МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

І.П. МУХА, І.І. ВІТКОВСЬКА, М.М. ГОЛОВЧЕНКО

ОСНОВИ ПРОГРАМУВАННЯ. ЧАСТИНА 2. МОДУЛЬНЕ ПРОГРАМУВАННЯ

Лабораторний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
спеціальності 121 «Інженерія програмного забезпечення»
за освітньою програмою «Інженерія програмного забезпечення інформаційних систем»

Київ
КПІ ім. Ігоря Сікорського
2022

Основи програмування. Частина 2. Модульне програмування [Електронний ресурс] : навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» / КПІ ім. Ігоря Сікорського; уклад.: І.П. Муха, І.І.Вітковська, М.М. Головченко. – Електронні текстові дані (1 файл: 585 КБ). – Київ: КПІ ім. Ігоря Сікорського, 2022. – 83 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол №3 від 01.12.2022 р.) за поданням Вченої ради факультету інформатики та обчислювальної техніки (протокол №8 від 09.11.2022 р.)

ОСНОВИ ПРОГАМУВАННЯ. ЧАСТИНА 2.

МОДУЛЬНЕ ПРОГРАМУВАННЯ

Лабораторний практикум

Укладачі: *МУХА Ірина Павлівна, канд. техн. наук, доц*

ВІТКОВСЬКА Ірина Іванівна, ст.викладач

ГОЛОВЧЕНКО Максим Миколайович, ст.викладач

Відповідальний редактор

Рецензент Ткач М. М., канд. техн. наук, доцент, доцент кафедри інформаційні системи та технології НТУУ «КПІ ім. Ігоря Сікорського»

Навчальний посібник «Основи програмування. Частина 2. Модульне програмування» призначений для проведення лабораторного практикуму для студентів денної та заочної форми навчання бакалаврського освітнього ступеня спеціальності 121 «Інженерія програмного забезпечення» за освітньою програмою «Інженерія програмного забезпечення інформаційних систем».

У навчальному посібнику наведено варіанти індивідуальних завдань. До кожної роботи надаються вказівки щодо виконання завдань, тестування програм та оформлення звіту, а також наведені контрольні питання для підготовки до захисту лабораторних робіт.

Реєстр. № НП 22/23-265. Обсяг 3,2 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

ВСТУП

Дисципліна «Основи програмування. Частина 2. Модульне програмування» є базовою спеціальною нормативною дисципліною підготовки фахівців рівня бакалавр спеціальності 121 «Інженерія програмного забезпечення».

Мета лабораторних занять – практичне підтвердження теоретичних положень дисципліни «Основи програмування. Частина 2. Модульне програмування» шляхом поглиблення знань студентів у застосуванні принципів об'єктно-орієнтованого програмування для розроблення надійних і стійких програм, а також вироблення практичних навичок з укладання та використання програм.

Кожна лабораторна робота повинна виконуватися особисто студентом згідно з його варіантом завдання під керівництвом викладача. Номер варіанта задається викладачем до кожної лабораторної роботи.

Перед виконанням кожної лабораторної роботи студент повинен продемонструвати викладачеві досягнутий рівень самостійної підготовки шляхом відповідей на додаткові запитання зі списку, наведеного в кінці кожної лабораторної роботи, та поданням робочих матеріалів щодо виконання лабораторної роботи (текст програми, методику налагодження).

Після виконання лабораторної роботи студент повинен захистити роботу: продемонструвати працюючу програму, оформити звіт, відповісти на запитання.

Тексти програм повинні відповідати встановленим вимогам до стилю оформлення.

Виконанню лабораторної роботи повинна передувати самостійна підготовка, у процесі якої за конспектом лекцій, теоретичною частиною відповідної лабораторної роботи або за підручниками, на які наведені посилання в переліку літератури, необхідно вивчити відповідні конструкції мов програмування C++/ C#, або Pyton, скласти програму, створити звіт та зберегти це на хмарне сховище Github або Google диск.

Лабораторна робота 1
ТЕКСТОВІ ФАЙЛИ

Мета роботи – вивчити особливості створення і обробки текстових файлів даних.

Теоретичні відомості

Текстовий файл є сукупністю символьних рядків змінної довжини. Кінець кожного рядка у такому файлі позначається спеціальним маркером кінця рядка Eoln, що представляє собою пару керуючих символів: «переведення каретки» (#13) і «переведення рядка» (#10). Наприкінці файла записується маркер кінця файла — керуючий символ з кодом #26. Текстові файли є файлами тільки послідовного доступу.

Доступ до фізичних файлів у програмах здійснюється за допомогою файлових змінних (об'єктів).

У С-програмах для цього використовується файловий покажчик стандартного типу FILE, який містить різноманітні відомості про файл (його ім'я, розташування на диску, режим роботи з файлом і т. ін.), у С++-програмах – об'єкт файлового потоку (наприклад, istream), у Python – об'єкти, з яким пов'язані уявлення про конкретну структуру даних, що зберігатиметься у файлі.

При роботі з файлами даних слід відрізняти власне обробку даних і їх зчитування з файлу або запису у файл. Обробка даних є логікою програми і не має відношення до роботи з файлами.

У С++ файлове введення / виведення забезпечують *файлові потоки*. Для їх реалізації передбачено класи:

- ifstream (для читання даних із файлу),
- ofstream (для запису даних у файл),
- fstream (для читання і для запису даних).

Дані класи описані у заголовному файлі <fstream>.

Наприклад,

```
#include <fstream>
ofstream outFile;       // створення файлового потоку виведення
ifstream inFile;        // створення файлового потоку введення
fstream ioFile;         // створення файлового потоку для введення і виведення
```

Перед початком роботи будь-який фізичний файл необхідно зв'язати з об'єктом, який буде представляти цей файл у програмі. Як правило, цей зв'язок встановлюється при *відкритті файлу*.

Оскільки можливі різні режими роботи з файлом, то при відкритті файлу, зазвичай, уточнюється, в якому режимі він відкривається: «читання», «запису» або «читання» і «запису».

Відкриття файлу засобами C++ здійснюється за допомогою функції open():

```
void open(const char filename, int mode, int access);
```

Тут параметр filename – це рядок, що визначає повне ім'я фізичного файлу, який відкривається (може містити специфікацію шляху до файлу у файловій структурі диску); mode – режим відкриття файлу; access – визначає, яким чином здійснюється доступ до файлу.

Усі можливі режими відкриття файлу (параметр mode) визначені у заголовному файлі <fstream> (табл. 1).

Таблиця 1. Специфікатори режиму відкриття файлів у C++

Режим	*Призначення*
ios::in	відкрити файл для зчитування
ios::out	відкрити файл для запису
ios::binary	відкрити файл, вважаючи його бінарним
ios::app	відкрити для запису даних в кінець файла
ios::ate	після відкриття перемістити позицію зчитування-запису на кінець файла
ios::trunc	вилучити дані з існуючого файла

Режими відкриття файлів можна комбінувати за допомогою порозрядної логі-

чної операції «АБО» (символ “|”), наприклад, режим ios::out | ios::trunc передбачає відкриття файла для запису, попередньо очистивши його.

Значення параметра access визначає режим доступу до файлу. Перелік можливих значень атрибутів файлів наведений у табл. 2.

Таблиця 2. Атрибути доступу до файлів

Атрибут	*Значення*
0	Звичайний файл, відкритий доступ
1	Файл тільки для читання
2	Прихований файл
4	Системний файл
8	Архівний файл

При відкритті файлу можна використовувати параметри за замовчуванням. Так при відкритті файлових ifstream-потоків параметр mode за замовчуванням має значення ios::in; ofstream-потоків – значення ios::out | ios::trunc; fstream-потоків – значення ios::in | ios::out. Для параметра access у будь-якому випадку за замовчуванням використовується варіант звичайного файлу (0). Наприклад, відкриття файлу з ім’ям FileName.txt як для зчитування, так і для запису:

```
#include <fstream>
fstream ioFile;                    // створення файлу для введення і виведення
ioFile.open(«FileName.txt»);
```

У програмах на C++ файл можна відкрити і при створенні відповідного файлового потоку:

```
#include <fstream>
ifstream inFile(«Test .txt»);     //створення текстового файлу введення
```

Для уникання помилок після відкриття файлу слід перевіряти, чи насправді він відкрився:

```
ifstream inFile («Test .txt»);    // відкриття файлу для читання
if (! inFile)                     // перевірка, чи файл відкрився
```

```
{  cout << «Cannot open file \n»;          // обробка помилок
   return;
}
```

Перевірити успішність відкриття файлу можна також за допомогою функції-члена is_open():

```
bool is_open();
```

Якщо потік вдалося зв'язати з відкритим файлом, ця функція повертає значення true, інакше – false.

