i);
  }
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Число 8 у степені
 степінь        значення
    0                   1
    1                   8
    2                  64
    3                 512
    4                4096
    5               32768
    6              262144
    7             2097152
    8            16777216
    9           134217728
   10          1073741824
   11          8589934592
   12         68719476736
   13        549755813888
   14       4398046511104
   15      35184372088832
   16     281474976710656
   17    2251799813685248
   18   18014398509481984
   19  144115188075855872
   20 1152921504606846976
```

Приклад 2.2. Не використовуючи робочої змінної, взаємно поміняти значення двох ділянок пам’яті, які містять цілі числа.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
/* Взаємний обмін значень двох ділянок пам’яті, які
   містять цілі числа */
  int a, b;   // введені числа
  system("chcp 1251 & cls");
  printf("Введіть два числа  ");
  scanf("%d%d", &a, &b);
  // переприсвоєння значень без використання робочої змінної
  a=a^b;
  b=a^b;
  a=a^b;   // або a=a^(b=(b^(a=a^b)));
  printf("Переприсвоєні значення: %d %d", a, b);
```

```
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Введіть два числа  -10 -15
Переприсвоєні значення: -15 -10
```

```
Введіть два числа  5 8
Переприсвоєні значення: 8 5
```

```
Введіть два числа  -1 3
Переприсвоєні значення: 3 -1
```

```
Введіть два числа  5 -2
Переприсвоєні значення: -2 5
```

Завдання. 1. Значення цілої змінної скоротити на множники, рівні 2, не використовуючи операцій ділення і знаходження остачі від ділення (використати операції зсуву).

2. Обчислити кількість одиниць у двійковому поданні цілого числа *x*, реалізувавши обчислення: а) з використанням операцій зсуву; б) кількість одиниць визначається кількістю ітерацій циклу знаходження порозрядної кон'юнкції чисел *x* і *x*-1 (результат присвоювати *x*), доки *x* не стане рівним нулю.

3. Робота зі структурами. Використання структур дає можливість згрупувати кілька логічно зв'язаних змінних, типи яких можуть бути різними, і надати їм одне ім'я. Структури можна присвоювати, передавати функціям як аргументи, функції можуть повертати структуру як результат (мати тип структури). Кожен елемент структури може мати будь-який допустимий тип даних, у тому числі масиви й інші структури. Можна визначати масиви структур. Кожен елемент масиву структур є структурою. Доступ до окремої структури в масиві здійснюється за допомогою імені структури й індекса. Доступ до окремих елементів (полів) структури здійснюється за допомогою складених імен: *ім'я структури.ім'я елемента*.

Приклад 3.1. Визначити масив структур, який містить інформацію про *N* осіб: прізвище, ім'я і вік. Дані про перших *N*-1 особу задати при визначенні масиву структур шляхом ініціалізації, про останню особу — ввести в діалоговому режимі. Усі дані про осіб відсортувати за віком за зростанням (застосувати метод бульбашки; див. Додаток Г).

```
#include <stdio.h>
#include <stdlib.h>
#define N 5
struct dani { // дані про особу
```

```
  char prizv[20];   // прізвище
  char imia[20];   // ім'я
  int vik;   // вік
};   // ; обов'язкова в оголошенні структури
struct dani vvids();
void sortbuls(struct dani osoba[], int n);
void vyvids(struct dani osoba[], int n);
int main() {
/* Робота зі структурами */
  struct dani osoba[N]={
            {"Кублій","Лариса",59},
            {"Ющук", "Іван", 86},
            {"Війтик", "Ольга", 85},
            {"Кублій", "Михайло", 27}
  };
  system("chcp 1251 & cls");
  printf("Введіть дані про останню особу\n");
  printf("\t(прізвище, ім'я і вік особи)\n");
  osoba[N-1]=vvids();
  printf("Початкові дані\n");
  vyvids(osoba, N);
  sortbuls(osoba, N);
  printf("Відсортовані за віком дані\n");
  vyvids(osoba, N);
  printf("\n\n");
  system("pause");
  return 0;
}
struct dani vvids() {
/* ввід даних в елементи структури */
  struct dani danir;
  scanf("%s%s%d", danir.prizv, danir.imia, &danir.vik);
  return danir;
}
void sortbuls(struct dani osoba[], int n) {
```

```
/* Сортування методом бульбашки за елементом "vik" */
  struct dani danir;
  int i, j;
  for (i=0; i<n-1; i++)
    for (j=0; j<n-i-1; j++)
      if (osoba[j].vik>osoba[j+1].vik) {
        danir=osoba[j];
        osoba[j]=osoba[j+1];
        osoba[j+1]=danir;
      }
}
void vyvids(struct dani osoba[], int n) {
  /* вивід полів структури */
  int i;
  for (i=0; i<n; i++)
    printf("\t%s %s %d\n", osoba[i].prizv,
    osoba[i].imia, osoba[i].vik);
}
```

```
Введіть дані про останню особу
        (прізвище, ім'я і вік особи)
Ющук Олег
  55
Початкові дані
        Кублій Лариса 59
        Ющук Іван 86
        Війтик Ольга 85
        Кублій Михайло 27
        Ющук Олег 55
Відсортовані за віком дані
        Кублій Михайло 27
        Ющук Олег 55
        Кублій Лариса 59
        Війтик Ольга 85
        Ющук Іван 86
```

Аналогічний результат можна одержати, якщо функцію vvids, яка повертає значення-структуру, замінити функцією vvidsa, в яку структура передається за адресою:

```
void vvidsa(struct dani *danir);
int main() {
...
  vvidsa(&osoba[N-1]);
...
```

```
}
void vvidsa(struct dani *danir) {
/* ввід даних в поля структури, переданої за адресою */
  scanf("%s%s%d",(*danir).prizv, (*danir).imia,
&(*danir).vik);   // замість (*p).a можна записати p->a
}
```

Завдання. Використовуючи структуру координати точки, створити структуру, яка містить координати трьох вершин трикутника і його площу, обчислену за формулою $S = ((x_1 y_2 + x_2 y_3 + x_3 y_1) - (y_1 x_2 + y_2 x_3 + y_3 x_1))/2$. У створену структуру внести дані і вивести вміст структури на екран.

Практичне заняття № 23 (*комп'ютерний практикум № 15*)

Робота з рядками символів

Мета заняття: набути практичних навичок детально розробляти і програмувати алгоритми обробки рядків символів, а також створювати програми модульної структури, в яких реалізуються ці алгоритми, і програмувати їх засобами мови С з використанням функцій.

Завдання. Розробити алгоритм введення текстової інформації, яка може набиратися через Enter і закінчується вказаним користувачем символом-роздільником (наприклад #; при цьому в рядку після символа-роздільника користувач може вводити інші символи, але вони не будуть сприйматися). Текстова інформація подається українською мовою з будь-якою кількістю пробільних символів (пробіл, Tab, Enter) між словами, числами і розділовими знаками; довжина тексту — до 240 символів. Алгоритм подати у вигляді функції.

Розробити алгоритм обробки рядка символів відповідно до варіанту завдання; при реалізації алгоритму для спрощення й однотипності обробки рядка символів можна використовувати фіктивні символи на початку, в кінці чи в середині рядка; алгоритм подати у вигляді функції; можна використовувати

стандартні функції обробки рядків символів, також можна розробити власні функції. Передбачити можливість багаторазового виконання алгоритму.

Глобальні змінні не використовувати. Всю обробку виконувати в тому самому рядку символів. Коментарі в програмі обов'язкові (17-25%).

За алгоритмом провести розрахунки не менш ніж з трьома різними наборами вхідних даних.

Варіанти завдань

1. Усі сполучники "та" замінити на "й", якщо перед чи після є голосна, у решті випадків — на "і".

2. Якщо в слові якась буква повторюється більше двох разів підряд, то залишити тільки подвоєння.

3. Слова, які починаються з голосної, вилучити з тексту.

4. Якщо в тексті після голосної стоїть сполучник "і", то замінити його на "й".

5. Якщо в середині слова після голосної стоїть апостроф, то його прибрати.

6. Якщо в середині речення стоїть крапка (після неї стоять пробіли і наступне слово починається з малої букви), то замінити її комою.

7. Якщо арифметичні операції виділено пробілами (перед, після чи з обох боків), то прибрати пробіли.

8. У введеному тексті поміняти регістр букв.

9. Якщо після слова, яке закінчується голосною, стоїть прийменник "у", то замінити його на "в".

10. Якщо Enter розриває речення, то його прибрати, а якщо розміщується в кінці речення після крапки, знака питання чи знака оклику, то залишити.

11. Якщо перед розділовим знаком (крім тире, відкриваючої дужки і відкриваючих лапок) стоять пробіли, то їх прибрати.

12. Якщо арифметичні операції не виділені пробілами з обох боків, то їх виділити.

13. Якщо після розділового знаку (крім десяткових дробів) немає пробілу, то його вставити.

14. Якщо перед розділовим знаком, крім тире, стоїть пробіл, то його прибрати.

15.У тексті кожне окреме слово (у слові може вживатися дефіс) замінити позначенням “<слово>”.

16. Якщо речення починається з маленької букви, то замінити її великою.

17. Занумерувати ті абзаци, перше слово у яких починається з голосної.

18. У введеному тексті кожне речення (речення закінчується крапкою, знаком питання чи знаком оклику) подати з нового рядка.

19.У тексті всі числа (дробові числа подаються з комою, але, якщо після коми іде пробіл, то це кінець числа) замінити позначенням “<число>”.

20. У введеному тексті перші букви слів, які починаються з голосної, зробити великими.

21. У введеному тексті тільки на початку речення залишити велику букву, а всі решта замінити малими.

22. Якщо в середині чи в кінці слова міститься велика буква, то її зробити малою.

23. Якщо в тексті використовується сполучник “та”, то замінити його на “і” після приголосної і на “й” після голосної.

24. Якщо в середині слова замість апострофа стоїть щось схоже на нього (", `, ' тощо), то поставити апостроф.

25. Якщо в середині слова стоїть велика буква, то перед нею поставити пробіл.

26. Прибрати в тексті зайві розділові знаки (кілька однакових знаків підряд через пробіл, Tab, Enter чи без).

27. Якщо перед словом, яке починається приголосною, стоїть прийменник “в”, то замінити його на “у”.

28. Якщо в середині слова перед приголосною стоїть апостроф, то його прибрати.

29. Якщо розділові знаки дублюються (підряд чи через кілька пробілів), то залишити тільки один.

30. У введеному тексті кожне слово починати з великої букви.

Практичне заняття № 24 (*комп'ютерний практикум № 16*)

Робота зі структурами

Мета заняття: набути практичних навичок детально розробляти і програмувати алгоритми з використанням структур, а також створювати програми модульної структури, в яких реалізуються ці алгоритми, і програмувати їх засобами мови С з використанням функцій.

Завдання. Визначити масив структур, який містить вказані в завданні варіанту елементи (поля). Дані про кілька перших об'єктів задати при визначенні масиву структур шляхом ініціалізації; дані про наступні о'єкти ввести в діалоговому режимі.

Розробити і запрограмувати у вигляді функцій алгоритми введення даних у структуру і виведення інформації на екран, сортування за вказаними полями (вибір поля здійснювати в діалоговому режимі в головній функції) масиву структур з використанням методу сортування (див. Додаток Г), вказаного у варіанті завдання. Передбачити можливість багаторазового виконання програми. Глобальні змінні не використовувати. Коментарі в програмі обов'язкові (17-25%).

За алгоритмом провести розрахунки не менш ніж з трьома різними наборами вхідних даних.

Варіанти завдань

1. Елементи структури: прізвище й ініціали, стать, соціальний стан, сімейний стан, вік, зріст. Сортувати за елементами: стать, зріст. Застосувати метод сортування: сортування простими вставками для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину в кінці масиву.

2. Елементи структури: прізвище й ініціали, місце народження, місце проживання, вік, стаж роботи. Сортувати за елементами: прізвище, стаж роботи. Застосувати метод сортування: метод бульбашки з індикатором перестановки і запам'ятовуванням позиції останньої перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

3. Елементи структури: назва фірми, місто, індекс, вулиця, номер будинку, код ЄДРПОУ (як цілочисельна інформація). Сортувати за елементами: назва фірми, код ЄДРПОУ. Застосувати метод сортування: шейкер-сортування з індикатором перестановки для впорядкування елементів масиву за зростанням.

4. Елементи структури: прізвище й ініціали, ідентифікаційний код (як символьна інформація), стать, вік, кількість років навчання. Сортувати за елементами: прізвище, кількість років навчання. Застосувати метод сортування: обмінне сортування простою вибіркою з пошуком максимуму для впорядкування елементів масиву за зростанням.

5. Елементи структури: прізвище, ім’я, по батькові, курс, група, середній бал. Сортувати за елементами: прізвище, середній бал. Застосувати метод сортування: метод бульбашки з індикатором перестановки для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину в кінці масиву.

6. Елементи структури: назва тістечок, виробник, сорт, кількість в упаковці, ціна за упаковку. Сортувати за елементами: назва тістечок, кількість в упаковці. Застосувати метод сортування: сортування простими вставками для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

7. Елементи структури: прізвище, ім’я, по батькові, кількість статей, кількість посібників. Сортувати за елементами: прізвище, кількість посібників Застосувати метод сортування: метод бульбашки з індикатором перестановки і запам’ятовуванням позиції останньої перестановки для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину в кінці масиву.