Обробка файлу полягає у реалізації операцій зчитування даних із файлу або запису даних у файл. Операції читання та запису завжди виконуються, починаючи з поточної позиції у файлі.

При читанні-запису текстових файлів використовується *потокове* та *символьне* введення-виведення.

Потокове введення-виведення передбачає використання операцій "<<" ("помістити в потік") і ">>" ("взяти з потоку"), які виконуються аналогічно консольному введенню-виведенню, тільки замість стандартних потоків cin і cout вказують відповідні файлові потоки:

```
filename << об'єкт1 << об'єкт2 << … << об'єктN;
filename >> об'єкт1 >> об'єкт2 >> … >> об'єктN;
```

Тут filename – ідентифікатор файлового потоку.

При потоковому введенні із файлу зчитування здійснюється до пробілу чи символу нового рядка. Тому таке зчитування придатне лише для окремих слів. Для зчитування з потоку рядка, що складається з декількох слів, використовується функція getline(), яка для рядків string має наступні прототипи:

```
istream& getline(istream&, string&);
istream& getline(istream&, string&, char);
```

В текстові файли можна записувати як символьні, так і несимвольні дані. При

записуванні в текстовий файл такі дані перетворюються із внутрішньої форми подання в символьну і зберігаються у символьному вигляді; при читанні виконується зворотне перетворення.

У Python відкриття файлів даних здійснюється вбудованою функцією open():

fileobj = open(filename, mode, …)

Тут fileobj – об'єкт файла, який повертається функцією open(), filename – повна назва файлу (шлях до файлу), mode – рядок, який вказує на тип файлу і режим роботи з ним.

Перша літера mode задає режим роботи з файлом:

- r – читання з файлу,
- w – запис в існуючий файл (якщо файлу не існує, він буде створений),
- x – запис у новий файл,
- a – додавання в кінець файлу, якщо файл існує.

Друга літера mode вказує на тип файлу. За умовчанням – t (текстовий файл).

При відкритті файлу можна вказати режим обробки символів нового рядка: записати вказівку newline = 'value', де value може набувати значень \n, \r, \r\n або порожнього рядка, в результаті чого тільки вказана послідовність буде використовуватися як ознака нового рядка.

Для запису даних у файл можна використовувати функції print() і write(). Функція write() повертає число записних у файл байтів, але не додає ніяких інших символів у файл, як це робить функція print(), яка також дозволяє записувати у файл.

Для читання даних із файлу у Python використовують функції:

- read() – читання вмісту усього файлу за один раз;
- readline() – читання вмісту усього файлу по одному рядку за раз;
- readlines() – повернення списку зчитаних рядків.

При читанні текстового файлу Python повертає рядки у форматі Unicode. То-

му, якщо інформація у файлі кодувалася не в UTF-8, то буде отримана помилка або незрозумілий текст. Щоб уникнути цього, необхідно явно передати функції open() назву кодування з параметром encoding, наприклад:

data = open('filename.txt', 'rt', encoding='cp1251').read().

Після завершення обробки файлу він повинен бути закритий. Для цього використовуються функція close().

Вимоги до програми

1) Створення файлу програмним шляхом.
2) Організація введення інформації аналогічно текстовим редакторам (ознака кінця рядка – натиснення Enter, ознака кінця введення файла – будь-яка комбінація клавіш або клавіша, що генерують розширений код).
3) Можливість доповнення файла даних.
4) Виведення усіх вхідних, проміжних і вихідних даних.

Варіанти завдань

1. Створити текстовий файл. Знайти найдовші слова в кожному його рядку та переписати їх в новий текстовий файл. Записати останнім рядком нового файлу загальну кількість слів у вихідному файлі. Вивести вміст вихідного і створеного файлів.

2. Створити текстовий файл, деякі рядки якого починаються із символа '#'. Сформувати новий текстовий файл, скопіювавши до нього вміст вихідного файлу таким чином, щоб рядки вихідного файлу, які починаються із символу '#', були переставлені в кінець файлу з видаленням самого символу '#' і додаванням у середину такого рядка символу '!'. Вивести вміст вихідного і створеного файлів.

3. Створити текстовий файл. Слова у файлі відділені пробілами, комами, крапками. У кожному реченні визначити саме довге слово. Створити новий текстовий файл, у якому кожне речення розміщується у окремому рядку і починається із числа, що дорівнює довжині самого довгого слова в ньому, за яким йде саме це сло-

во. Вивести вміст вихідного і створеного файлів.

4. Створити текстовий файл. Сформувати новий текстовий файл, що складається з рядків вхідного файлу, у яких вилучені всі односимвольні слова, а також зайві роздільники (пробіли, коми, крапки), що повторюються підряд. Визначити і дописати в кінець нового файлу кількість вилучених слів і зайвих символів кожного виду. Вивести вміст вихідного і створеного файлів.

5. Створити текстовий файл. В кожному непарному рядку визначити кількість цифрових символів і записати цю величину останнім елементом рядка. Вивести вміст вихідного і перетвореного файлів.

6. Створити два текстових файли. Переписати в новий текстовий файл рядки, які є у другому вихідному файлі, і відсутні у першому. Визначити кількість таких рядків. Вивести вміст вихідного і створеного файлів.

7. Створити текстовий файл. Сформувати новий текстовий файл, що складається зі слів вхідного файлу, які зустрічаються у ньому більше N раз. Розмістити ці слова в новому файлі в порядку зростання їхньої довжини. Вивести вміст вихідного і створеного файлів.

8. Створити текстовий файл. Кожен парний рядок вихідного файлу переписати в перший новий текстовий файл, кожен непарний – у другий. У файлі з парними рядками лексично впорядкувати рядки за алфавітом. У файлі з непарними рядками впорядкувати слова в кожному із перших N рядків. Вивести вміст вихідного і створеного файлів.

9. Створити текстовий файл, рядки якого містять розділені пробілами слова, що складаються із цифр або символів. В кожному рядку, що містить числа, знайти найбільше із них. Переписати такі рядки (що містять числа) у новий текстовий файл наступним чином: починається такий рядок знайденим найбільшим числовим значенням, далі послідовно записуються тільки числові значення відповідного рядка вихідного файлу, розділені комами. Вивести вміст вихідного і створеного файлів.

10. Створити текстовий файл. Сформувати новий текстовий файл, що складається з рядків вихідного файлу, розміщених у порядку зростання кількості символів у них. Наприкінці кожного рядка нового файлу дописати кількість символів у рядку. Вивести вміст вихідного і створеного файлів.

11. Створити текстовий файл. Визначити в кожному його рядку кількість слів і довжину найбільшого слова. Дописати значення кількості слів на початок відповідного рядка, довжину максимального слова – в його кінець. Вивести вміст вхідного і перетвореного файлів.

12. Створити текстовий файл. Визначити в кожному його рядку слова, які починаються на задану літеру. Зберегти знайдені слова у новому текстовому файлі. Як роздільники між словами застосовувати пробіли.

13. Створити текстовий файл. Слова у файлі відділені комами, крапкою з комою, пробілами, крапками. У кожному реченні визначити саме коротке слово. Створити новий текстовий файл, у якому кожне речення розміщується у окремому рядку і закінчується числом, що дорівнює довжині самого короткого слова в ньому, за яким йде саме це слово. Вивести вміст вихідного і створеного файлів.

14. Створити текстовий файл, що містить програму на мові С++. Перевірити текст на рівну кількість відкритих і закритих дужок, вважаючи, що кожен оператор в програмі займає не більше одного рядка вихідного файлу. Вивести вміст файлу і результат аналізу.

15. Створити текстовий файл. Переписати його компоненти до нового текстового файлу, вставляючи в кінець кожного рядка літеру, що є останньою в першому слові рядка. Вивести вміст вихідного і створеного файлів.

16. Створити текстовий файл. Переписати його компоненти до нового текстового файлу, вставляючи на початку кожного рядка поточну довжину рядка у вихідному файлі. Вивести вміст вихідного і створеного файлів.

17. Створити текстовий файл. Переписати його компоненти до нового текстового файлу, вставляючи на початку рядка порядковий номер, в кінці рядка -поточну

довжину рядка у вихідному файлі. Вивести вміст вихідного і створеного файлів.

18. Створити текстовий файл. Сформувати новий текстовий файл, що складається з рядків вихідного файлу, у яких вилучені всі двосимвольні слова. Визначити і дописати в кінець нового файлу кількість вилучених слів. Вивести вміст вихідного і створеного файлів.