8. Елементи структури: прізвище, ім’я, по батькові, факультет, вік, стипендія. Сортувати за елементами: прізвище, стипендія. Застосувати метод сортування: шейкер-сортування для впорядкування елементів масиву за спаданням.

9. Елементи структури: назва товару, виробник, місто, кількість одиниць, ціна за одиницю. Сортувати за елементами: назва товару, ціна за одиницю. Застосувати метод сортування: обмінне сортування простою вибіркою

з пошуком мінімуму для впорядкування елементів масиву за спаданням.

10. Елементи структури: викладач, посада, науковий ступінь (кандидат…), вік, педагогічний стаж. Сортувати за елементами: викладач, педагогічний стаж Застосувати метод сортування: метод бульбашки з індикатором перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину в кінці масиву.

11. Елементи структури: рік, пора року, характеристика погоди (одним чи кількома словами), кількість опадів за сезон, середня температура. Сортувати за елементами: пора року, середня температура. Застосувати метод сортування: сортування простими вставками для впорядкування елементів масиву за спаданням, розміщую чи вихідну множину в кінці масиву.

12. Елементи структури: назва підручника, автор, видавництво, рік видання, кількість сторінок. Сортувати за елементами: назва підручника, рік. Застосувати метод сортування: обмінне сортування простою вибіркою з пошуком максимуму для впорядкування елементів масиву за спаданням.

13. Елементи структури: предмет, лектор, асистент, кількість лекцій, кількість практичних. Сортувати за елементами: предмет, кількість лекцій Застосувати метод сортування: метод бульбашки з індикатором перестановки і запам’ятовуванням позиції останньої перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину в кінці масиву.

14. Елементи структури: рослина, висота, тип квітки, колір квітки, кількість квіток у суцвітті. Сортувати за елементами: рослина, кількість квіток у суцвітті. Застосувати метод сортування: шейкер-сортування з індикатором перестановки для впорядкування елементів масиву за спаданням.

15. Елементи структури: прізвище, ім’я, по батькові, курс, група, середній бал. Сортувати за елементами: прізвище, курс Застосувати метод сортування: обмінне сортування простою вибіркою з пошуком мінімуму для впорядкування елементів масиву за зростанням.

16. Елементи структури: прізвище, ім’я, по батькові, вік, курс, група, стипендія. Сортувати за елементами: прізвище, вік. Застосувати метод сортуван-

ня: метод бульбашки з індикатором перестановки для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину на початку масиву.

17. Елементи структури: прізвище й ініціали, світа, професія, вік. Сортувати за елементами: прізвище, кількість наукових праць. Застосувати метод сортування: сортування простими вставками для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

18. Елементи структури: назва книги, автор, видавництво, рік видання, кількість сторінок. Сортувати за елементами: автор, рік видання. Застосувати метод сортування: метод бульбашки з запам'ятовуванням позиції останньої перестановки й індикатором перестановки для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину на початку масиву.

19. Елементи структури: предмет, лектор, викладач на практичних заняттях, кількість лекцій, кількість практичних. Сортувати за елементами: викладач на практичних заняттях, кількість практичних. Застосувати метод сортування: шейкер-сортування для впорядкування елементів масиву за зростанням.

20. Елементи структури: викладач, посада, наукове звання (доцент, професор…), вік, педагогічний стаж. Сортувати за елементами: наукове звання, педагогічний стаж. Застосувати метод сортування: обмінне сортування простою вибіркою з пошуком мінімуму для впорядкування елементів масиву за спаданням.

21. Елементи структури: назва тістечок, виробник, дата виготовлення, ціна одиниці. Сортувати за елементами: назва тістечок, ціна одиниці. Застосувати метод сортування: метод бульбашки з індикатором перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

22. Елементи структури: вид транспорту, прізвище й ініціали водія, номер атп, номер маршруту, кількість місць. Сортувати за елементами: вид транспорту, кількість місць. Застосувати метод сортування: сортування простими вставками для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину в кінці масиву.

23. Елементи структури: назва цукерок, виробник, сорт, , ціна за кілограм. Сортувати за елементами: назва цукерок, маса упаковки. Застосувати метод сортування: метод бульбашки з запам’ятовуванням позиції останньої перестановки й індикатором перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

24. Елементи структури: вид тканини, відтінок основного кольору (яскраво-червоний, жовтогарячий …), ширина, кількість метрів, ціна за метр. Сортувати за елементами: вид тканини, ціна за метр. Застосувати метод сортування: шейкер-сортування з індикатором перестановки для впорядкування елементів масиву за зростанням.

25. Елементи структури: вакансія, освіта, оплата, кількість робочих годин на тиждень. Сортувати за елементами: вакансія, кількість робочих годин на тиждень. Застосувати метод сортування: метод бульбашки з індикатором перестановки і запам’ятовуванням позиції останньої перестановки для впорядкування елементів масиву за спаданням, розміщуючи вихідну множину на початку масиву.

26. Елементи структури: викладач, посада, наукове звання (доцент, професор…), вік, кількість наукових праць. Сортувати за елементами: посада, кількість наукових праць. Застосувати метод сортування: обмінне сортування простою вибіркою з пошуком мінімуму для впорядкування елементів масиву за спаданням.

27. Елементи структури: вид приміщення, кількість кімнат, площа, рік введення в експлуатацію. Сортувати за елементами: вид приміщення, площа. Застосувати метод сортування: метод бульбашки з індикатором перестановки для впорядкування елементів масиву за зростанням, розміщуючи вихідну множину на початку масиву.

28. Елементи структури: птах, вид, рід, маса. Сортувати за елементами: птах, маса. Застосувати метод сортування: обмінне сортування простою вибіркою з пошуком максимуму для впорядкування елементів масиву за зростанням.

29. Елементи структури: предмет, викладач, асистент, семестр, кількість годин. Сортувати за елементами: предмет, кількість годин. Застосувати метод сортування: шейкер-сортування для впорядкування елементів масиву за зростанням.

30. Елементи структури: товар, виробник, кількість одиниць товару, вартість одиниці товару, маса одиниці товару. Сортувати за елементами: товар, кількість одиниць товару. Застосувати метод сортування: обмінне сортування простою вибіркою з пошуком максимуму для впорядкування елементів масиву за спаданням.

Практичне заняття № 25 (*комп'ютерний практикум № 17*)
Виконання додаткового завдання

Мета заняття: продемонструвати вміння самостійно працювати і застосовувати набуті протягом семестру знання при розв'язуванні практичних задач.

Завдання. Додаткове завдання кожен студент виконує за бажанням і завдання вибирає самостійно із поданого нижче списку. На вибір для програмування пропонуються методи й алгоритми дискретної математики, яку студенти вивчають паралельно з предметом "Алгоритмізація та програмування".

1. Виконання теоретико-множинних операцій над множинами (числовими; нечисловими).

2. Побудова замикань відношень відносно рефлексивності, симетричності й транзитивності (метод Уоршалла).

3. Обчислення значень виразів алгебри логіки на основі таблиць істинності логічних функцій від двох аргументів.

4. Побудова ДДНФ і ДКНФ за вектором значень логічної функції.

5. Побудова полінома Жегалкіна методом переходу від вектора значень функції до вектора коефіцієнтів полінома.

6. Побудова полінома Жегалкіна методом невизначених коефіцієнтів.

7. Метод Петрика для визначення повноти системи логічних функцій (за теоремою Поста).

8. Побудова скороченої ДНФ за вектором значень функції методом Квайна.

9. Метод Петрика для побудови тупикових ДНФ за імплікантною таблицею Квайна.

10. Алгоритм визначення дводольності графа.

11. Алгоритм побудови ейлерового циклу в графі.

12. Знаходження діаметра, радіуса і центра графа (матричний метод).

13. Побудова каркасного дерева графа методом пошуку вглиб.

14. Побудова каркасного дерева графа методом пошуку вшир.

15. Метод Краскала побудови мінімального каркасу зваженого графа.

16. Алгоритм Дейкстри знаходження найкоротшого шляху (ланцюга) у зваженому графі.

17. Розфарбовування графів.

18. Алгоримт Дейкстри побудови зворотного польського запису (арифметичні вирази).

Треба написати програму, яка, крім реалізації вибраного алгоритму, також має функції введення даних для обробки і виведення результатів обчислень у зручному для сприйняття вигляді. Передбачити можливість багаторазового виконання програми. Глобальні змінні необґрунтовано не використовувати. Коментарі в програмі обов’язкові (17-25%).

Вибране завдання кожен студент виконує протягом семестру, а на практичному занятті пояснює і обґрунтовує структуру розробленої програми, вказує на нюанси і труднощі, які виникли при реалізації алгоритму, демонструє роботу програми.

Практичне заняття № 26

Робота з динамічною пам'яттю

Мета заняття: закріпити лекційний матеріал і набути основних практичних навичок роботи з динамічною пам'яттю.

1. Динамічне виділення пам'яті під масиви. При динамічному виділенні пам'ять резервується не на етапі компіляції, а на етапі виконання програми, що дає можливість ефективніше її використовувати. Для динамічного виділення пам'яті використовують функцію malloc (параметром функції є обсяг пам'яті; якщо пам'яті не вистачає, то функція повертає значення 0; операційна система записує службову інформацію перед коміркою, на яку вказує повернутий функцією вказівник) або функцію calloc (працює так само, як і функція malloc, але має два параметри: кількість елементів і розмір одного елемета; також вона обнуляє пам'ять), а для звільнення виділеної пам'яті — функцію free (параметром є вказівник на початок виділеної ділянки). Функція realloc змінює обсяг виділеної функцією malloc, calloc або realloc пам'яті (параметрами є вказівник на початок виділеної ділянки і новий обсяг пам'яті; якщо вказаний в параметрі обсяг пам'яті більший від раніше виділеного і поряд недостатньо пам'яті, щоб утворити неперервну ділянку, то виділяється нова ділянка і туди копіюються існуючі дані).

Приклад 1.1. Знайти кут між двома векторами (використати формулу $\cos\alpha = \frac{(\vec{a},\vec{b})}{|\vec{a}|\cdot|\vec{b}|}$; перехід від радіанів до градусів здійснити за формулою $\varphi_{\circ} = \varphi_r \cdot 180/\pi$). Пам'ять під масиви виділяти динамічно. Передбачити можливість проведення багатократних обчислень.

```
#include <stdio.h>
#include <stdlib.h>
#define _USE_MATH_DEFINES
#include <math.h>
void vvid(int *a, int n);
int dobutok(int *a, int *b, int n);
```

```
int main() {
/* робота з динамічним одновимірним масивом */
  int *a,*b;
  int ind;   // індикатор виконання програми
  char vidp;   // відповідь щодо продовження роботи
  int n;   // реальна довжина вектора
  double alfa;   // значення кута між векторами
  system("chcp 1251 & cls");
  ind=1;
  while (ind) {   // багаторазове виконання обчислень
    printf("Вкажіть розмірність векторів:  ");
    scanf("%d",&n);
    /* Виділення пам'яті під масиви */
    a=(int *)malloc(n*sizeof(int));
    b=(int *)malloc(n*sizeof(int));
    if (!a || !b)
      printf("  Недостатньо пам'яті");
    else {
      printf("Введіть координати вектора a:  ");
      vvid(a,n);
      printf("Введіть координати вектора b:  ");
      vvid(b,n);
      /* обчислення кута */
      alfa=acos(dobutok(a,b,n)/sqrt((double)dobutok(a,a,n)
      *dobutok(b,b,n)))/M_PI*180.;
      printf("  Кут між векторами %0.2f°",alfa);
    }
    /* звільнення місця в пам'яті */
    free(a);
    free(b);
    /* запит про продовження роботи */
    printf("\n\nПродовжувати роботу (Y - так)? ");
    fseek(stdin,0,SEEK_END);
    vidp=getchar();
    fseek(stdin,0,SEEK_END);
```

```
    if (!(vidp=='Y'||vidp=='y'||vidp=='T'||vidp=='т'))
      ind=0;
  }
  printf("\n\n");
  system("pause");
  return 0;
}
/* Ввід координат вектора */
void vvid(int *a, int n) {
  int i;
  for (i=0; i<n; i++)
    scanf("%d",&a[i]);
}
/* Знаходження скалярного добутку двох векторів */
int dobutok(int *a, int *b, int n) {
  int i;
  int s=0;
  for (i=0; i<n; i++)
    s+=a[i]*b[i];
  return s;
}
```

```
Вкажіть розмірність векторів:  5
Введіть координати вектора a:  0 1 0 0 0
Введіть координати вектора b:  0 0 0 0 1
  Кут між векторами 90.00°

Продовжувати роботу (Y - так)? так
Вкажіть розмірність векторів:  3
Введіть координати вектора a:  1 2 3
Введіть координати вектора b:  2 4 6
  Кут між векторами 0.00°

Продовжувати роботу (Y - так)? yes
Вкажіть розмірність векторів:  4
Введіть координати вектора a:  1 2 3 4
Введіть координати вектора b:  5 6 7 8
  Кут між векторами 14.34°

Продовжувати роботу (Y - так)? Y
Вкажіть розмірність векторів:  500000000
  Недостатньо пам'яті

Продовжувати роботу (Y - так)? ні
```

Аналогічний результат буде одержано, якщо замість функції malloc використати функцію calloc, записавши при цьому замість операторів:

```
a=(int *)malloc(n*sizeof(int));
b=(int *)malloc(n*sizeof(int));
```

оператори:

```
a=(int *)calloc(n,sizeof(int));
b=(int *)calloc(n,sizeof(int));
```

Приклад 1.2. Динамічно виділяючи пам’ять, утворювати матриці розмірів *n* на *m*, заповнювати їх посідовними числами і виводити на екран. Передбачити можливість багаторазового виконання програми.

```
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
int main() {
  /* робота з динамічними двовимірними масивами */
  int **a;   // a - вказівник на масив вказівників
  int n, m;   // розміри матриці
  int i, j, k;  // параметри циклу і номер елемента
  int ind;   // індикатор виконання програми
  char vidp;   // відповідь щодо продовження роботи
  system("chcp 1251 & cls");
  ind=1;
  while (ind) {   // багаторазове виконання програми
    printf("Вкажіть кількість рядків матриці:  ");
    scanf("%d",&n);
    fseek(stdin,0,SEEK_END);
    printf("Вкажіть кількість стовпчиків матриці:  ");
    scanf("%d",&m);
    fseek(stdin,0,SEEK_END);
    /* Виділення пам'яті під двовимірний масив - матрицю
    з n рядками по m елементів */
    a=(int **)malloc(n*sizeof(int *));   // Виділення пам’яті
         // для масиву вказівників на одновимірні підмасиви
    if (!a)
      printf("  Недостатньо пам'яті для масиву вказівників\n");
```

```
    else {
      for (i = 0; i < n; i++)
        a[i] = (int *)malloc(m*sizeof(int));   // Виділення
                     // пам’яті для одновимірного підмасиву
      if (!a[n-1])
        printf("  Недостатньо пам'яті для підмасивів\n");
      else {
        k=0;
        /* заповнення масиву послідовністю чисел */
        for (i = 0; i < n; i++)
          for (j = 0; j < m; j++)
            a[i][j]=k++;
        /* виведення масиву */
        for (i = 0; i < n; i++) {
          for (j = 0; j < m; j++)
            printf("%3d", a[i][j]);
          printf("\n");
        }
      }
      /* Звільнення пам'яті двовимірного масиву */
      for(i = 0; i < n; i++)
        free(a[i]); // звільнення пам’яті одновимірного підмасиву
      free(a);   // звільнення пам’яті масиву вказівників на
                 // одновимірні підмасиви
    }
    /* запит про продовження роботи */
    printf("\nПродовжувати роботу (Y - так)? ");
    fseek(stdin,0,SEEK_END);
    vidp=getchar();
    fseek(stdin,0,SEEK_END);
    if (!(vidp=='Y'||vidp=='y'||vidp=='T'||vidp=='т'))
      ind=0;
  }
  printf("\n\n");
  system("pause");
  return 0;
}
```

```
Вкажіть кількість рядків матриці:  500000000
Вкажіть кількість стовпчиків матриці:  10
  Недостатньо пам'яті для масиву вказівників

Продовжувати роботу (Y - так)? т
Вкажіть кількість рядків матриці:  10
Вкажіть кількість стовпчиків матриці:  50000000
  Недостатньо пам'яті для підмасивів

Продовжувати роботу (Y - так)? т
Вкажіть кількість рядків матриці:  1
Вкажіть кількість стовпчиків матриці:  8
  0  1  2  3  4  5  6  7

Продовжувати роботу (Y - так)? т
Вкажіть кількість рядків матриці:  3
Вкажіть кількість стовпчиків матриці:  4
  0  1  2  3
  4  5  6  7
  8  9 10 11

Продовжувати роботу (Y - так)? т
Вкажіть кількість рядків матриці:  7
Вкажіть кількість стовпчиків матриці:  10
  0  1  2  3  4  5  6  7  8  9
 10 11 12 13 14 15 16 17 18 19
 20 21 22 23 24 25 26 27 28 29
 30 31 32 33 34 35 36 37 38 39
 40 41 42 43 44 45 46 47 48 49
 50 51 52 53 54 55 56 57 58 59
 60 61 62 63 64 65 66 67 68 69

Продовжувати роботу (Y - так)? ні


Press any key to continue . . .
```

Якщо для заповнення масиву числами і виведення масиву написати і використати функції zapovn і vyvid, то в програму треба внести такі зміни:

```
...
void zapovn(int **a, int n, int m);
void vyvid(int **a, int n, int m);
int main() {
...
  /* заповнення масиву послідовністю чисел */
  zapovn(a,n,m);
  /* виведення масиву */
  vyvid(a,n,m);
...
}
```

```
void zapovn(int **a, int n, int m) {
  int i, j, k;
  k=0;
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      a[i][j]=k++;
}
void vyvid(int **a, int n, int m) {
  int i, j;
  for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++)
      printf("%3d", a[i][j]);
    printf("\n");
  }
}
```

Якщо в функції заповнення масиву дані вводити з клавіатури, то треба використати оператор `scanf("%d",&a[i][j]);`.

Завдання. Написати порограму модульної структури для множення матриць $C_{m\times n} = A_{m\times k} \cdot B_{k\times n}$ (значення *m*, *n* і *k* задавати в діалоговому режимі), використовуючи динамічне виділення пам'яті. Передбачити можливість багаторазового виконання алгоритму програми.

2. Робота зі списками. Список — це структура даних, у якій елементи (вузли) лінійно впорядковані, але порядок визначається не номерами елементів, а вказівниками, які входять до складу елементів списку і вказують в однобічно зв'язаних списках на наступний за даним елемент, а у двобічно зв'язаних списках на наступний і попередній елементи. Перший елемент списку називають головою, а останній — хвостом. Вказівник останнього елемента є нульовим (NULL). Для однотипності роботи (вставка чи вилучення елемента) з усіма елементами списку (як правило, перший і останній елементи обробляють за спеціальними алгоритмами) можна використовувати фіктивні елементи, між якими розміщуються всі робочі елементи. Часом утворюють циклічний список, коли вказівник останнього елемента вказує

на перший елемент (у випадках однобічно й двобічно зв'язаних списків), а першого (у випадку двобічно зв'язаного списку) — на останній.

Приклад 2.1. За введеним текстом, використовуючи однобічно зв'язаний список, створити словник, який містить впорядковані за алфавітом українські слова і їхні частоти (числа, розділові знаки і слова, які починаються з латинських букв, у словник не вносити). Для спрощення обробки використати фіктивні перший і останній елементи. Зауваження: оскільки в кодовій таблиці ANSI (також і ASCII) символи і, ї, є, ґ розміщуються поза алфавітом, то для української мови при застосуванні функції strcmp для порівняння рядків символів буде порушено порядок (врахування такого розміщення при впорядкуванні — окрема задача і для повного впорядкування можна написати свою функцію).

```
/* Робота з однобічно зв'язаним списком */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>   // функції strcpy, strcmp
#define N 15
struct slova{   // елемент списку слів (словника)
    char si[N];   // окреме слово
    int ni;   // частота слова
    struct slova *p;};   // вказівник на наступний елемент
struct slova * kintsevi();
void vstavka(char slovo[N], struct slova *p);
void vyvids(struct slova *pochatok);
void zvilnennia(struct slova *pochatok);
int main() {
/* На основі введеного тексту створюється впорядкована за
   алфавітом послідовність слів (словник) з їхніми частотами */
  struct slova *pochatok, *r;  // дані про слова словника
  char slovo[N]="";
  system("chcp 1251 & cls");   // кодування ANSI
  /* ініціалізація фіктивних голови і хвоста списку слів */
  pochatok=kintsevi();
```

```
  /* зчитування слів введеного тексту (без аналізу) і запис у список */
  printf("Введіть текст до символів - пробіл#:\n\n");
  for (; scanf("%15s",slovo), slovo[0]!='#'; ) {
    r=pochatok;
    vstavka(slovo, r);
  }
  /* Вивід списку-словника */
  vyvids(pochatok);
  /* звільнення пам'яті, виділеної під список */
  zvilnennia(pochatok);
  printf("\n\n");
  system("pause");
  return 0;
}
/* Ініціалізація фіктивних початку (голови) і кінця (хвоста)
   списку слів */
struct slova *kintsevi() {
  struct slova *pochatok, *kinets;
  pochatok=(struct slova *) malloc(sizeof(struct slova));
  kinets=(struct slova *) malloc(sizeof(struct slova));
  strcpy((*pochatok).si, "Ґ");   // в ANSI код Ґ - Alt+0165
  (*pochatok).ni=0;
  (*pochatok).p=kinets;
  strcpy((*kinets).si, "яяя");
  (*kinets).ni=0;
  (*kinets).p=NULL;
  return(pochatok);
}
/* Вставка слів у список-словник і обчислення їхніх частот.
   Зчитане "слово", яке починається не з букви українського
   алфавіту,в словник не додається */
void vstavka(char slovo[N], struct slova *r) {
  struct slova *poperednie;   // вказівник на попередній елемент
  struct slova *nove;   // вказівник на новий елемент
  int ind;   // -1 - зчитане слово менше, 0 - рівне,
             // 1 - більше від слова у списку
```

```
  int nom;   // номер елемента списку
  nom=1;   // вибрано початковий елемент списку
  /* перегляд елементів списку і порівняння з введеним словом */
  do {
    ind=strcmp(slovo,(*r).si);
    if (!ind)   // таке слово є в списку
      (*r).ni=(*r).ni+1;   // збільшення частоти слова
                           // на 1 і вихід з циклу
    else
      if (ind==-1) {   // вставка елемента списку і вихід з циклу
        if (nom!=1) {   // слово не перед початком списку
          nove=(struct slova *) malloc(sizeof(struct slova));
          (*nove).p=(*poperednie).p;  // новий елемент
                                      // вказує на наступний
          (*poperednie).p=nove;   // попередній елемент
                                  // вказує на новий
          strcpy((*nove).si,slovo);   // запис слова в
                                      // новий елемент
          (*nove).ni=1;   // записане слово має частоту 1
        }
      }
      else  {  // вибрати наступний елемент списку
        nom++;
        if ((*r).p!=NULL) {   // не останній елемент списку
          poperednie=r;
          r=(*r).p;
        }
        else ind=-1;   // дійшли до кінця списку -
                       // треба виходити з циклу
      }
  }   while (ind==1);
}
/* Виведення всіх елементів списку */
void vyvids(struct slova *poperednie) {
  struct slova *r;   // вказівник на поточний елемент
  printf("\nСлово\t\t частота\n");
  r=poperednie;   // встановлення вказівника на початок списку
```

```
  do {
    printf(" %-15s%3d\n", (*r).si, (*r).ni);
    r=(*r).p;}   // перехід до наступного елемента списку
  while (r!=NULL);
}
/* звільнення пам'яті, зайнятої списком */
void zvilnennia(struct slova *pochatok) {
  struct slova *r, *nastupne;   // вказівники на поточний
                                // і наступний елементи
  r=pochatok;   // встановлення вказівника на початок списку
  do {
    nastupne=(*r).p;   // наступний елемент списку
    free(r);   // звільнення пам'яті поточного елемента
    r=nastupne;   // вказівник наступного елемента стає поточним
  } while (r!=NULL);
}
```

```
Введіть текст до символів - пробіл#:

123 Ґ я 45 АБВ я я Ґ 678
слава не вмре , не поляже .
буде слава славна поміж козаками ,
поміж друзями , поміж лицарями (Дума) #

Слово            частота
 Ґ                2
 АБВ              1
 буде             1
 вмре             1
 друзями          1
 козаками         1
 лицарями         1
 не               2
 поляже           1
 поміж            3
 слава            2
 славна           1
 я                3
 яяя              0
```

Завдання. Написати програми побудови каркасних дерев зв’язного неорієнтовагого графа, використовуючи методи обходу вглиб і вшир. Для роботи зі стеком і чергою використати динамічне виділення пам’яті.

Практичне заняття № 27

Диференційований залік

Мета заняття: виконання залікової контрольної роботи для покращення сумарного залікового балу. Залік проводиться у письмовому вигляді. Залікова контрольна робота містить 4 завдання (2 теоретичні і 2 практичні), які забезпечують перевірку результатів навчання. Завдання передбачають інтегроване застосування знань матеріалу навчальної дисципліни.

Теоретичні питання охоплюють весь матеріал курсу. При написанні відповіді на питання треба чітко й лаконічно розкрити суть питання, і, якщо можливо, подати приклад.

Перелік теоретичних питань, які виносяться на залікову роботу:

1. Системи числення. Двійкове, вісімкове, шістнадцяткове подання цілих і дробових чисел.
2. Структура програми мовою С: декларативна й виконувана частини.
3. Лексичні одиниці мови С: ключові слова, ідентифікатори, константи.
4. Типи даних в мові С. Перетворення типів у виразаї і при присвоюванні.
5. Введення й виведення даних (функції printf, scanf). Формати.
6. Запис цілих, дійсних, логічних, символьних констант мови С.
7. Оголошення й використання констант у мові С.
8. Змінні стандартних типів мови С: цілі, дійсні, символьні.
9. Операції мови С. Операція присвоювання.
10. Операції мови С. Операція кома.
11. Арифметичні вирази.
12. Операції порівняння. Логічні вирази.
13. Умовний оператор мови С. Вкладені умовні оператори.
14. Тернарна операція ?:
15. Оператор вибору мови С.
16. Поняття циклу. Оператори циклу мови С. Вкладені цикли.
17. Оператор циклу з параметром. Схема виконання.