19. Створити текстовий файл. Сформувати новий текстовий файл, що складається з рядків вихідного файлу, у яких вилучені початкові та кінцеві пробіли і подвоєні проміжні. Визначити і дописати в кінець кожного рядка нового файла кількість доданих у нього пробілів. Вивести вміст вихідного і створеного файлів.

20. Створити текстовий файл. Переписати до нового текстового файлу всі компоненти вихідного файлу, замінивши в них символ 0 на 1 і навпаки. Вивести вміст вихідного і створеного файлів.

21. Створити текстовий файл. Переписати його вміст у новий текстовий файл, замінюючи кожен рядок символом ‘-‘, якщо він у вихідному файлі закінчується на літеру, відмінну від символа пропуску, коми або крапки. Вивести вміст вихідного і створеного файлів.

22. Створити текстовий файл. Переписати його компоненти до нового текстового файлу, вставляючи на початок кожного рядка літеру, що є першою в останньому слові рядка. Вивести вміст вихідного і створеного файлів.

23. Створити два текстових файли. Переписати в новий текстовий файл рядки, які є в обох вихідних файлах. Визначити кількість таких рядків. Вивести вміст вихідних і створеного файлів.

24. Створити текстовий файл. Слова у файлі відділені пробілами, символами «,» та «;». Переписати в новий текстовий файл рядки вихідного файлу, вставляючи на початок кожного рядка число, що дорівнює кількості однакових слів у цьому рядку. Вивести вміст вихідного і створеного файлів.

25. Створити текстовий файл. Переписати його вміст у новий текстовий файл, переформатувавши рядки вихідного файлу на задану користувачем довжину шля-

хом рівномірної вставки пробілів між словами. Вивести вміст вихідного і створеного файлів.

26. Створити текстовий файл. Скопіювати останні N рядків вихідного файлу до нового текстового файлу. У новому файлі знайти повторювані рядки і видалити їх. Визначити кількість видалених рядків. Вивести вміст вихідного і створеного файлів.

27. Створити текстовий файл. Кожен парний рядок вихідного файлу переписати в перший новий текстовий файл, кожен непарний – у другий. У файлі з парними рядками змінити рядки таким чином, щоб слова кожного рядка були лексично впорядковані за алфавітом. Вивести вміст вихідного і створених файлів.

28. Створити текстовий файл. Сформувати новий текстовий файл, що складається з слів вхідного файлу, які зустрічаються у ньому менше N раз. Розмістити ці слова в новому файлі в порядку спадання їхньої довжини. Вивести вміст вихідного і створеного файлів.

29. Створити два текстових файли. Переписати в новий текстовий файл рядки, які є в першому вихідному файлі, і відсутні у другому. Визначити кількість таких рядків. Вивести вміст вихідних і створеного файлів.

30. Створити текстовий файл. Сформувати новий текстовий файл, що складається з рядків вихідного файлу, розміщених у порядку спадання кількості символів у них. На початку кожного рядка нового файлу вказати кількість символів у рядку. Вивести вміст вихідного і створеного файлів.

31. Створити текстовий файл. Переписати його компоненти до нового текстового файлу, вставляючи на початку рядка поточну довжину рядка у вихідному файлі, в кінці рядка – його порядковий номер. Вивести вміст вихідного і створеного файлів.

32. Створити текстовий файл. Сформувати новий текстовий файл, що складається з рядків вихідного файлу, у яких зустрічається задане користувачем слово. Визначити розмір, дату і час створення нового файлу. Вивести вміст вихідного і створе-

ного файлів.

33. Створити текстовий файл, що є програмою на мові C++. Визначити і записати в новий текстовий файл всі ідентифікатори (по одному ідентифікатору у рядок), які зустрічаються в лівій частині операторів присвоєння у вихідному файлі. Вивести вміст вихідного і створеного файлів.

34. Створити текстовий файл. Кожен парний рядок вихідного файлу переписати в перший новий текстовий файл, кожен непарний – у другий. У файлі з непарними рядками лексично впорядкувати N перших рядків за алфавітом. Вивести вміст вихідного і створених файлів.

35. Створити текстовий файл. Слова у файлі відділені комами, крапкою з комою, пробілами, крапками. У кожному реченні визначити саме довге слово. Створити новий текстовий файл, у якому кожне речення розміщується у окремому рядку і починається числом, що дорівнює довжині самого довгого слова в ньому, за яким йде саме це слово. Вивести вміст вихідного і створеного файлів.

Контрольні питання

1) Особливості організації текстових файлів у C/C++.
2) Чим зумовлена потреба у відкритті та закритті файлів?
3) Як визначається кінець файла у програмі?
4) Чи можливий прямий доступ до рядків текстового файлу?
5) Чим відрізняються текстові файли від бінарних, що складаються із рядків?
6) Коли і для чого при обробці файлів застосовується функція fflush ?
7) Чи можна в текстових файлах зберігати несимвольні дані ?

Лабораторна робота 2
БІНАРНІ ФАЙЛИ

Мета роботи - вивчити особливості створення і обробки бінарних файлів.

Теоретичні відомості

Бінарні файли є послідовністю байтів інформації. Вони використовуються для збереження компонент визначеного типу. При перегляді такого файлу неможливо зрозуміти, що в ньому записано - можна побачити лише послідовність байт в шістнадцятковому форматі. Такий файл не можна створювати або виправляти вручну у текстовому редакторі.

Для отримання доступу до бінарного файлу потрібно створити потік відповідного типу (ifstream, ofstream або fstream) і відкрити його у двійковому режимі з використанням прапорця ios::binary. Наприклад,

```
ofstream outFile("FileName.dat", ios::binary);
```

Читання і запис двійкових даних може здійснюватися *потоковими* засобами, *посимвольно* і *поблоково*. Найчастіше використовується блокове введення-виведення, яке служить для зчитування /запису заданої кількості елементів даних заданого розміру. Блоки бінарних даних можна читати /записувати за допомогою функцій-членів read() та write() класів ifstream та ofstream відповідно:

```
istream &read(unsigned char *buf, int num);
ostream &write(const unsigned char *buf, int num);
```

Функція read() читає num символів з асоційованого потоку і посилає їх в буфер buf; функція write() пише num символів в асоційований потік з буфера buf. Ці функції працюють з символами (байтами). Для них немає жодного значення, в який спосіб організовані і що собою являють дані, – вони просто переносять їх посимвольно (побайтово) з буфера до файлу і навпаки.

Якщо кінець файлу досягається до того, як буде прочитано задане число символів (байт), функція read() припиняє роботу. При цьому буфер містить стільки

символів, скільки було прочитано. Щоб визначити, скільки символів було прочитано, використовують функцію-член gcount(), що має наступний прототип:

```
int gcount();
```

Символьне введення-виведення здійснюється за допомогою перевантажених функцій get() і put(). При читанні символу (а не розширення символу), ці функції читають і записують байти.

Функція put() виводить (записує) одиночний символ у вихідний потік і повертає посилання на цей потік:

```
ostream &put(char ch);
```

Функція get(), що служить для введення (читання) окремого символу з вхідного потоку, перевантажена наступним чином:

```
int get();                  // повертає символ як своє значення
istream &get (char &c);     //передає символ і повертає посилання на свій потік
istream &get(char *buf, long len, char t = '\n');
```

Дві перші форми функції призначені для читання з потоку одиночного символу. Остання форма - витягує з потоку послідовність символів.

Хоча дані у бінарний файл записуються послідовно, у порядку їхнього надходження, оброблятися такі файли можуть двома способами: *послідовно* (доступ до елемента можливий тільки після перегляду усіх попередніх компонент файла) та за допомогою *прямого* доступу (доступ до елемента здійснюється безпосередньо за його номером).

Для реалізації прямого доступу у C++ використовуються два покажчика, асоційовані з потоками:

- get-покажчик зчитування, який задає місце у потоці, звідки буде вводитися наступна порція інформації;
- put-покажчик запису, який задає місце у потоці, куди буде виводитися наступна порція інформації.

При кожному введенні або виведенні відповідний вказівник послідовно просувається далі.

У C++-програмах позиціонування всередині файлу забезпечується функціями seekg() і seekp():

```
istream& seekg(long offset, seek_dir dir);
ostream& seekp(long offset, seek_dir dir);
```

Функція seekg() переміщує поточний get-покажчик на offset байт в напрямку, заданому dir. Функція seekp() переміщує на offset байт в напрямку, заданому dir, поточний put-покажчик. Параметр dir задає напрямок зсуву, який може приймати значення, визначені типом-перерахуванням seek_dir[1]:

ios :: beg – зсув від початку файла;

ios :: cur - зсув відносно поточної позиції;

ios :: end - зсув від кінця файла.