18. Нестандартні можливості використання оператора циклу з параметром.

19. Оператор циклу з передумовою. Схема виконання.

20. Оператор циклу з післяумовою. Схема виконання.

21. Функції мови С: стандартні й описувані.

22. Функції мови С: з параметрами і без параметрів.

23. Функції, які повертають значення. Оператор return.

24. Локальні, глобальні й статичні змінні.

25. Фактичні й формальні параметри.

26. Механізми передачі параметрів за адресою і за значенням.

27. Рекурсивні функції.

28. Одновимірні масиви мови С: опис, ініціалізація, введення-виведення.

29. Двовимірні масиви мови С: опис, ініціалізація, введення-виведення.

30. Передача параметрів-масивів у функції.

31. Робота з розрідженими матрицями.

32. Змінні символьного типу і рядки символів.

33. Основні функції роботи з рядками символів.

34. Перераховувані дані: опис, ініціалізація, використання.

35. Побітові операції.

36. Структури: опис, робота, передача в функції.

37. Масиви структур: опис, робота, передача в функції.

38. Динамічна пам’ять. Динамічні одновимірні й двовимірні масиви.

39. Основні функції роботи з файлами.

40. Методи сортування елементів масиву: обмінне сортування простою вибіркою з пошуком мінімуму чи максимуму.

41. Методи сортування елементів масиву: обмінне сортування (метод бульбашки) з індикатором перестановки.

42. Методи сортування елементів масиву: шейкер-сортування.

43. Методи сортування елементів масиву: сортування простими вставками.

44. Поняття алгоритму. Властивості алгоритму.

45. Подання алгоритмів. Словесне подання алгоритму (метамова). Подання алгоритму за допомогою блок-схеми.

46. Структура й принципи роботи ПЕОМ.

47. Подання даних у комп'ютері: цілі й дійсні числа, звук, графіка.

48. Типи обчислювальних структур: лінійні, розгалужені, циклічні.

49. Основні поняття програмування. Синтаксис, синтаксичні помилки. Семантика, семантичні помилки. Налагодження й тестування програм.

50. Методи трансляції програм: інтерпретація, компіляція.

51. Основні положення структурного програмування.

52. Модульне програмування. Переваги й недоліки.

53. Проектування й програмування згори вниз і знизу вгору.

54. Структурне програмування і коректність програм. Наскрізний структурний контроль.

Практичні завдання. При виконанні практичних завдань треба написати програми з коментарями.

Перше практичне завдання пов'язане з обробкою одновимірних масивів (виконання перестановок елементів у масиві), друге — з обробкою двовимірних масивів (виконання обчислень з елементами сектора матриці).

Зразки першого практичного завдання. Нехай одновимірний масив містить *n* елементів, де *n* може бути як парним, так і непарним числом. У масиві:

— елементи з певною ознакою зібрати на початку / в кінці масиву, зберігши їхній порядок, решту елементів розмістити в кінці / на початку масиву, замінивши нулями;

— переставити місцями певні частини масиву, зберігши в них порядок елементів;

— переставити місцями кожні два сусідні елементи (1 і 2, 3 і 4 і т.д.);

— елементи з парними / непарними номерами / усього / першої половини / другої половини масиву розмістити у зворотному порядку тощо.

Усі дії виконувати в тому самому масиві. Алгоритм оформити у вигляді функції. Глобальні змінні не використовувати, оброблений масив повернути через параметри. Подати блок-схему алгоритму.

Зразки другого практичного завдання. Нехай квадратна матриця має порядок *n*, де *n* може бути як парним, так і непарним числом. У частині X матриці знайти елемент з певною ознакою / порахувати кількість / суму / добуток тощо елементів з певною ознакою. Алгоритм оформити у вигляді функції, яка повертає значення. Глобальні змінні не використовувати.

Додаток А

Транслітерація. Нормативна таблиця для відтворення українських власних назв засобами англійської мови

N	Укр. літера	Лат. літера	Примітки	Приклади застосування
1	А	A	—	Алушта — Alushta
2	Б	B	—	Борщагівка — Borschahivka
3	В	V	—	Вишгород —Vyshhorod
4	Г	H, gh	H — у більшості випадків; gh — при відтворенні сполуки -зг-	Гадяч — Hadiach; Згорани — Zghorany
5	Ґ	G	—	Ґалаґан — Galagan
6	Д	D	—	Дон — Don
7	Е	E	—	Рівне — Rivne
8	Є	Ye, ie	Ye — на початку слова; ie — в інших позиціях	Єнакієве — Yenakiieve; Нуєнко — Nuienko
9	Ж	Zh	—	Житомир — Zhytomyr
10	З	Z	—	Закарпаття — Zakarpattia
11	И	Y	—	Медвин — Medvyn
12	І	I	—	Іршава — Irshava
13	Ї	Yi, i	Yi — на початку слова; i — в інших позиціях	Їжакевич — Yizhakevych; Кадіївка — Kadiivka
14	Й	Y, i	Y — на початку слова; i — в інших позиціях	Йосипівка — Yosypivka; Стрий — Stryi
15	К	K	—	Київ — Kyiv
16	Л	L	—	Лебедин — Lebedyn
17	М	M	—	Миколаїв — Mykolaiv
18	Н	N	—	Ніжин — Nizhyn
19	О	O	—	Одеса — Odesa
20	П	P	—	Полтава — Poltava
21	Р	R	—	Ромни — Romny
22	С	S	—	Суми — Sumy
23	Т	T	—	Тетерів — Teteriv
24	У	U	—	Ужгород — Uzhhorod
25	Ф	F	—	Фастів — Fastiv
26	Х	Kh	—	Харків — Kharkiv

27	Ц	Ts	—	Біла Церква — Bila Tserkva
28	Ч	Ch	—	Чернівці — Chernivtsi
29	Ш	Sh	—	Шостка — Shostka
30	Щ	Shch	—	Гоща — Hoshcha
31	Ь	'	(див. коментар)	Русь — Rus'; Львів — L'viv
32	Ю	Yu, iu	Yu — на початку слова; iu — в інших позиціях	Юрій — Yurii; Крюківка — Kriukivka
33	Я	Ya, ia	Ya — на початку слова; ia — в інших випадках	Яготин — Yahotyn; Ічня — Ichnia
'		(апостроф)	(див. коментар)	Знам’янка — Znamianka

1. Відтворення українських власних назв засобами англійської мови відбувається з їх української форми, записаної відповідно до чинного правопису, без посередництва будь-якої іншої мови.

2. Відтворення українських власних назв засобами англійської мови відбувається шляхом транслітерації (політерного перезапису за допомогою латинського алфавіту).

…

Коментар до нормативної таблиці. У певних сферах відтворення українських власних назв вживається спрощений варіант запису, що передбачає:

а) орфографічне спрощення громіздкого подвоєння приголосних ж, х, ц, ч, ш, які відтворюються буквосполученнями zh, kh, ts, ch, sh, наприклад: Запоріжжя — Zaporizhia;

б) апостроф і знак м’якшення (за винятком буквосполучень -ьо-, -ьї-, що завжди передаються як ’о, ’і) у спрощеній транслітерації не відтворюються, наприклад:

Українська форма	**Спрощена транслітерація**	**Точна транслітерація**
Львів	Lviv	L'viv
Ананьїв	Ananiv	Anan'iv
Стеф'юк	Stefiuk	Stef'iuk
Короп'є	Koropie	Korop'ie

(*Рішення N 9 Української комісії з питань правничої термінології, протокол N 2 від 19 квітня 1996 р.* [*])

[*] Такі самі норми встановлено *Постановою Кабінету Міністрів України від 27 січня 2010 р. № 55 Про впорядкування транслітерації українського алфавіту латиницею*

Додаток Б

Зразок оформлення звіту з робіт комп'ютерного практикуму

Кублій Лариса Іванівна
кімната 509-5, варіант № 0

Комп'ютерний практикум № 2. Оформлення звітів
засобами текстового редактора Word

Варіант № 0

Мета роботи: навчитися оформляти звіти з робіт комп'ютерного практикуму засобами текстового редактора Word.

Завдання: Набрати текст поданої програми, транслітеруючи коментарі символами англійського алфавіту (дотримуватися правил транслітерації). Виконати цю програму з кількома різними значеннями кодів символів.

Оформити звіт з роботи комп'ютерного практикуму: вказати її мету, завдання, подати текст програми (шрифт Courier New чи Consolas, напівжирне накреслення; структуризація тексту програми — 2 пробіли), намалювати блок-схему алгоритму програми, подати не менше 3 зображень результату роботи, сформулювати висновки, у верхніх колонтитулах подати номер роботи, її назву, своє прізвище, номер своєї групи і номер варіанту, в нижні колонтитули вставити нумерацію сторінок.

Текст програми:

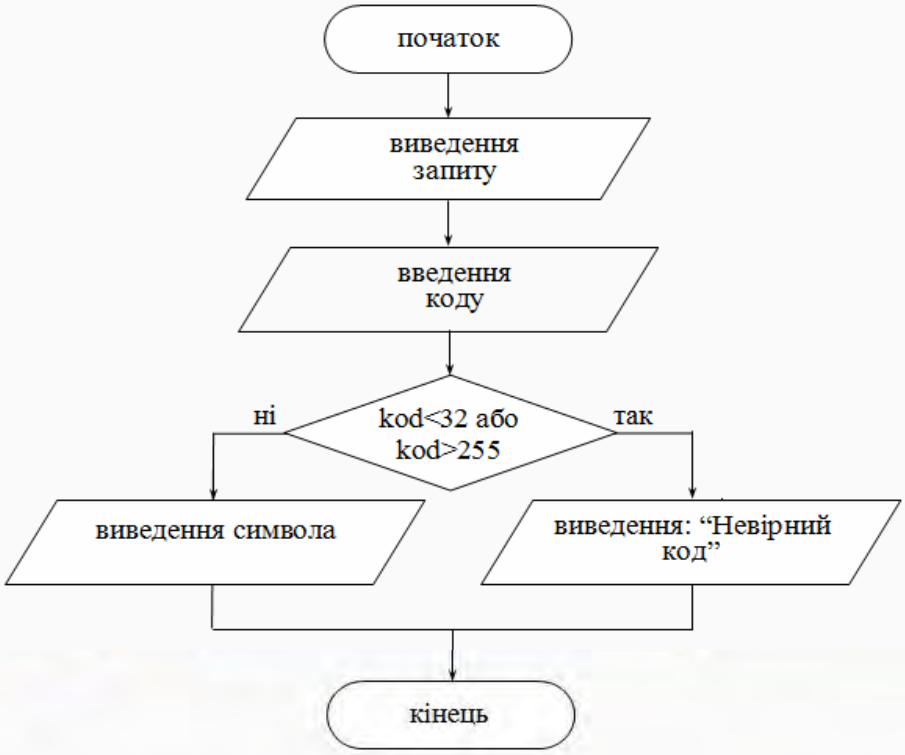
```
/* Prizvyshche Imia Po batkovi, hrupa TX-XX, variant № X
 * Rozrakhunkova robota № 2. Oformlennia zvitiv z rozrakhunkovykh
 * robit zasobamy tekstovoho redaktora Word */
#include <stdio.h>   /* pidmykannia biblioteky stdio.h; u prohrami
     vykorystovuiutsia funktsii printf, scanf, fflush, getchar */
#include <stdlib.h>   /* pidmykannia stdlib.h; u prohrami
     vykorystovuietsia funktsiia system */
/* Za vvedenym korystuvachem chyslovym kodom vid 32 do 255
     (koduvannia ASCII) vyvodyt symvol */
int main()
{
  int kod;   /* kod symvola */
  /* symvoly z kodamy 0-31 ye sluzhbovymy; probil maie kod 32; kod
     13 vvodytsia pry natyskanni na klavishu Enter, 27 — na Esc */
  printf("vvedit ASCII-kod symvola vid 32 do 255\n");
  scanf("%d",&kod);
  if (kod<32 || kod>255) printf("nevirnyi kod\n");
  else printf("symvol   %c\n",/*(char)*/kod);
  system("pause");  /* zatrymka ekrana */
  /* abo dlia zatrymky ekrana vyprobuite shche 2 varianty:
     fflush(stdin); getchar(); — очистка буфера клавіатури (stdin —
       standartnyi potik vvodu) i ochikuvannia Enter; abo
     while (getchar()!='\n'); getchar(); */
  return 0;
}
```

Зразок оформлення звіту з робіт комп'ютерного практикуму (*продовження*)

№ 2. Оформлення звітів
засобами текстового редактора Word

Кублій Лариса Іванівна
кімната 509-5, варіант № 0

Блок-схема алгоритму:



Приклади роботи програми:

```
vvedit ASCII-kod symvola vid 32 do 255
12
nevirnyi kod
Press any key to continue . . .
```

```
vvedit ASCII-kod symvola vid 32 do 255
134
symvol   Ж
Press any key to continue . . .
```

```
vvedit ASCII-kod symvola vid 32 do 255
200
symvol   Ш
Press any key to continue . . .
```

Висновок: у результаті виконання роботи комп'ютерного практикуму на тему "Оформлення звітів засобами текстового редактора Word" я навчилася оформляти звіти засобами редактора Word: форматувати текст звіту, вставляти в звіт текст програми з с-файла, розробляти й малювати блок-схему, дотримуючись ГОСТ 19.701–90, одержувати, вставляти й оформляти скриншоти результатів роботи програми. Також я транслітерувала коментарі символами англійського алфавіту, дотримуючись правил транслітерації.