Якщо параметр dir відсутній, то переміщення покажчика здійснюється з початку файла.

Окрім безпосередньої установки файлового покажчика на задану позицію у файлі, можна отримати поточне значення позиції в потоці введення або виведення:

```
long tellg();
long tellp();
```

Вимоги до програми

1) Відповідність модульному підходу.
2) Виведення вхідних, проміжкових і вихідних даних.
3) Створення файлу програмним шляхом.
4) Можливість доповнення файлу.

[1] enum seek_dir { beg, cur, end };

Варіанти завдань

1. Створити файл з інформацією про працівників підприємства: ПІБ працівника, його дата народження (ДД.ММ.РРРР), стать (чоловік / жінка), посада. Створити новий файл, що містить список з працівників, які у поточному році виходять на пенсію. Видалити з нового списку (звільнити) працівників чоловічої статі, яким виповнилося 70 років, та жінок – які досягли 65-річного віку.

2. Створити файл з переліком продуктів у магазині: найменування, дата випуску, кінцевий термін придатності. Створити два нових файли, в один з яких помістити перелік продуктів, що швидко псуються (термін зберігання яких не більше 5-ти діб), в другий – продуктів тривалого зберігання. Видалити із створених файлів продукти, термін яких закінчився.

3. Створити файл з інформацією про наявність побутової техніки в офісі: найменування, дата покупки, термін гарантії (у днях). Визначити, чи є в офісі побутова техніка на гарантії. Перенести в інший файл інформацію про техніку, у якої закінчився термін гарантії.

4. Створити файл з інформацією про абітурієнтів: його ПІБ, дата народження (ДД.ММ.РРРР), стать (чоловік / жінка). Видалити зі списку абітурієнтів старше 35 років. Створити новий файл, до якого занести інформацію про юнаків призовного віку (від 18 до 27 років).

5. Створити файл з інформацією про телефонні переговори: номер телефону, початок та кінець переговорів (за шаблоном - ГГ:ХХ). З вихідного файлу створити два нових, в одному з яких мають бути зафіксовані денні переговори (з 6:00 до 20:00), а в іншому – нічні.

6. Створити файл з інформацією про товари меблевого магазину: назва товару, його вид (наприклад, стілець кухонний, стілець барний, стілець офісний, стіл кухонний, стіл комп’ютерний тощо), колір, ціна та наявна кількість. В новому бінарному файлі сформувати список усіх наявних у магазині стільців вказаного ви-

ду, їх кількість та вартість. Видалити з нового файлу інформацію про стільці вартістю від 300 до 500 грн.

7. Створити файл з розкладом руху приміських поїздів декількома напрямками (по кожному напрямку по 3-5 рейсів протягом дня): номер рейсу, напрямок руху, час відправлення, час прибуття в кінцевий пункт. На основі даного розкладу сформувати зимовий розклад (новий файл), в якому мають бути тільки ранкові (до 10:00) та вечірні (після 18:00) рейси.

8. Створити файл з інформацією про наявність лікарських ампул: назва, термін дії після відкриття (за шаблоном - ГГ:ХХ), термін зберігання (роки) нерозпакованої ампули. Створити новий файл, який містить інформацію про розкриті ампули: назва ліків, час відкриття (вводяться користувачем) та термін, до якого ампула має бути використана (розраховується згідно з даними з першого файлу). Видалити з нового файлу інформацію про ампули, термін зберігання яких у відкритому вигляді закінчився.

9. Створити файл із списком клієнтів перукарні на день: прізвище та ім'я клієнта, час (у форматі ГГ:ХХ) та передбачувана тривалість процедури. При створенні файлу перевіряти, чи не зайнятий час і чи достатньо у майстра вільного часу для виконання необхідної процедури. Вивести список усіх клієнтів, які прийдуть після 16:30.

10. Створити файл із списком пацієнтів, записаних на прийом до лікаря: прізвище пацієнта, дата попереднього відвідування лікаря та час, на який пацієнт записаний. Видалити з файлу записи про пацієнтів, час прийому яких минув. Створити два нових файли: в один занести відомості про вторинних хворих (попереднє відвідування яких було протягом 10-ти останніх днів), а в другий – про решту пацієнтів.

11. Створити файл із списком товарів у магазині: найменування, дата випуску, кінцевий термін придатності, ціна. Знизити ціну на 5% на всі товари, максимальний термін зберігання яких не перевищує двох тижнів. Підвищити ціну на 3% на товари, максимальний термін зберігання яких більш одного року. Максимальний

термін зберігання розрахувати як різницю кінцевого терміну придатності та дати випуску товару. Скопіювати в новий файл дані про товари, вироблені раніше поточного місяця.

12. Створити файл із списком справ на поточний день: умовна назва, час початку, передбачувана тривалість. Занести до нового файлу інформацію про справи, які потрібно виконати з 12.45 до 17.30. Вивести інформацію про вільний час (початок і кінець тимчасового проміжку та його тривалість).

13. Створити файл з інформацією про телефонні переговори: номер телефону, початок та кінець переговорів (за шаблоном - ГГ:ХХ). Розрахувати оплату за переговори, вважаючи, що хвилина розмови вдень (з 9:00 до 20:00) коштує 1,5 грн., а в вночі – 0,90 грн. Видалити з файлу дані про розмови тривалістю менше 3 хв.

14. Створити файл з розкладом руху міжміських автобусів: пункт призначення, час відправлення та тривалість поїздки (у годинах та хвилинах). Видалити з файлу інформацію про рейси, в яких хоча б частина шляху потрапляє на нічний час (з 23:00 до 6:00). Виняток становлять маршрути з тривалістю шляху понад 17 годин. Визначити час відправлення останнього автобуса в заданий пункт призначення.

15. Створити файл із списком працівників підприємства: ПІБ, дата народження, дата прийому на роботу. Вивести список співробітників, старших за 40 років, та співробітників, які працюють на підприємстві не менше 20 років. Видалити з файлу інформацію про співробітників, які працюють менше року.

16. Створити файл із списком співробітників підприємства: ПІБ працівника, його дата народження (ДД.ММ.РРРР), табельний номер, стать (чоловік / жінка). При додаванні даних виконувати перевірку на вік: він повинен бути не менше 20-ти та не більше 60-ти років. Створити два нових файли: перший містить список працівників, яким не більше 40 років, другий – список інших працівників.

17. Створити файл із списком лікарських ампул: назва, термін дії після відкриття (за шаблоном - ГГ:ХХ), термін зберігання нерозпакованої ампули (в роках). Видалити інформацію про ампули, термін зберігання яких перевищив один рік.

Видати попередження про закінчення терміну зберігання ампули за 10 днів.

18. Створити файл із переліком технічних перерв у роботі каси: час початку та час кінця перерви. При введенні даних перевіряти, чи не накладається нова перерва на вже наявну. Визначити, чи встигне касир обслужити N клієнтів (N ввести з клавіатури), які стоять у черзі, якщо на одного клієнта в середньому витрачається 15 хв.

19. Створити файл із списком творів автора: назва, дата написання, рік видання (якщо твір не видано, то - 0). Скопіюйте в окремий файл твори, написані протягом останніх чотирьох років. Вивести твори, що були надруковані більш ніж через 5 років після написання

20. Створити файл із списком телепередач: назва, час початку та час закінчення передачі. Визначити тривалість кожної передачі. Створити новий файл, що містить передачі, які відображаються в денний час (з 9:00 до 18:00).

21. Створити файл із списком автомобілів автосалону: назва, дата випуску, дата надходження у продаж. Створити список нових автомобілів (які надійшли у продаж не більш як через 2 місяці після випуску). Вивести інформацію про автомобілі, які були випущені не раніше вказаного року.

22. Створити файл із списком справ на поточний день: умовна назва, час початку, передбачувана тривалість. Визначити, яка справа за списком наступна (найближча до поточного часу). Створити файл з інформацією про вільний час у другій половині дня (після 13:00): початок та закінчення тимчасового проміжку та його тривалість (розрахувати).

23. Створити файл із списком клієнтів, обслужених менеджером протягом дня: прізвище, час приходу та час закінчення обслуговування. При введенні даних перевіряти їхню допустимість (чи не перетинаються клієнти з наявними). Створити новий файл, який містить інформацію про клієнтів, з якими менеджер спілкувався понад 30 хв.

24. Створити файл із списком працівників підприємства: прізвище, дата наро-

дження, дата прийому на роботу. Вивести список працівників, у яких день народження у поточному місяці та які пропрацювали на підприємстві не менше 5-ти років. Створити новий файл з інформацією про співробітників, які оформилися на роботу на дане підприємство у віці не старше 25-ти років та пропрацювали на ньому не менше 10-ти років.