2

Додаток В

Основні відомості з мови програмування С

1. Структура програми

<table>
<tr><td colspan="3">Директиви препроцесорові</td><td>

```
#include <stdio.h>
#define _USE_MATH_DEFINES
#include <math.h>
#define N 255
```

</td></tr>
<tr><td colspan="3">Прототипи (заголовки) функцій користувача</td><td>

```
int fm(int a, int b);
double fa(double x);
```

</td></tr>
<tr><td colspan="3">Заголовок головної функції (програми)</td><td>

```
int main()
```

</td></tr>
<tr><td colspan="4">{</td></tr>
<tr><td rowspan="2">Тіло функції</td><td>декларативна частина</td><td>визначення констант

опис змінних</td><td>

```
const double A=2.128e-2;
const int B=286;
long int l; unsigned char uc;
int i,j,k; int a=25, h=6;
char str[10];
static int
  b[2][3]={{1,2,3},{4,5,6}};
```

</td></tr>
<tr><td>виконувана частина</td><td>оператори програми, кожен з яких закінчується ;</td><td>оператор присвоювання

```
  a=a+sin(x)-1;
```

оператор прийняття рішення (умовний)

```
  if-else
```

оператор варіанту

```
  switch-case-default
```

оператор циклу

```
  for, while, do-while
```

оператор функції

```
printf("x=%f\n",x);
  scanf("%f",&x);
```

оператори переходу <code>return, goto,
  continue, break</code>
блок операторів <code>{…}</code> ({} — *операторні дужки*)
порожній оператор <code>; {}</code></td></tr>
<tr><td colspan="4">}</td></tr>
<tr><td colspan="3">Описи функцій користувача</td><td>

```
int fm(int a,int b) {
  if (a>b) return a;
  return b;}
double fa(double x);
  {if (x<0) x=-x; return x;}
```

</td></tr>
</table>

2. Ключові слова мови С (стандарт С89, або ANSI C)

auto — автоматичний (тип пам'яті); **break** — завершити; **case** — варіант; **char** — символьна величина; **const** — константа **continue** — закінчити поточну ітерацію циклу; **default** — за замовчуванням (дія при відсутності варіанту вибору); **do** — виконати; **double** — дійсна величина подвійної точності; **else** — інакше; **enum** — перераховуваний тип; **extern** — зовнішній (тип пам'яті);	**float** — дійсна величина з плаваючою точкою; **for** — для; **goto** — перейти; **if** — якщо; **int** — цілочисельна величина; **long** — цілочисельна величина подвійної точності; **register** — регістровий (тип пам'яті); **return** — повернення; **short** — цілочисельна величина скороченої довжини; **signed** — ціле число зі знаком (старший розряд вважається знаком); **sizeof** — визначення розміру операнда в байтах;	**static** — статичний (тип пам'яті); **struct** — величина структурного типу (структура); **switch** — перемикач; **typedef** — визначення типу (визначає скорочене ім'я для позначення типу); **union** — об'єднання; **unsigned** — величина беззнакового типу (старший розряд не вважається знаком); **void** — величина, значення якої відсутнє; **volatile** — об'єкт, значення якого може змінюватися без явних вказівок програміста; **while** — доки.

Ключові (зарезервовані) слова не можна використовувати як ідентифікатори. Але, якщо підімкнуто певну бібліотеку й ім'я змінної збігається з іменем бібліотечної функції, то не можна буде скористатися відповідною функцією підімкнутої бібліотеки.

У стандарті С99 додано ще 5 ключових слів: **inline**, **restrict**, **_Bool**, **_Complex**, **_Imaginary**. Але не у всіх версіях Visual Studio вони реалізовані Наприклад, у середовищі VS 2010 ці слова, крім слова **inline** (вказівка компілятору оптимізувати виклик функції), не є ключовими.

У стандарті С11 додано ще 7 ключових слів: **_Alignas**, **_Alignof**, **_Atomic**, **_Generic**, **_Noreturn**, **_Static_assert**, **_Thread_local**.

3. Типи даних мови С

Інформацію про основні типи даних мови С, обсяги пам'яті, яку вони займають, і діапазони їхніх значень (середовище Microsoft Visual Studio) подано в таблиці:

Тип	Пам'ять (байтів)	Діапазон значень
char / signed char	1	–128 … 127 (-2^7 … 2^7-1)
unsigned char	1	0 … 255 (2^8-1)
short int / signed short int	2	–32768 … 32767 (-2^{15} … $2^{15}-1$)
unsigned short int	2	0 … 65535 ($2^{16}-1$)
int / signed int / long int / signed long int	4	–2147483648…2147483647 (-2^{31}… $2^{31}-1$)
unsigned int / unsigned long int	4	0 ... 4294967295 ($2^{32}-1$)
long long int	8	–9223372036854775808 … 9223372036854775807 (-2^{63} … $2^{63}-1$); *стандарт С99*
unsigned long long int	8	0 ... 18446744073709551615 (0 … $2^{64}-1$); *стандарт С99*
Float	4	9.9e–45 … 3.4e+38; точність — 6-7 десяткових цифр; мантиса — 24 біти
double / long double	8	4.94e–324 ... 1.7e+308; точність — 15-16 десяткових цифр; мантиса — 53 біти

Перетворення типів операндів у виразах виконуються відповідно до таблиці при перегляді її згори вниз:

№	Початкові типи операндів	Типи операндів після перетворення	Пояснення
1.	float α	**double** α	**усі** операнди типу **float** перетворюються на **double**
2.	long double α будь-який тип β	long double α **long double** β	якщо один операнд має тип long double, то і другий операнд перетвориться на long double

3.	double α будь-який тип β	double α **double β**	якщо один операнд має тип double, то і другий операнд перетвориться на double
4.	char α	**int α**	**усі** операнди типу **char** або **short** перетворюються на **int**
5.	short int α	**int α**	
6.	unsigned char α	**unsigned int α**	**усі** операнди типу **unsigned char** або **unsigned short** перетворюються на **unsigned int**
7.	unsigned short int α	**unsigned int α**	
8.	unsigned long int α будь-який тип β	unsigned long int α **unsigned long int β**	якщо один операнд має тип unsigned long, то і другий перетвориться на unsigned long
9.	long int α будь-який тип β	long int α **long int β**	якщо один операнд має тип long, то і другий перетвориться на long
10.	unsigned int α будь-який тип β	unsigned int α **unsigned int α**	якщо один операнд має тип unsigned int, то і другий перетвориться на unsigned int

При виконанні операції присвоювання можлива втрата інформації за рахунок меншого обсягу пам'яті результуючої змінної або іншого формату подання даних в пам'яті комп'ютера :

Тип змінної	**Тип виразу**	**Втрата інформації**
signed char (8 бітів)	char (8 бітів)	якщо значення >127, то результат від'ємний
char (8 бітів)	short int (16 бітів)	старші 8 бітів
char (8 бітів)	int / long int (32 біти)	старші 24 біти
short int (16 бітів)	int (32 біти)	старші 16 бітів
int (32 біти)	long int (32 біти)	нема
long int (32 біти)	long long int (64 біти)	старші 32 біти (стандарт C99)
int	float	дробова частина
float	double	результат заокруглюється
double	long double	результат заокруглюється

Наприклад, якщо в програмі є такі оператори: `int a=283; char c; c=a;`, то змінна c набуде значення 27 (оскільки 283_{10}= 1 0001 1011_2 і старший розряд при присвоєнні втрачається, то залишиться число 0001 1011_2, яке інтерпретується як 27_{10}).

4. Операції мови С

(операції подано за спаданням пріоритетів; у кожній групі, відокремленій суцільною лінією, пріоритети однакові)

Група операцій	Знак операції	Опис
первинні	( )	підвищення пріоритету (дужки)
	[ ]	виділення елемента масиву
	.	виділення елемента запису
	->	виділення елемента запису
унарні	!	логічне заперечення
	~	порозрядне заперечення
	-	зміна знака
	++	збільшення на одиницю (інкремент)
	--	зменшення на одиницю (декремент)
	&	взяття адреси
	*	звернення за адресою
	(тип)	перетворення типу
	sizeof( )	визначення розміру в байтах
мультиплікативні	*	множення
	/	ділення
	%	визначення остачі від ділення
адитивні	+	додавання
	-	віднімання
зрушення	<<	зсув вліво
	>>	зсув вправо
відношень (порівняння)	<	менше ніж
	<=	менше або дорівнює
	>	більше ніж
	>=	більше або дорівнює
	==	дорівнює
	!=	не дорівнює
порозрядні	&	порозрядне логічне “і”
	^	порозрядне виключне “або”
	|	порозрядне логічне “або”
логічні	&&	логічне “і”
	||	логічне “або”
умови	?:	умовна (тернарна) операція
присвоювання	=	присвоювання
	+=, - =, *=, /=, %=, <<=, >>=, &=, |=, ^=	складені операції присвоювання (a*=b означає a=a*b)
кома	,	операція кома

5. Основні математичні функції і константи мови С

Усі функції мови С є бібліотечними. При зверненні до функцій їхні аргументи записують в круглих дужках: `sin(x)`, `abs(x-1)`, `fabs(sin(a))` тощо. Якщо функція не має аргументів, дужки подавати обов'язково: `getchar()`, `rand()`.

Для обчислення значення функції x^y (реалізує функція `pow(x,y)`), якщо $x > 0$, можна використовувати рівність $x^y = e^{y \ln x}$ (`exp(y*log(x))`. При обчисленні значень x^2 і x^3 краще степені подати як `x*x` і `x*x*x`.

Значення $\log_a b$ обчислюється за формулою $\log_a b = \dfrac{\ln b}{\ln a}$.

Значення $\operatorname{ctg}\alpha$ обчислюється за формулою $\operatorname{ctg}\alpha = \dfrac{\cos\alpha}{\sin\alpha}$ або $\operatorname{ctg}\alpha = \dfrac{1}{\operatorname{tg}\alpha}$.

Значення числа $e = 2.718281828459045$ можна одержати, скориставшись функцією e^x: `e=exp(1)`.

Якщо в тексті програми курсор навести на ім'я функції, то відобразяться всі можливі її прототипи.

Якщо в тексті програми в директиві `#include <xxxxx.h>` клацнути правою клавішею мишки по імені бібліотеки і в контекстному меню вибрати команду Open Document `<xxxxx.h>`, то відкриється заголовочний файл бібліотеки, в якому за допомогою пошуку (Ctrl+F) можна знайти потрібну інформацію.

Бібліотеки мови С, їхні функції й константи описано на сайті www.cplusplus.com/reference/

а) бібліотека math.h* (www.cplusplus.com/reference/cmath/ldexp/) — **функції і константи**

Найосновніші функції бібліотеки math.h подано в таблиці:

Ім'я функції**	**Прототип**	**Опис**
sin	double sin(double x);	синус $\sin x$
cos	double cos(double x);	косинус $\cos x$

tan	double tan(double x);	тангенс $\operatorname{tg} x$
asin	double asin(double x);	арксинус $\arcsin x$
acos	double acos(double x);	арккосинус $\arccos x$
atan	double atan(double x);	арктангенс $\operatorname{arctg} x$
sinh	double sinh(double x);	гіперболічний синус $\operatorname{sh} x = (e^x - e^{-x})/2$
cosh	double cosh(double x);	гіперболічний косинус $\operatorname{ch} x = (e^x + e^{-x})/2$
tanh	double tanh(double x);	гіперболічний тангенс $\operatorname{th} x = \operatorname{sh} x / \operatorname{ch} x$
sqrt	double sqrt(double x);	квадратний корінь $\sqrt{x}$
cbrt	double cbrt(double x);	кубічний корінь $\sqrt[3]{x}$; *стандарт С99*
pow(x,y)	double pow(double x, double y); або double pow(double x, int y);	значення x^y
exp	double exp(double x);	показникова функція e^x
log	double log(double x);	натуральний логарифм $\ln x$
log10	double log10(double x);	десятковий логарифм $\log_{10} x$
abs	int abs(double x); або int abs(int x);	ціле абсолютне значення $\lvert x \rvert$
fabs	double fabs(double x);	абсолютне значення $\lvert x \rvert$
round	double round(double x);	заокруглює до найближчого цілого числа (`round(3.1)=3; round(3.5)=4; round(3.8)=4; round(-3.1)= -3; round(-3.5)= -4; round(-3.8)= -4`); *стандарт С99*
trunc	double trunc(double x);	відкидає дробову частину числа (`trunc(3.1)=3; trunc(3.5)=3; trunc(3.8)=3; trunc(-3.1)= -3; trunc(-3.5)= -3; trunc(-3.8)= -3`); *стандарт С99*

ceil	double ceil(double x);	заокруглює до найближчого більшого цілого числа (`ceil(3.1)=4; ceil(3.5)=4; ceil(3.8)=4; ceil(-3.1)= -3; ceil(-3.5)= -3; ceil(-3.8)= -3`)
floor	double floor(double x);	заокруглює до найближчого меншого цілого числа (`floor(3.1)=3; floor(3.5)=3; floor(3.8)=3; floor(-3.1)= -4; floor(-3.5)= -4; floor(-3.8)= -4`)
fmod	double fmod(double x, double y);	залишок від ділення двох чисел x/y (`fmod(3.1, 3.8)=3.1; fmod(3.1, 1.4)=0.3; fmod(-3.1, 1.4)= -0.3`)
fmax	double fmax(double x, double y);	максимальне значення $\max(x, y)$; *стандарт С99*
fmin	double fmin(double x, double y);	мінімальне значення $\min(x, y)$; *стандарт С99*

*Бібліотека <tgmath.h> (стандарт С99) дає можливість виконувати всі функції бібліотеки <math.h>; при цьому аргументи функцій можуть бути як дійсними, так і цілими (у VS 2010 цієї бібліотеки нема).