25. Створити файл із списком товарів у магазині: найменування, дата випуску, кінцевий термін придатності, ціна. Перенести в інший файл дані про товари, у яких закінчується термін придатності (до закінчення залишилося менше 10% допустимого часу зберігання). Вивести інформацію про товари, вироблені за останні 10 днів.

26. Створити файл із списком клієнтів, обслужених менеджером протягом дня: прізвище клієнта, час приходу та час закінчення обслуговування клієнта. При введенні даних перевіряти їх допустимість (чи не перетинаються клієнти з наявними). Визначити, чи мав менеджер протягом робочого дня (з 9:00 до 17:00) вільний час (проміжки), і якщо так, то скільки і в якій половині дня.

27. Створити файл з інформацією про телефонні розмови: дата, час початку та час закінчення розмови. Визначити розмір оплати за розмови з урахуванням того, що в нічний час (з 22:00 до 7:00) дзвінки безкоштовні, у бізнес-час (з 9:00 до 18:00) вартість хвилини розмови 1,7 грн., а в інший час – на 30 % дешевше, ніж у бізнес-час.

28. Створити файл із списком автомобілів автосалону: назва, дата випуску, дата надходження у продаж. Створити список автомобілів, що надійшли у продаж за останній місяць. Вивести інформацію про вживані автомобілі (які були випущені більш ніж за рік до надходження у продаж).

29. Створити файл із списком покупців, які придбали товари зі знижкою на день акції: прізвище, стать, дата народження, кількість одиниць товару. Передбачається, що вартість однієї одиниці товару - 100 грн., знижка на товар дорівнює віку особи. Пенсіонерам (з 60-ти років) надається додаткова знижка 5%. Визначити виторг магазину за день. Створити новий файл з інформацією про покупців, які

придбали товару більш ніж на 250 грн.

30. Створити файл із розкладом занять у навчальному закладі протягом дня: назва предмета, час початку пари (за шаблоном ГГ:ХХ), час закінчення пари. Тривалість пари становить 1 год. 45 хв., а перерва між парами - не більше 45 хв. та не менше 5 хв. При введенні даних виконувати перевірку початку кожної пари (чи не накладається на попередню пару ти чи дотримані обмеження на тривалість перерви), а кінець пари розраховується автоматично.

31. Створити файл з інформацією про ВНЗ: назва, місто, в якому він знаходиться, рівень акредитації, кількість факультетів, кількість студентів кожного факультету. У вказаному місті визначити факультет, на якому навчається найбільша кількість студентів (із зазначенням відповідного ВНЗ). В новому файлі зберегти дані про ВНЗ III і IV рівня акредитації (назва ВНЗ, місто, в якому він знаходиться, рівень акредитації).

32. Створити файл із розкладом руху приміських поїздів декількома напрямками (по кожному напрямку по 3-5 рейсів протягом дня): номер рейсу, напрямок руху, час відправлення, час прибуття в кінцевий пункт. Зі складеного розкладу сформувати зимовий розклад (новий файл), в якому мають бути лише ранкові (до 10:00) та вечірні (після 18:00) рейси.

33. Створити файл із списком творів автора: назва, дата написання, рік видання (якщо твір не видано, то - 0). Визначити кількість зимових творів автора. Перенести в окремий файл інформацію про твори, написані та видані в минулому столітті.

34. Створити файл з інформацією про студентів: ПІБ, дата народження, форма навчання, група, середній рейтинг успішності за останню сесію. На кожному потоці (групи потоку мають однакову абревіатуру) визначити студентів з мінімальним середнім балом успішності і групи, в яких вони навчаються. В новому файлі сформувати відсортований за прізвищами список студентів-четверокурсників денної форми навчання, середній бал успішності яких не менше за вказаний.

35. Створити файл структур, що являє собою бібліотечний каталог і містить інформацію про книги, які є в наявності у бібліотеці: назва, автор (автори), рік видання, мова видання кількість екземплярів. Визначити кількість наявних екземплярів кожної книги вказаного автора. В новому файлі створити впорядкований за назвою каталог книг вказаного автора.

Контрольні питання

1) Особливості організації бінарних файлів у C/C++.
2) Чим відрізняються текстові файли від бінарних?
3) Чим схожі файли і масиви? Чим вони відрізняються?
4) Що таке файловий покажчик? Які стандартні функції змінюють його значення?
5) Як здійснюється навігація по файлу?
6) Чим відрізняється послідовний доступ до компонентів файлу від прямого доступу?
7) Чи можливий прямий доступ до елементів бінарного файлу?

Лабораторна робота 3
КЛАСИ ТА ОБ'ЄКТИ

Мета роботи - вивчити механізми створення і використання класів та об'єктів.

Теоретичні відомості

Класи і *об'єкти* є фундаментальними поняттями ООП. З абстрактної точки зору, *об'єкт* - це програмна модель реальної сутності предметної області задачі, яка характеризується певними властивостями і певною поведінкою (функціональністю). Сукупність об'єктів з ідентичною структурою і поведінкою, називають *класом*.

З точки зору мови програмування, *клас* – це тип даних користувача, який визначає як дані (поля), так і методи їхньої обробки; *об'єкт* - окремо взятий екземпляр деякого класу.

Реалізація класу у С++. Специфікація класу у С++ здебільшого складається з двох частин: *оголошення класу* (опису усіх його атрибутів і методів) і *визначення його методів* (тіл конкретних функцій).

Формат оголошення класу:

```
class  ім'я_класу
{ [ специфікатор_доступу_1 ] :  визначення_атрибутів;
визначення _методів;
специфікатор_доступу_2  :   визначення _атрибутів;
 визначення _методів;
      …
специфікатор_доступу_n  :  визначення _атрибутів;
визначення _методів;
};
```

Атрибути класу оголошуються так само, як і поля структури (запису):

```
тип  атрибут.
```

Атрибути класу можуть бути будь-якого типу, у тому числі і класом. Ініціалі-

зація атрибутів при оголошенні класу не допускається.

Оголошення методів класу здійснюється, як правило, за допомогою прототипів відповідних функцій.

Методи класу мають доступ до будь-яких атрибутів і методів даного класу. Однак, доступ до членів класу з об'єктів інших класів може бути обмеженим (за допомогою *специфікаторів доступу*):

- public *(відкритий/загальний)* – доступ з поза класу не обмежений;
- protected *(захищений)* – доступ можливий тільки власним методам і методам похідних класів;
- private *(закритий/приватний)* – доступ можливий тільки власним методам (прийнятий за умовчанням).

Дія будь-якого специфікатора поширюється до наступного специфікатора або до кінця класу. Можна задавати кілька секцій private, protected і public, порядок їх слідування значення не має.

Згідно принципу інкапсуляції, атрибути класу завжди повинні бути захищеними від несанкціонованого доступ (private). Доступ до них здійснюється тільки через методи цього ж класу. Тому методи, що реалізують інтерфейс класу із зовнішнім середовищем, повинні бути відкритими (public). Якщо який-небудь метод має допоміжне значення для інших методів класу, його слід описувати як закритий (private) або захищений (protected). Це забезпечує логічну цілісність інформації.

Кожен метод класу повинен бути визначений або безпосередньо (в тілі класу), або поза його межами (після тіла класу). Хороший стиль програмування передбачає визначення методів поза межами класу.

Оскільки різні класи можуть мати однойменні методи, при визначенні зовнішнього методу необхідно вказати до якого класу він належить (за допомогою оператора "::"):

```
тип_результату  ім'я_класу :: ім'я_методу ( [параметри] )
{    тіло_методу
};
```

Наприклад,

```
class  TPoint          // інформація про точку
{   int  x, y;         // кординати
public :
void  getX();          // виведення кординати X
void  getY();          // виведення кординати Y
void  setXY(int, int); // введення кординат  точки
};
void  TPoint :: getX()
{  cout << "X= " <<  x;
}
void  TPoint :: getY()
{  cout << "Y= " <<  y;
}
void TPoint:: setXY(int  xx,  int  yy)
{  x = xx;
 y = yy;
}
```

Хорошим стилем в C++ вважається винесення оголошення класу (його інтерфейсу) у окремий модуль - заголовний файл (.h-файл), який називають *файлом інтерфейсу* (або *інтерфейсом)*. При цьому в оголошенні класу, окрім атрибутів, повинні міститися тільки прототипи функцій. Визначення ж методів розміщується в окремому однойменному файлі з розширенням .cpp, який називають *файлом реалізації*. Така організація програми забезпечує незалежність всіх модулів, що використовують заголовний файл із оголошенням класу, від якихось змін у реалізації методів цього класу.