**Згідно зі стандартом С99 для роботи з типами float і long double існують функції-відповідники з постфіксами f і l до всіх вказаних у таблиці функцій (наприклад, поданій у таблиці функції sin — double sin(double x) відповідають функції sinf — float sinf(float x) і sinl — long double sinl(long double x)).

Усі функції, які одержують або повертають значення кута, працюють з радіанами, наприклад:

180° відповідає число $\pi \approx 3{,}14159265$;

90° — $\pi/2 \approx 1{,}57079632$;

45° — $\pi/4 \approx 0{,}78539816$;

30° — $\pi/6 \approx 0{,}52359878$.

Для переходу від градусної міри до радіанної треба виконати перетворення:

$\varphi_r = \varphi_\circ \cdot \pi/180$;

навпаки:

$\varphi_\circ = \varphi_r \cdot 180/\pi$.

Найосновніші константи бібліотеки math.h подано в таблиці:

Ім'я константи*	**Значення**	**Опис**
M_E	2,71828182845904523536	число e
M_LOG2E	1,44269504088896340736	$\log_2 e$
M_LOG10E	0,434294481903251827651	$\log_{10} e$
M_LN2	0,693147180559945309417	$\ln 2$
M_LN10	2,30258509299404568402	$\ln 10$
M_PI	3,14159265358979323846	число π
M_PI_2	1,57079632679489661923	$\pi/2$
M_PI_4	0,78539816339744830962	$\pi/4$
M_1_PI	0,31830988618379067153	$1/\pi$
M_2_PI	0,63661977236758134308	$2/\pi$
M_2_SQRTPI	1,12837916709551257390	$2/\sqrt{\pi}$
M_SQRT2	1,41421356237309504880	$\sqrt{2}$
M_SQRT1_2	0,70710678118654752440	$1/\sqrt{2}$

*Щоб скористатися цими константами, треба препроцесорові вказати директиву `#define _USE_MATH_DEFINES` ще до директиви підмикання бібліотеки `#include <math.h>`

б) бібліотека stdlib.h (http://www.cplusplus.com/reference/cstdlib/) — **функції**

Ім'я функції	**Прототип**	**Опис**
abs labs llabs	int abs (int n); long int labs (long int n); long long int llabs (long long int n);	абсолютне значення* $\lvert x \rvert$

srand	void srand (unsigned int seed);	ініціалізація генерації випадкових чисел (`srand (1);` або `#include <time.h> /*…*/ srand(time(0));`)
rand	int rand (void);	генерація випадкових чисел (максимальне згенероване число визначається константою RAND_MAX = 32767, мінімальне — 0; `rand()%11` — від 0 до 10; `rand()%3-1` — –1, 0 чи 1)

*Функції abs, labs і llabs розміщуються також і в бібліотеці math.h. Функція abs з бібліотеки math.h дає можливість працювати з даними типу int, long int і long long int. Також функція abs з бібліотеки math.h може працювати з дійсними аргументами, але результатом при цьому є ціле значення (дробова частина відкидається). Функція abs з бібліотеки stdlib.h аргументом може мати тільки ціле значення.

в) бібліотека limits.h (www.cplusplus.com/reference/climits/) — **мінімальні й максимальні цілі значення**

Ім'я константи	**Значення в середовищі VS* і формат виведення функцією printf**	**Опис**
SHRT_MIN	–32768 (-2^{15}) %d або %hd	мінімальне значення об'єкта типу short int
SHRT_MAX	32767 ($2^{15}-1$) %d або %hd	максимальне значення об'єкта типу short int
USHRT_MAX	65535 ($2^{16}-1$) %d, або %hu, або %u	максимальне значення об'єкта типу unsigned short int
INT_MIN	–2147483648 (-2^{31}) %d	мінімальне значення об'єкта типу int
INT_MAX	2147483647 ($2^{31}-1$) %d або %u	максимальне значення об'єкта типу int
UINT_MAX	4294967295 ($2^{32}-1$) %u	максимальне значення об'єкта типу unsigned int
LONG_MIN	–2147483648 (-2^{31}) %d або %ld	мінімальне значення об'єкта типу long int

LONG_MAX	2147483647 (2^{31}–1) %d, або %ld, або %u	максимальне значення об'єкта типу long int
ULONG_MAX	4294967295 (2^{32}–1) %u або %lu	максимальне значення об'єкта типу unsigned long int
LLONG_MIN	–9223372036854775808 (–2^{63}) %lld	мінімальне значення об'єкта типу long long int
LLONG_MAX	9223372036854775807 (2^{63}–1) %lld	максимальне значення об'єкта типу long long int
ULLONG_MAX	18446744073709551615 (2^{64}–1) %llu	максимальне значення об'єкта типу unsigned long long int

*Значення залежить від конкретної системи і реалізації бібліотеки

г) бібліотека float.h (www.cplusplus.com/reference/cfloat/) — **мінімальні й максимальні дійсні значення**

Ім'я константи	**Значення в середовищі VS* і формат виведення функцією printf**	**Опис**
DBL_EPSILON LDBL_EPSILON	2,2204460492503131e–016 %e або %g	найменше додатне значення типу double чи long double таке, що 1,0+DBL_EPSILON ≠1,0
DBL_MAX LDBL_MAX	1,7976931348623158e+308 %f, або %e, або %g	максимальне значення типу double чи long double
DBL_MIN LDBL_MIN	2,2250738585072014e–308 %e або %g	мінімальне додатне значення типу double чи long double
FLT_EPSILON	1,192092896e–07F %e або %g	найменше додатне значення типу float таке, що 1,0+ FLT_EPSILON ≠1,0
FLT_MAX	3,402823466e+38F %f, або %e, або %g	максимальне значення типу float
FLT_MIN	1,175494351e–38F %e або %g	мінімальне додатне значення типу float

6. Основні стандартні функції мови С для введення й виведення інформації

а) бібліотека stdio.h (www.cplusplus.com/reference/cstdio)**, яка реалізує основні можливості введення й виведення інформації**

Ім'я функції	Прототип	Опис
printf	int printf(const char * format, ...);	форматований вивід даних у стандартний потік виведення (stdout)
scanf	int scanf(const char * format, ...);	форматований ввід даних із стандартного потоку введення (stdin)
getchar	int getchar(void);	ввід наступного символа із стандартного потоку введення; чекає натискання клавіші Entet
gets	char * gets(char * str);	ввід рядка символів (можуть бути пробіли) із стандартного потоку введення; чекає натискання клавіші Entet
fseek	int fseek(FILE * filestream, long int offset, int origin);	переміщення покажчика позиції в потоці

Функція printf

Функція printf повертає кількість реально виведених символів (у випадку помилки — від'ємне значення). Звернення до функції:

printf ("рядок формату", список аргументів);

Аргументи у списку відокремлюються комами; аргументами можуть бути вирази у широкому розумінні (вирази, звернення до функцій, змінні, константи). Рядок формату містить специфікатори форматів і символи для виведення.

Специфікатор формату функції printf має вигляд (квадратні дужки не друкуються — вони лише вказують, що параметр не обов'язковий):

% [модифікатори] код_формату

Специфікації формату зіставляються з аргументами відповідно до порядку в списку аргументів (наприклад, `printf("i=%d - %c",325,'?');` дає `i=325 - ?`; `printf("%-+7.5d=i",325);` дає `+00325 =i`).

Коди форматів, які означають, що значенням аргумента є:

d або **i** — десяткове ціле число зі знаком (ці коди еквівалентні);

u — беззнакове десяткове ціле число;

o — вісьміркове ціле число без знака;

x або **X** — шістнадцяткове ціле число без знака з цифрами 0,..., 9, a, b, c, d, e, f або 0,..., 9, A, B, C, D, E, F відповідно;

f — дійсне десяткове число з фіксованою точкою;

e або **E** — дійсне десяткове число в експоненційній формі виду 1.23457e+002 або 1.23457E+002 відповідно;

g або **G** — використовується, як і код f чи e; незначущі нулі не виводяться; залежно від того, яке подання числа буде коротшим, використовується код f чи e (якщо менше чотирьох значущих нулів, то f);

c — символ;

s — рядок символів (символи рядка виводяться до символа кінця рядка або, доки не буде виведено кількість символів, задану точністю);

p — вказівник (виводиться машинна адреса, формат якої сумісний з типом адресації комп'ютера);

n — ніяка інформація на екран не виводиться; аргумент, який відповідає цьому коду, має бути вказівником на цілочисельну змінну (у цю змінну записується кількість виведених символів).

Модифікатори (не обов'язкові; подано в порядку їхнього використання):

мінус (**-**) — вирівнювання значеня по лівому краю у своєму полі (ставиться зразу після %; якщо мінуса нема — по правому);

плюс (+) — число виводиться зі знаком;

ціле число — специфікація мінімальної ширини поля (зайві позиції заповнюються пробілами; якщо довжина числа чи рядка символів більша, то значення виводиться повністю);

ціле число з нулем попереду — специфікація мінімальної ширини поля (зайві позиції заповнюються нулями; якщо довжина числа чи рядка символів більша, то значення виводиться повністю);

крапка і ціле число — модифікатор точності вказується після специфікації мінімальної ширини. Для форматів e, E, f — кількість цифр після коми; g чи G — максимальна кількість значущих цифр; d чи i — мінімальна кількість цифр і в разі необхідності перед числом буде додано нулі; s — максимальна довжина поля (якщо рядок буде довшим, то кінцеві символи не виведуться);

зірочка (*) — значення специфікації мінімальної ширини поля і модифікатора точності можна задати не константами, а аргументами функції, вказавши зірочку, яка зіставляється з відповідним аргументом у списку аргументів;

h — визначає аргумент типу short int і записується перед кодами форматів d, i, o, u, x, X; перед кодом n — вказує, що відповідний аргумент має тип short int;

l — визначає аргумен типу long int (записується перед кодами форматів d, i, u, o, x, X) чи long double (стандарт C99; записується перед кодами форматів f, e, E, g, G, але **не дає ніякого ефекту**); перед кодом n — вказує, що відповідний аргумент має тип long int; перед кодом c — вказує, що відповідний аргумент є символом в розширеному 16-бітовому алфавіті;

L — визначає аргумен типу long double (записується перед кодами форматів f, e, E, g, G);

hh (стандарт C99) — визначає аргумен типу signed чи unsigned char і записується перед кодами форматів d, i, u, o, x, X або вказівник на змінну типу signed char для коду n;

ll (стандарт C99) — визначає аргумен типу signed чи unsigned long long int і записується перед кодами форматів d, i, u, o, x, X або вказівник на змінну типу long long int для коду n;

F і **N** — визначають відповідно вказівники типу far і near;

— записується перед кодами форматів f, e, E, g, G, щоб завжди виводити десяткову точку; перед x чи X — виводити префікс 0x чи

0X перед шістнадцятковим поданням числа; перед o — виводити префікс 0 перед вісьмірковим поданням числа;

якщо після % записано не символ перетворення, то він виводиться на екран (якщо задати %%, то на екран буде виведено символ %).

Найчастіше використовувані *символьні константи керування*, які мають зарезервовані позначення:

\n — перехід на новий рядок;

\t — горизонтальна табуляція (8 позицій);

\r — повернення курсора на початок поточного рядка;

\b — повернення курсора на на одну позицію вліво (назад) в поточному рядку;

\a —короткий звуковий сигнал;

\\ — виведення символа \ ;

\' — виведення символа ' ;

\" — виведення символа " ;

\? — виведення символа ? (символ “?” потрапив до зарезервованих символів у зв’язку з тим, що в Сі є операція, яка позначається знаком “?”; у деяких ситуаціях це могло б призвести до неоднозначного тлумачення конструкцій мови. Символьну константу “знак питання” можна записати будь-яким способом: '?', '\77', '\x3F' чи '\?').

Для переносу тексту на наступний рядок використовується символ \ :

```
printf ("AAAA\   /*початок тексту*/
BBB\n "); /*закінчення тексту*/
printf("AAAABBB\n"); /*еквівалентний оператор*/
```

Функція scanf

Функція scanf повертає кількість реально введених і присвоєних змінним елементів даних (полів). При виявленні помилки до присвоєння значення першого поля функція повертає значення EOF. Звернення до функції:

scanf ("рядок формату", список аргументів);

Усі аргументи повинні бути вказівниками на змінні — адресами змінних (наприклад, **`&aaa`**, **`&I`**, **`str`**, де **`str`** — рядок символів). Рядок формату містить символи трьох категорій: специфікатори форматів, пробільні символи і символи, відмінні від пробільних.

Специфікатор формату функції scanf має вигляд (модифікатори не обов'язкові):

% [модифікатори] код_формату

Специфікації формату зіставляються з аргументами відповідно до їхнього порядку в списку аргументів (наприклад: `scanf("%d%f%c%s",&a,&b,&ch,str);` — введення цілого і дійсного чисел, символа і рядка; або `scanf("%d %f %c %10s",&a,&b,&ch,str);` — пробіли між форматами ролі не грають).

Коди форматів, які означають, що зчитуваним значенням аргумента є:

d — десяткове ціле число зі знаком;

i —ціле число зі знаком у будь-якому форматі (десяткове, вісьміркове, шістнадцяткове);

u — беззнакове десяткове ціле число;

f або **F** (стандарт С99)— дійсне десяткове число типу float з фіксованою точкою;

e або **E** — дійсне десяткове число типу float в експоненційній формі (при введенні символ E має бути на тому ж регістрі, що й код);

g або **G** — дійсне десяткове число типу float;

o — вісьміркове ціле число без знака;

x або **X** — шістнадцяткове ціле число без знака (при введенні цифри від A до F мають бути на тому ж регістрі, що й код);

c — один символ;

s — рядок символів (до першого пробільного символа; інакше — застосовувати функцію gets);

p — вказівник;

%% — читає знак відсотка.

Модифікатори (не обов'язкові):

l — перед кодами f, e, E, g, G для зчитування даних у змінні типу double; перед c, s — для зчитування у двобайтові символи чи рядки двобайтових символів (для зчитування даних типу double %f не підходить — підходить %lf);

L — перед кодами f, e, E, g, G для зчитування даних у змінні типу long double;

hh (стандарт C99) — визначає аргумен типу signed чи unsigned char і записується перед кодами форматів d, i, u, o, x, X або вказівник на змінну типу signed char для коду n.

ll (стандарт C99) — визначає аргумен типу signed чи unsigned long long int і записується перед кодами форматів d, i, u, o, x, X або вказівник на змінну типу long long int для коду n;

ціле число — модифікатор максимальної довжини поля (ставиться між знаком % і кодом формату) обмежує кількість символів, які зчитуються; якщо роздільник буде зчитано раніше, ніж досягнуто вказаної максимальної довжини поля, то введення даних припиниться;

зірочка (*) — читає, але не присвоює дані заданого типу (ставиться між знаком % і кодом формату).

Пробільні символи — роздільники (якщо в рядку форматування поточним символом є роздільник, то у вхідному потоці пропускається один чи кілька роздільників до першого символа, відмінного від роздільника):

пробіли;

символи табуляції (Tab);

роздільники рядків (Enter);

символи кома, крапка з комою тощо не є роздільниками.

Не зважаючи на те, що пробіли, символи табуляції і роздільники рядків використовуються як роздільники полів зчитування, при зчитуванні окремого символа (формат %c) вони читаються, як і будь-який інший символ.

Символи, *відмінні від пробільних* — якщо в рядку форматування поточним символом є символ, відмінний від роздільника, то функція прочитає і відкине аналогічний символ у вхідному потоці; якщо вказаного символа у вхідному потоці нема, то функція закінчує роботу.

б) бібліотека stdlib.h (www.cplusplus.com/reference/stdlib)

Ім'я функції	**Прототип**	**Опис**
system	int system(const char* command);	виконання системної команди (`system("pause")` — чекає натискання будь-якої клавіші на клавіатурі; `system("cls");` — очищає екран; `system("chcp 1251")` — установка кодової сторінки win-cp 1251 (кодування ANSI) в потік вводу і виводу (change codepage); `system("chcp 1251 & cls");` `system(NULL); system("dir");` `system("color 0B");`)

Додаток Г

Методи сортування масивів

В описаних нижче методах сортування вважається, що вхідна (не відсортована) і вихідна (відсортована) множини розміщуються в тій самій ділянці пам'яті (одному масиві). Перед початком сортування вхідна множина займає всю ділянку пам'яті, а вихідна — порожня. У процесі сортування вихідна множина розширюється, а вхідна — звужується. Наведені методи досить прості, вони потребують $O(n^2)$ порівнянь елементів, але кількість перестановок елементів у них може бути різною.

1. Обмінне сортування простою вибіркою з пошуком мінімуму чи максимуму. Спочатку вихідна множина порожня, вхідною множиною є весь масив, а поточним елементом — елемент з номером 1. У вхідній множині вибираємо мінімальний (максимальний) елемент і міняємо його місцем з поточним. Після цього вихідна множина, яка розміщується на початку масиву, містить один елемент, вхідна — всі решту елементів, а поточним елементом є елемент з номером 2. Знову у вхідній множині вибираємо мінімальний (максимальний) елемент і міняємо його з поточним і т.д., доки не досягнемо кінця масиву. Кількість порівнянь — $n(n–1)/2$, кількість перестановок — $(n–1)$.

2. Обмінне сортування простою вибіркою з пошуком мінімуму й максимуму. Спочатку вихідна множина порожня, вхідною множиною є весь масив, а поточними елементами — елемент з номером 1 і елемент з номером n. У вхідній множині вибираємо мінімальний і максимальний елементи і міняємо їх місцями відповідно з поточним першим і поточним останнім (коли мінімальний і максимальний елементи є відповідно поточним останнім і поточним першим, треба виконати тільки одну перестановку). Після цього вихідна множина, яка розміщується на початку й у кінці масиву, містить два крайні елементи, вхідна — всі решту елементів, а поточними елементами є елементи з номерами 2 і $(n–1)$. Знову у вхідній множині вибираємо мінімальний і максимальний елементи і міняємо їх з поточними і т.д.,

доки не досягнемо середини масиву. Кількість порівнянь — $n(n-1)/2$, кількість перестановок — $(n-1)$.

3. Обмінне сортування: метод бульбашки. Спочатку вихідна множина порожня, вхідною множиною є весь масив. Попарно порівнюємо сусідні елементи множини і, якщо їхній порядок не відповідає критерію впорядкованості, то елементи міняються місцями. У результаті одного такого проходу по масиву при сортуванні за зростанням (спаданням) елемент з найбільшим (найменшим) значенням стане останнім у масиві — отже, вихідна множина розміститься в кінці масиву і міститиме один елемент, а вхідна — перші $(n-1)$ елемент. При наступному проході на своє місце стане наступний елемент і т.д. За $(n-1)$. прохід множина буде впорядкована. Кількість порівнянь — $n(n-1)/2$, середня кількість перестановок — теж $n(n-1)/2$.

4. ***Метод бульбашки з індикатором перестановки***. Спочатку вихідна множина порожня, вхідною множиною є весь масив, індикатор перестановки набуває значення false. Попарно порівнюємо сусідні елементи множини і, якщо їхній порядок не відповідає критерію впорядкованості, то елементи міняються місцями і індикатор набуває значення true. У результаті одного такого проходу по масиву при сортуванні за зростанням (спаданням) елемент з найбільшим (найменшим) значенням стане останнім у масиві — отже, вихідна множина розміститься в кінці масиву і міститиме один елемент, а вхідна — перші $(n-1)$ елемент. Якщо значення індикатора після проходу залишилося рівним false, то це означає, що всі елементи відсортовано і сортування закінчено. Якщо ж значення індикатора стало true, то виконуємо наступний прохід. При наступному проході на своє місце стане наступний елемент і т.д. За не більш ніж $(n-1)$ прохід множина буде впорядкована. Якщо вхідна множина впорядкована, то буде зроблено всього один прохід.

5. Метод бульбашки з індикатором перестановки і запам'ятовуванням позиції останньої перестановки (враховує, що на своє місце за один прохід може стати не лише один, а й більше елементів множини). Спочатку вихідна множина порожня, вхідною множиною є весь масив, позиція остан-

ньої перестановки має значення *n*, індикатор перестановки набуває значення false. Попарно порівнюємо сусідні елементи множини і, якщо їхній порядок не відповідає критерію впорядкованості, то позиція перестановки набуває значення номера першого з елементів, які переставляються, а індикатор перестановки набуває значення true. У результаті одного такого проходу по масиву при сортуванні за зростанням (спаданням) кілька елементів з найбільшими (найменшими) значеннями стануть останніми у масиві — отже, вихідна множина розміститься в кінці масиву, а вхідна — на початку. Якщо позиція останньої перестановки дорівнює 1 або індикатор має значення false, то це означає, що всі елементи відсортовано і сортування закінчено. Якщо сортування не закінчено, то при наступному проході на своє місце стануть наступні елементи і т.д. За не більш ніж (*n*–1) прохід множина буде впорядкована. Якщо вхідна множина вже впорядкована, то буде зроблено один прохід.

6. Шейкер-сортування *(модифікація методу бульбашки)*. Спочатку вихідна множина порожня, вхідною множиною є весь масив. При першому (непарному) проході масив проглядається від початку до кінця: попарно порівнюються сусідні елементи множини і, якщо їхній порядок не відповідає критерію впорядкованості, то елементи міняються місцями. У результаті такого проходу при сортуванні за зростанням (спаданням) елемент з найбільшим (найменшим) значенням стане останнім у масиві — отже, вихідна множина розміститься в кінці масиву і міститиме один елемент, а вхідна — на початку і міститиме перші (*n*–1) елемент. При наступному (парному) проході вхідна множина переглядається від кінця до початку і при сортуванні за зростанням (спаданням) елемент з найменшим (найбільшим) значенням стане першим у масиві. Таким чином вихідна множина формується в кінці і на початку масиву. Проходи від початку до кінця і від кінця до початку масиву чергуються, доки не буде досягнуто середини масиву.

7. Шейкер-сортування з індикатором перестановки. Спочатку вихідна множина порожня, вхідною множиною є весь масив, індикатор перестановки набуває значення false. При першому (непарному) проході масив прогля-

дається від початку до кінця: попарно порівнюються сусідні елементи множини і, якщо їхній порядок не відповідає критерію впорядкованості, то елементи міняються місцями і індикатор набуває значення true. У результаті такого проходу при сортуванні за зростанням (спаданням) елемент з найбільшим (найменшим) значенням стане останнім у масиві — отже, вихідна множина розміститься в кінці масиву і міститиме один елемент, а вхідна — перші (n–1) елемент. Якщо значення індикатора після проходу залишилося рівним false, то це означає, що всі елементи відсортовано і сортування закінчено. Якщо ж значення індикатора стало true, то індикатору присвоюється значення false і виконується наступний прохід. При наступному (парному) проході вхідна множина переглядається від кінця до початку і при сортуванні за зростанням (спаданням) елемент з найменшим (найбільшим) значенням стане першим у масиві. Таким чином вихідна множина формується в кінці і на початку масиву. Проходи від початку до кінця і від кінця до початку масиву чергуються, доки не буде досягнуто середини масиву чи значення індикатора не залишиться рівним false.

8. Сортування простими вставками. Спочатку вважаємо, що перший елемент масиву утворює вихідну впорядковану множину, а вхідною множиною є решта невпорядкованих елементів масиву. Перший елемент вхідної множини (спочатку це другий елемент всього масиву) стає поточним елементом. Для вставки поточного елемента в відсортовану за зростанням (спаданням) частину масиву вихідна множина переглядається від кінця до початку і кожен її елемент порівнюється з поточним: якщо поточний елемент менший (більший) від порівнюваного, то порівнюваний елемент зсувається на одну позицію вправо (стає наступним), якщо поточний елемент не менший (не більший) від порівнюваного, то він ставиться на наступну позицію після порівнюваного (цю позицію займав елемент, який перемістився вправо, або це позиція поточного елемента) ставимо поточний елемент. На кожному кроці сортування вихідна множина збільшується на один елемент, а вхідна зменшується на один елемент.

Додаток Ґ

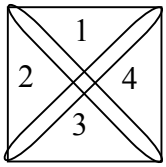
Обхід секторів квадратної матриці

Нехай є квадратна матриця розміру $n \times n$; при цьому n може бути як парним, так і непарним. Розглянемо алгоритми вибору елементів з частин (секторів) матриці. Незалежно від того, порядок матриці n є парним чи непарним числом, в алгоритмах використано ту властивість, що результатом ділення двох цілих чисел є ціле число — дробова частина відкидається.

Щоб краще зрозуміти ці алгоритми, можна скористатися матрицями індексів, наприклад, розмірів 4×4 (n — парне) і 5×5 (n — непарне):

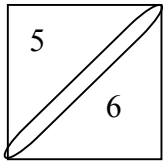
$$\begin{matrix} ij \\ \begin{pmatrix} \mathbf{00} & 01 & 02 & \mathbf{03} \\ 10 & \mathbf{11} & \mathbf{12} & 13 \\ 20 & \mathbf{21} & \mathbf{22} & 23 \\ \mathbf{30} & 31 & 32 & \mathbf{33} \end{pmatrix} \end{matrix} \qquad \begin{matrix} ij \\ \begin{pmatrix} \mathbf{00} & 01 & 02 & 03 & \mathbf{04} \\ 10 & \mathbf{11} & 12 & \mathbf{13} & 14 \\ 20 & 21 & \mathbf{22} & 23 & 24 \\ 30 & \mathbf{31} & 32 & \mathbf{33} & 34 \\ \mathbf{40} & 41 & 42 & 43 & \mathbf{44} \end{pmatrix} \end{matrix}$$

Подані нижче алгоритми є найефективнішими, оскільки в них цикли виконуються тільки для тих елементів, які мають бути вибрані, і тут немає зайвих перевірок.