Визначення класу не призводить до виділення пам'яті. Пам'ять виділяється тільки тоді, коли визначається конкретна змінна "класового" типу, яку називають *екземпляром класу* або *об'єктом*.

Способи створення об'єкта залежать від наявності або відсутності в класі

конструктора - спеціального методу, що визначає необхідні для створення об'єкта дії. Зокрема, за відсутності конструктора і наявності protected- або private-атрибутів, можливе створення лише неініціалізованих об'єктів.

Створення об'єкта здійснюється оголошенням відповідної змінної (для *статичних* об'єктів) або динамічним розміщенням об'єкта в пам'яті (для *динамічних* об'єктів):

ім'я_класу ім'я_об'єкта; *// статичний об'єкт*

ім'я_класу * покажчик_на_об'єкт = new ім'я_класу *// динамічний об'єкт*

Наприклад, для оголошеного раніше класу TPoint визначення

TPoint point; *// статичний об'єкт класу TPoint*

виділяє у статичній пам'яті область, достатню для зберігання атрибутів класу TPoint, а визначення

TPoint *p_point = new TPoint; *// динамічний об'єкт класу TPoint*

- виділяє аналогічну область у динамічній пам'яті.

Способи ініціалізації атрибутів залежать від наявності / відсутності в класі конструктора, а також області видимості того чи іншого атрибуту.

Зокрема, за відсутності конструктора і protected- та private-атрибутів, оголошення ініціалізованих об'єктів здійснюється так же, як і оголошення ініціалізованих структур:

TPoint point = { 3, 4, 2 }.

Ініціалізуючі значення повинні перераховуватися в порядку задання атрибутів в оголошенні класу.

За наявності конструктора, він автоматично ініціалізує атрибути при створенні об'єкта.

Значення атрибутів неініціалізованих об'єктів можна задати і в процесі роботи програми: захищених і прихованих - тільки за допомогою методів класу, загальнодоступних - за допомогою методів класу або безпосереднім присвоєнням в програмі.

Доступ до членів класу визначається областю їх видимості відносно методу з якого здійснюється звернення:

– до члена класу із методів цього самого класу - безпосередньо за іменем атрибута чи метода;
– до public-членів класу ззовні (із методів інших класів або із зовнішніх функцій) - за допомогою імен об'єктів або вказівників на об'єкт.

Синтаксис звеpенення ззовні:

ім'я_об'єкта. ім'я_члена_класу

ім'я_покажчика_на_об'єкт -> ім'я_члена_класу.

Наприклад,

```
TPoint  point;
point.setXY (19, 26);
cout << "Координати: x - " << point.getX() << ", y - " << point.getY();
```

Конструктори. *Конструктор класу* – це спеціальний public-метод, який автоматично викликається при створенні об'єкта класу і виконує дії по виділенню пам'яті під об'єкт та, за необхідності, ініціалізацію його атрибутів.

Конструктор розпізнається по імені, яке у C++ обов'язково збігається з іменем класу. Він може мати аргументи, але не повертає жодного значення, навіть типу void. Наприклад,

```
class  TPoint
{      int  x, y;
public :
TPoint( );                  // оголошення конструктора
void getX();
void getY();
void setXY(int, int);
};
```

Якщо у класі явно не описаний жоден конструктор, компілятор автоматично генерує «порожній» конструктор (без параметрів та операторів). Такий конструктор не присвоює початкові значення атрибутам класу (хоча пам'ять під об'єкт виділяє).

Як і будь-яка інша функція з параметрами, конструктор може бути перевантажений, що дозволяє використовувати різні способи ініціалізації полів об'єктів.

Конструктор за замовчуванням - це конструктор, який не має параметрів і ініціалізує атрибути класу константними значеннями. Наприклад,

```
TPoint::TPoint()
{  x=0; y=0;
}
```

Конструктор з параметрами – це конструктор, який разом із створенням об'єкту присвоює його атрибутам деякі початкові значення, отримуючи їх як параметри. Наприклад,

```
TPoint::TPoint(int x1, int y1)
{  x=x1; y=y2;
}
```

При використанні конструктора з параметрами атрибути класу ініціалізуються кожен раз, як тільки створюється екземпляр класу. Наприклад, створення об’єкта point класу TPoint:

```
TPoint  point(5, 20, 3).
```

Особливістю конструктора з параметрами у C++ є те, що він не може бути використаний для ініціалізації атрибутів-констант чи атрибутів-посилань, оскільки їм не можуть бути присвоєні значення. Для таких випадків застосовується спеціальна конструкція, що включається в опис конструктора, - *список ініціалізації*:

```
 ім'я_класу (параметри) : список_ініціалізації
{  тіло_конструктора
}
```

Список ініціалізації в означенні конструктора має наступний формат:

ім'я_атрибута (ініціалізатори) [, ім'я_атрибута (ініціалізатори)],

де ініціалізатори визначають значення, які використовуються для ініціалізації зазначених полів об'єкта.

У разі використання списку ініціалізації тіло конструктора може виявитися порожнім (якщо всі атрибути ініціалізуються через список ініціалізації). Але і в

цьому випадку воно обов'язково має бути присутнім. Наприклад,

TPoint(): x(0), y(10) { }.

Конструктор ініціалізації може застосовуватися і для ініціалізації звичайних полів класу.

Реалізація класу у Python. Специфікація класу в Python має наступний формат:

```
class  ім'я_класу:
[рядок з описом класу]
[визначення атрибутів]
визначення методів
```

Всі члени класу, як правило, є загальнодоступними.

Конструктор має ім'я _ _init_ _ і може бути лише вбудованою функцією-членом. Формат його визначення:

```
def __init__(self):
    тіло_конструктора
```

Наприклад,

```
class Person:
    """Дані про особу"""
def __init__(self, Name):
    self.name = Name
def display(self):
    return 'Name: ' + self.name
```

Першим параметром будь-якого методу є параметр self - посилання на сам поточний об'єкт. Дане посилання передається автоматично і надає конкретному об'єкту доступ до атрибутів і методів класу.

Формат створення об'єктів:

```
ім'я_об'єкта = ім'я_класу([параметри]) .
```

Наприклад,

```
student = Person ('Ivanov')
print(student. display ())
```

Слід зазначити, що в Python прийняті наступні домовленості: ім’я, яке починається з символу верхнього регістра (наприклад, Person), позначає клас, а ім’я, записане в нижньому регістрі (наприклад, student), позначає окремий екземпляр, створений на основі класу.

В Python клас не є чимось статичним, тому додати атрибути до нього можна і після визначення:

student.salary = 1000

Вимоги до програми

1. Реалізація коду відповідно до модульного підходу.
2. Виведення усіх вихідних, проміжних та результуючих даних.

Варіанти завдань

1. Розробити клас ”трикутник на площині”, який заданий координатами своїх вершин. Створити масив об’єктів даного класу. Визначити трикутник з найбільшою площею.
2. Розробити клас ”опуклий чотирикутник на площині”, який заданий координатами своїх вершин. Створити масив об’єктів даного класу. Визначити чотирикутник з найбільшим периметром.
3. Розробити клас ”круг на площині”, який заданий радіусом та координатами центру. Створити масив об’єктів даного класу. Визначити круг з найбільшою площею.
4. Розробити клас ”тривимірний вектор”, який заданий своїми координатами. Створити масив об’єктів даного класу. Визначити вектор, який має найбільшу довжину.
5. Розробити клас ”кубічний многочлен” (многочлен виду $a_3x^3 + a_2x^2 + a_1x + a_0$), який заданий своїми коєфіцієнтами. Створити масив об’єктів даного

класу. Визначити многочлен, який приймає найменше мінімальне значення на введеному користувачем відрізку [a, b] із заданою точністю ε (використати метод простого перебору).