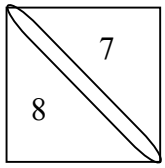
Частина 1 $\vee$ без діагональних елементів:

```
for (i=0; i<(n-1)/2; i++)
  for (j=i+1; j<n-i-1; j++) a[i][j]=1;
```



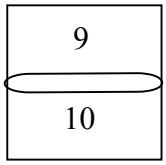
Частина 1 $\vee$ з діагональними елементами:

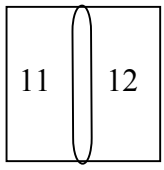
```
for (i=0; i<(n+1)/2; i++)
  for (j=i; j<n-i; j++) a[i][j]=1;
```



Частина 2 > без діагональних елементів:

```
for (i=1; i<n-1; i++) {
  if (i<(n+1)/2)
    jr=i;
  else
    jr=n-i-1;
  for (j=0; j<jr; j++) a[i][j]=1;
}
```



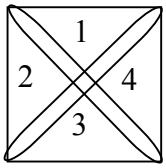


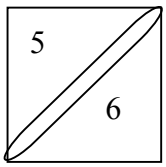
Частина 2 > без діагональних елементів (без перевірки умови) обробляється так само, як і частина 1, тільки треба поміняти *i* й *j* у заголовках циклів:

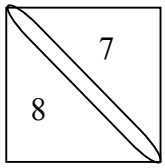
```
for (j=0; j<(n-1)/2; j++)
  for (i=j+1; i<n-j-1; i++) a[i][j]=1;
```

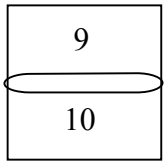
Частина 2 > з діагональними елементами:

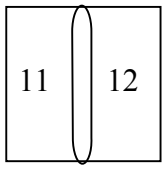
```
for (i=0; i<n; i++) {
  if (i<(n+1)/2)
    jr=i+1;
  else
    jr=n-i;
  for (j=0; j<jr; j++) a[i][j]=1;
}
```











Частина 2 > з діагональними елементами (без перевірки умови) обробляється так само, як і частина 1, тільки треба поміняти *i* й *j* у заголовках циклів:

```
for (j=0; j<(n+1)/2; j++)
  for (i=j; i<n-j; i++) a[i][j]=1;
```

Частина 3 ∧ без діагональних елементів:

```
for (i=n/2+1; i<n; i++)
  for (j=n-i; j<i; j++) a[i][j]=1
```

Частина 3 ∧ з діагональними елементами:

```
for (i=n/2; i<n; i++)
  for (j=n-i-1; j<i+1; j++) a[i][j]=1;
```

Частина 4 < без діагональних елементів:

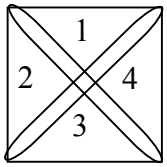
```
for (i=1; i<n-1; i++) {
  if (i<(n+1)/2)
    jr=n-i;
  else
    jr=i+1;
  for (j=jr; j<n; j++) a[i][j]=1;
}
```

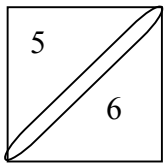
Частина 4 < без діагональних елементів (без перевірки умови) обробляється так само, як і частина 3, тільки треба поміняти *i* й *j* у заголовках циклів:

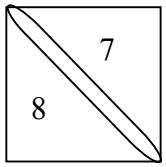
```
for (j=n/2+1; j<n; j++)
  for (i=n-j; i<j; i++) a[i][j]=1;
```

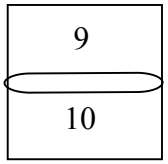
Частина 4 < з діагональними елементами:

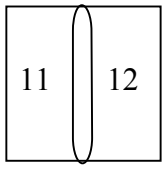
```
for (i=0; i<n; i++) {
  if (i<(n+1)/2)
    jr=n-i-1;
  else
    jr=i;
  for (j=jr; j<n; j++) a[i][j]=1;
}
```











Частина 4 < з діагональними елементами (без перевірки умови) обробляється так само, як і частина 3, тільки треба поміняти *i* й *j* у заголовках циклів:

```
for (j=n/2; j<n; j++)
  for (i=n-j-1; i<j+1; i++) a[i][j]=1;
```

Частина 5 / без діагональних елементів:

```
for (i=0; i<n-1; i++)
  for (j=0; j<n-i-1; j++) a[i][j]=1;
```

Частина 5 / з діагональними елементами:

```
for (i=0; i<n; i++)
  for (j=0; j<n-i; j++) a[i][j]=1;
```

Частина / 6 без діагональних елементів:

```
for (i=1; i<n; i++)
  for (j=n-i; j<n; j++) a[i][j]=1;
```

Частина / 6 з діагональними елементами:

```
for (i=0; i<n; i++)
  for (j=n-i-1; j<n; j++) a[i][j]=1;
```

Частина \ 7 без діагональних елементів:

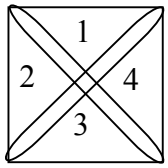
```
for (i=0; i<n-1; i++)
  for (j=i+1; j<n; j++) a[i][j]=1
```

Частина \ 7 з діагональними елементами:

```
for (i=0; i<n; i++)
  for (j=i; j<n; j++) a[i][j]=1;
```

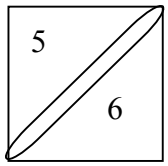
Частина 8 \ без діагональних елементів \

```
for (i=1; i<n; i++)
  for (j=0; j<i; j++) a[i][j]=1;
```



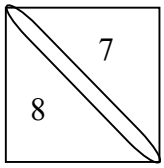
Частина 8 \ з діагональними елементами:

```
for (i=0; i<n; i++)
  for (j=0; j<i+1; j++) a[i][j]=1;
```



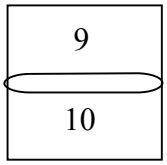
Частина 9 — без елементів центрального рядка:

```
for (i=0; i<n/2; i++)
  for (j=0; j<n; j++) a[i][j]=1;
```



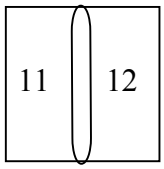
Частина 9 — з елементами центрального рядка при непарних *n*:

```
for (i=0; i<(n+1)/2; i++)
  for (j=0; j<n; j++) a[i][j]=1;
```



Частина — 10 без елементів центрального рядка:

```
for (i=(n+1)/2; i<n; i++)
    for (j=0; j<n; j++) a[i][j]=1;
```



Частина — 10 з елементами центрального рядка при непарних *n*:

```
for (i=n/2; i<n; i++)
    for (j=0; j<n; j++) a[i][j]=1;
```

Частина 11 | без елементів центрального стовпчика:

```
for (i=0; i<n; i++)
  for (j=0; j<n/2; j++) a[i][j]=1;
```

Частина 11 | з елементами центрального стовпчика при непарних *n*:

```
for (i=0; i<n; i++)
  for (j=0; j<(n+1)/2; j++) a[i][j]=1;
```

Частина | 12 без елементів центрального стовпчика:

```
for (i=0; i<n; i++)
  for (j=(n+1)/2; j<n; j++) a[i][j]=1;
```

Частина | 12 з елементами центрального стовпчика при непарних n:

```
for (i=0; i<n; i++)
  for (j=n/2; j<n; j++) a[i][j]=1;
```

Розглянутий спосіб вибору частин матриці є найефективнішим, але в кожному випадку швидко згадати потрібний алгоритм, який буде правильно працювати для матриць парних і непарних розмірів, часом не зовсім просто.

Існує інший, менш ефективний, але набагато простіший спосіб вибору елементів матриці — переглядаються всі можливі n^2 комбінацій індексів і серед них вибираються потрібні. У цьому випадку порівняно з попереднім підходом виконується приблизно в 2 чи в 4 рази більше проходів циклів і, крім того, виконуються перевірки в цих циклах, тобто машинний час витрачається неефективно. При виборі потрібних комбінацій індексів використовуються властивості суми й різниці індексів. Розглянемо це на прикладі матриць 4×4 і 5×5.

$$ij\begin{pmatrix}\mathbf{00} & 01 & 02 & \mathbf{03}\\ 10 & \mathbf{11} & \mathbf{12} & 13\\ 20 & \mathbf{21} & \mathbf{22} & 23\\ \mathbf{30} & 31 & 32 & \mathbf{33}\end{pmatrix}\qquad i+j\begin{pmatrix}0 & 1 & 2 & \mathbf{3}\\ 1 & 2 & \mathbf{3} & 4\\ 2 & \mathbf{3} & 4 & 5\\ \mathbf{3} & 4 & 5 & 6\end{pmatrix}\qquad i-j\begin{pmatrix}\mathbf{0} & -1 & -2 & -3\\ 1 & \mathbf{0} & -1 & -2\\ 2 & 1 & \mathbf{0} & -1\\ 3 & 2 & 1 & \mathbf{0}\end{pmatrix}$$

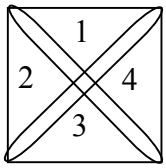
$$ij\begin{pmatrix}\mathbf{00} & 01 & 02 & 03 & \mathbf{04}\\ 10 & \mathbf{11} & 12 & \mathbf{13} & 14\\ 20 & 21 & \mathbf{22} & 23 & 24\\ 30 & \mathbf{31} & 32 & \mathbf{33} & 34\\ \mathbf{40} & 41 & 42 & 43 & \mathbf{44}\end{pmatrix}\qquad i+j\begin{pmatrix}0 & 1 & 2 & 3 & \mathbf{4}\\ 1 & 2 & 3 & \mathbf{4} & 5\\ 2 & 3 & \mathbf{4} & 5 & 6\\ 3 & \mathbf{4} & 5 & 6 & 7\\ \mathbf{4} & 5 & 6 & 7 & 8\end{pmatrix}\qquad i-j\begin{pmatrix}\mathbf{0} & -1 & -2 & -3 & -4\\ 1 & \mathbf{0} & -1 & -2 & -3\\ 2 & 1 & \mathbf{0} & -1 & -2\\ 3 & 2 & 1 & \mathbf{0} & -1\\ 4 & 3 & 2 & 1 & \mathbf{0}\end{pmatrix}$$

Як видно з наведених матриць, для $i+j$ на побічній діагоналі стоять числа $(n-1)$, вище від неї — числа, менші від $(n-1)$, а нижче — більші від $(n-1)$; для $i-j$ на головній діагоналі стоять нулі, вище від неї — від'ємні

числа, а нижче — додатні числа. Використовуючи ці властивості, можна вибирати певні частини матриці.

Усі подані нижче алгоритми мають однаковий вигляд:

```
for (i=0; i<n; i++)
  for (j=0; j<n; j++)
    if (умова) a[i][j]=1;
```



і відрізняються лише умовою. Умови будуть такими:

`i+j<n-1 && i-j<0` — частина 1 $\vee$ без діагональних елементів;

`i+j<=n-1 && i-j<=0` — частина 1 $\vee$ з діагональними елементами;

`i+j<n-1 && i-j>0` — частина 2 > без діагональних елементів;

`i+j<=n-1 && i-j>=0` — частина 2 > з діагональними елементами;

`i+j>n-1 && i-j>0` — частина 3 $\wedge$ без діагональних елементів;

`i+j>=n-1 && i-j>=0` — частина 3 $\wedge$ з діагональними елементами;

`i+j>n-1 && i-j<0` — частина 4 < без діагональних елементів;

`i+j>=n-1 && i-j<=0` — частина 4 < з діагональними елементами.

З наведених умов видно, що якщо елементи діагоналі входять в частину матриці, то використовується порівняння <= чи >=, а якщо не входять, то використовується порівняння < чи >.

Рекомендована література

1. Керниган Б., Ритчи Д. Язык программирования Си. Пер. с англ., 3-е изд., испр. — СПб.: Невский Диалект, 2001. — 352 с.

2. Шилдт Г. Полный справочник по C++. 4-е издание / Герберт Шилдт. — М., СПб., К.: Вильямс, 2006. — 801 с. — С. 27-222.

3. Вінник В.Ю. Алгоритмічні мови та основи програмування: мова Сі. — Житомир: ЖДТУ, 2007. — 328 с.

4. Шапошникова С.В. Особенности языка С. Учебное пособие [Електронний ресурс] / С.В. Шапошникова. — 2012. — 101 с. — Режим доступу: http://younglinux.info/sites/default/files/programmingC.pdf

5. Поляков К. Программирование на языке Си: Интернет-издание, в 4-х частях [Електронний ресурс] / К. Поляков. — 1995-2014. — 228 с. — Режим доступу: http://kpolyakov.spb.ru/school/c.htm

6. IEEE Standard for Floating-Point Arithmetic. — New York, 2008. — 58 p.

7. Хьюз Дж., Мичтом Дж. Структурный подход к программированию. — М.: Мир, 1980. — 280 с.

8. Мейер Б., Бодуэн К. Методы программирования: В 2-х томах. Т. 2. — М.: Мир, 1982. — 368 с.

Лариса Іванівна Кублій

e-mail: kublii_l_i@ukr.net

тел.: (063)-71-91-231