6. Розробити клас ”площина $Ax + By + Cz + D = 0$”, яка задана своїми коєфіцієнтами. Створити масив об’єктів даного класу. Визначити площини, яким належить введена користувачем точка.
7. Розробити клас ”тетраедр”, який заданий координатами своїх вершин в просторі. Створити масив об’єктів даного класу. Визначити тетраедр з найбільшим об’ємом.
8. Розробити клас ”конус”, який заданий координатами центру основи, координатами вершини та радіусом основи. Створити масив об’єктів даного класу. Визначити конус з найбільшою твірною.
9. Розробити клас ”Особа”, який характеризується ПІБ особи та її датою народження (у форматі ДД-ММ-РРРР). Створити масив об’єктів даного класу. Визначити людей, які народилися в щасливі дні (ділення суми цифр числа, місяця, року народження на 7 дають три однакові остачі).
10. Розробити клас ”пряма $ax + by + c = 0$”, яка задана своїми коефіцієнтами. Створити масив об’єктів даного класу. Визначити номери тих прямих, яким належить введена користувачем точка (її координати).
11. Розробити клас ”точка в просторі”, який заданий своїми координатами. Створити масив об’єктів даного класу. Визначити кількість точок з цілочисельними координатами, які попадають в перший октант.
12. Розробити клас ”Абонент”, який характеризується ПІБ абонента, його адресою і номером телефону (у форматі ХХХ-ХХХ-ХХ-ХХ). Створити масив об’єктів даного класу. Визначити абонента, у якого найбільша сума цифр телефону.
13. Розробити клас "сфера в просторі”, яка задається радіусом та координатами центру. Створити масив об’єктів даного класу. Визначити номери тих куль, в які попадає введена користувачем точка.
14. Розробити клас ”геометрична прогресія”, який задається першим членом a_0,

знаменником *q* та кількістю членів *n*. Створити масив об’єктів даного класу. Визначити прогресію (її номер), що має найбільший останній член.

15. Розробити клас ”матеріальна точка”, яка характеризується координатами початкового пололження та вектором рівномірної швидкості $\vec{\upsilon} = (\upsilon_1, \upsilon_2, \upsilon_3)$. Створити масив об’єктів даного класу. Визначити точки (їх номера), які за введений користувачем час *t* попадають в перший октант.
16. Розробити клас ”квадратне рівняння” (многочлен виду $ax^2 + bx + c = 0$), який заданий своїми коєфіцієнтами. Створити масив об’єктів даного класу. Визначити рівняння (їх номери), які не мають дійсних розв’язків.
17. Розробити клас ”арифметична прогресія”, який задається першим членом *a*, різницею *d* та кількістю членів *n*. Створити масив об’єктів даного класу. Визначити прогресію (її номер), що має найбільшу суму елементів.
18. Розробити клас ”функція *sin(ax+b)*”. Створити масив об’єктів даного класу. Визначити функцію (її номер), яка приймає найбільше значення у введеній користувачем точці *x*.
19. Розробити клас ”квадратна матриця” вказаної розмірності. Створити масив об’єктів даного класу. Визначити матрицю з найбільшою сумою діагональних елементів (найбільшим слідом).
20. Розробити клас ”числовий масив” вказаної розмірності. Створити масив об’єктів даного класу. Визначити масив з найменшим максимальним елементом.
21. Розробити клас ”слово”. Створити масив об’єктів даного класу. Визначити слово з найбільшою кількістю цифр.
22. Розробити клас ”Студент”, який характеризується ПІБ студента, номером його групи, датою народження (у форматі ММ-ДД-РРРР). Створити масив об’єктів даного класу. Визначити найстаршого студента вказаної групи (на вказану дату).
23. Розробити клас ”Працівник”, який характеризується ПІБ працівника, датою його народження (у форматі ММ-ДД-РРРР). Створити масив об’єктів даного класу. Визначити найстаршого працівника (на вказану дату).

24. Розробити клас ”Книга”, який характеризується ПІБ автора, назвою, роком видання, кількістю сторінок. Створити масив об’єктів даного класу. Визначити найтовщу книгу, видану у вказаний період часу (діапазон років).
25. Розробити клас ”Працівник”, який характеризується ПІБ працівника, датою його влаштування на роботу. Створити масив об’єктів даного класу. Визначити працівника, що має найбільший стаж (на вказану дату).
26. Розробити клас ”Поїзд”, який характеризується номером поїзда, пунктом призначення, часом його відправлення (у форматі ГГ-ХХ). Створити масив об’єктів даного класу. Визначити самий пізній поїзд (його номер), який відправляється в заданий пункт призначення.
27. Розробити клас ”Поїзд”, який характеризується номером поїзда, пунктом призначення, часом його відправлення (у форматі ГГ-ХХ). Створити масив об’єктів даного класу. Визначити поїзди (їх номери), які відправляються з вокзалу у заданий період часу (діапазон годин).
28. Розробити клас ”Продукт”, який характеризується найменуванням, датою випуску, кінцевим терміном придатності (у форматі ММ-ДД-РРРР). Створити масив об’єктів даного класу. Визначити продукти, термін яких закінчився (на вказану дату).
29. Розробити клас, який є абстракцією тексту та підтримує операції додавання рядка до тексту і визначення частоти появи у тексті заданого символу (кількість символів/загальна кількість символів). Створити масив об’єктів даного класу. Доповнити декілька текстів новими рядками. Визначити символ, який найчастіше зустрічається у текстах.
30. Розробити клас, який є абстракцією тексту та підтримує операції додавання рядка до тексту і пошук найдовшого рядка у тексті. Створити масив об’єктів даного класу. Доповнити декілька текстів новими рядками. Визначити найкоротший із найдовших рядків заданих текстів.
31. Розробити клас, який є абстракцією тексту та підтримує операції додавання рядка до тексту і визначення відсотка приголосних літер у тексті. Створити масив об’єктів даного класу. Доповнити декілька текстів новими рядками.

Визначити текст з найбільшим відсотком приголосних літер.

32. Розробити клас, який є абстракцією тексту та підтримує операції додавання рядка до тексту і визначення відсотка символів-цифр у тексті. Створити масив об’єктів даного класу. Доповнити декілька текстів новими рядками. Визначити текст з найменшим відсотком символів-цифр.
33. Розробити клас, який є абстракцією тексту та підтримує операції додавання рядка до тексту і визначення кількості подвоєних символів у тексті. Створити масив об’єктів даного класу. Доповнити декілька текстів новими рядками. Визначити текст з найбільшою кількістю подвоєних символів.
34. Розробити клас, який є абстракцією тексту та підтримує операції додавання рядка до тексту і визначення кількості голосних літер у тексті. Створити масив об’єктів даного класу. Доповнити декілька текстів новими рядками. Визначити текст з найбільшою кількістю голосних літер.
35. Розробити клас, який є абстракцією тексту та підтримує операції додавання рядка до тексту і визначення кількості рядків заданої довжини. Створити масив об’єктів даного класу. Доповнити декілька текстів новими рядками. Визначити текст з найбільшою кількістю рядків заданої довжини.

Контрольні питання

1. Що таке клас? Що відрізняє клас і об’єкт?
2. З яких частин складається опис класу?
3. Що таке конструктор та деструктор?
4. Наведіть приклади типів конструкторів.
5. Для чого використовується конструктор?
6. Для чого використовується деструктор?
7. Наведіть приклади використання методів.
8. Що таке конструктор за замовчуванням?
9. Що таке конструктор з параметрами?

Лабораторна робота 4
ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ

Мета роботи - вивчити механізми створення класів з використанням перевантажених операторів (операцій).

Теоретичні відомості

Перевантаження операторів – це можливість призначати новий сенс стандартним операторам при використанні їх з об'єктами певних класів. Використання механізму перевантаження операторів покращує читабельність програм і полегшує їх розуміння, виражаючи операції класу зрозумілішим чином. Перевантажуватися можуть не всі оператори (операції) мов програмування. Так, у C++ дозволені до перевантаження наступні оператори:

+	-	*	/	%	^	&	|	~	!	=
< >	+=	-=	*=	/=	%=	^=	&=	|=	<<	>>
>>=	<<=	==	!=	<=	>=	&&	||	++	--	->*
,	->	[]	()	new	new[]	delete	delete[]			

Не можна перевантажувати оператори: ″::" (дозволу області видимості), ″.″ (доступу до члена класу), ″?:″ (тернарний оператор), ″.*″ (доступу до розіменованого вказівника-члена класу). Крім того, не рекомендується перевантажувати логічні оператори && і ||, оскільки на їхні перевантажені версії не поширюється правило скорочених обчислень логічних виразів (якщо на деякому етапі значення усього виразу стає визначеним, подальші обчислення припиняються).

Основним механізмом перевантаження операторів у C++ є використання *операторних функцій*.

Операторна функція – це функція із спеціальним ім'ям operator@, де @ - символ операції, що перевизначається. Її прототип:

<тип_результату> operator@(<список_аргументів>);

Тіло операторної функції власне і визначає нову дію відповідного оператора. Тип значення, що повертається операторною функцією (<тип_результату>), хоча і може бути будь-яким, однак, частіше за все збігається з іменем класу, для якого оператор перевизначається. Операторні функції повинні мати прямий доступ до членів класу. Тому, вони можуть бути або членами класу, або дружніми функціями.

Операторна дружня функція класу визначається наступим прототипом:
friend <тип результату> operator@(<список аргументів>);

Операторні функції-члени і не члени класу (дружні функції) відрізняються між собою механізмом перевизначення: операторним функціям-членам аргумент передається в першу чергу неявно (через покажчик this), тоді як дружнім функціям усі аргументи передаються виключно явно (дружній функції вказівник this не передається).

Тому, при перевизначенні *унарних операторів* операторні функції-члени не мають явних параметрів (їм аргумент-операнд передається неявно), тоді, як дружнім функціям операнд передається явно.

У випадку перевантаження *бінарного оператора*, операторним функціям-членам передається один параметр, що задає другий операнд (вказаний праворуч від оператора). Перший операнд (вказаний зліва від оператора) передається такій функції неявно (через вказівник this). У дружню операторну функцію явно передаються обидва операнди бінарного оператора.

При перевизначенні операторів відношення та логічних операторів, операторні функції слід реалізовувати так, щоб вони повертали не об’єкт класу, для якого визначені, а ціле значення, яке інтерпретується як значення true або false. Це відповідає звичайному застосуванню цих операторів і дає змогу використовувати їх в умовних виразах.

Обмеження щодо застосування перевантажених операторів:

1) не можна змінювати пріоритет і асоціативність операторів (порядок виконання операцій одного пріоритету);
2) не можна змінювати кількість операндів оператора;

3) значення операндів оператора не можна задавати за замовчуванням;
4) хоча зміст оператора перевантажувати можна, його природу змінювати заборонено (унарні оператори не можна перевантажувати як бінарні і навпаки).

Приклад. Перевантаження арифметичних операції для класу комплексних чисел.

```
class TComplex {
double Re;
double Im;
public:
TComplex(double x, double y):Re(x), Im(y) { }
TComplex(TComplex& obj) {
Re = obj.Re; Im = obj.Im;
}
~TComplex() { }
void print(char *str);
TComplex  operator+(const TComplex obj);
friend TComplex operator+(float x, TComplex y);
TComplex  operator-(const TComplex obj);
TComplex  operator*(const TComplex obj);
TComplex  operator~();
TComplex  operator/(TComplex obj);
double  modul2(const TComplex obj);
};
int main() {
TComplex ObjA(1,1), ObjB(2,2), ObjC(0,0), ObjD(0,0);
ObjC=ObjA+ObjB;          // додавання об'єктів ObjA і ObjB
ObjC.print("ObjC");
ObjC=2.5+ObjC;
ObjC.print("ObjC");
```

```
ObjD=ObjC*ObjA/ObjB;
ObjD.print(“ObjD”);
cin.get();
}
void TComplex::print(char *str) {
printf("%s=%lf + i*%lf\n", str, Re, Im);
}
TComplex TComplex::operator+(const TComplex obj) {
TComplex tmp (0,0);
tmp.Re = Re+obj.Re;
tmp.Im = Im+obj.Im;
return tmp;
 }
TComplex TComplex::operator-(const TComplex obj) {
TComplex tmp(0,0);
 tmp.Re = Re-obj.Re;
 tmp.Im = Im-obj.Im;
return tmp;
 }
 TComplex TComplex::operator*(const TComplex obj) {
TComplex tmp(0,0);
tmp.Re = Re*obj.Re-Im*obj.Im;
tmp.Im = Re*obj.Im+Im*obj.Re;
return tmp;
 }
 TComplex TComplex::operator~() {     // спряжене число
TComplex tmp(0,0);
tmp.Re = Re;
tmp.Im = -Im;
return tmp;
```

```
 }
TComplex TComplex::operator/(TComplex obj) {
TComplex tmp(0,0);
tmp=(*this)*(~obj);
tmp.Re = tmp.Re/modul2(obj);
tmp.Im = tmp.Im/modul2(obj);
return tmp;
 }
 double TComplex::modul2(const TComplex obj) {
return obj.Re*obj.Re+obj.Im*obj.Im;
 }
TComplex operator+(float x, TComplex y) {
TComplex  tmp(0,0);
tmp.Re = x+y.Re;
tmp.Im = x+y.Im;
return tmp;
}
```

У Python для перевизначення операторів використовується спеціальні методи (табл. 3). Імена таких методів обмежують подвійним підкресленням “__”.

Таблиця 3. Деякі оператори і відповідні їм спеціальні функції Python

Операція	Синтаксис	Функція
Арифметичні		
заперечення	-a	__neg__(a)
додавання	a + b	__add__(a, b)
віднімання	a - b	__sub__(a, b)
множення	a * b	__mul__(a, b)
ділення	a / b	__truediv__(a, b)
цілочисельне ділення	a // b	__floordiv__(a, b)
залишок від ділення	a % b	__mod__(a, b)
піднесення до степеня	a ** b	__pow__(a, b)
Індексація і зрізи		
доступ за індексом	obj[k]	__getitem__(obj, k)

Операція	Синтаксис	Функція
присвоєння за індексом	obj[k] = v	__setitem__(obj, k, v)
видалення за індексом	del obj[k]	__delitem__(obj, k)
зріз	seq[i:j]	__getitem__(seq, slice(i, j))
присвоєння зрізу	seq[i:j] = values	__setitem__(seq, slice(i, j), values)
видалення зрізу	del seq[i:j]	__delitem__(seq, slice(i, j))
конкатенація	seq1 + seq2	__concat__(seq1, seq2)
Ідентифікація та порівняння		
ідентифікація	a is b	__is__(a, b)
	a is not b	__is_not__(a, b)
перевірка на входження	obj in seq	__contains__(seq, obj)
перетворення в логічний тип	bool(obj)	__truth__(obj)
рівність	a == b	__eq__(a, b)
нерівність	a != b	__ne__(a, b)
порівняння	a < b	__lt__(a, b)
	a > b	__gt__(a, b)
	a <= b	__le__(a, b)
	a >= b	__ge__(a, b)

Наприклад,

```
class Point2D:
    """Точка на площині"""
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __str__(self):
        """Повернути рядок у вигляді 'Точка 2D (x, y)' """
        return "Точка 2D ({}, {})".format(self.x, self.y)
    def __add__(self, other):
        """Створити новий об’єкт як суму координат 'self' и 'other' """
        return Point2D(self.x + other.x, self.y + other.y)
    def __sub__(self, other):
```

```
        """ Створити новий об’єкт як різницю координат 'self' и 'other' """
        return Point2D(self.x - other.x, self.y - other.y)
    def __neg__(self):
        """ Повернути новий об’єкт, інвертувавши координати"""
        return Point2D(-self.x, -self.y)
    def __eq__(self, other):
        """ Повернути відповідь, чи є точки однаковими"""
        return self.x == other.x and self.y == other.y
    def __ne__(self, other):
        """ Повернути відповідь, чи є точки різними"""
        return not (self == other)
p1 = Point2D(0, 5)
p2 = Point2D(-5, 10)
print(p1 + p2)              # Точка 2D (-5, 15)
print(p1 - p2)              # Точка 2D (5, -5)
print(-p2)            # Точка 2D (5, -10)
print(p1 == p2, p1 != p2)   # False True
```

Вимоги до програми

1 Реалізація коду відповідно до модульного підходу.

2 Виведення усіх вхідних, проміжних і вихідних даних.

Варіанти завдань

1. Розробити клас "Вектор", який задається полярними координатами (довжина і кут). Реалізувати для нього декілька конструкторів, геттери, метод повороту вектору на будь-який кут. Перевантажити оператори "+" та "/" для знаходження суми векторів і "зменшення" вектору відповідно. Створити три вектори (B1, B2, B3), використовуючи різні конструктори. Повернути вектор B3 на 45°, вектор

B2 "зменшити" у 2 рази. Визначити вектор B1 як суму змінених векторів B2 та B3.

2. Визначити клас "Пряма на площині", який задається коефіцієнтами рівняння прямої ($ax+by+c=0$). Реалізувати для нього декілька конструкторів, геттери, метод обчислення координат точок перетину прямої із осями OX і OY. Перевантажити оператори: префіксний "++" - для повороту прямої (зміни кутового коефіцієнта) на 1 градус за годинниковою стрілкою; "||" - для перевірки паралельності прямих. Створити три прямих (P1, P2, P3), використовуючи різні конструктори. Перевірити, чи паралельні між собою прямі P1 та P2. Повернути пряму P3 на 1 градус за годинниковою стрілкою. Визначити координат точок перетину прямої P3 із осями OX і OY.
3. Визначити клас "Час" для роботи із часом в межах доби. Членами класу є години, хвилини та секунди. Реалізувати для нього декілька конструкторів, геттери, метод обчислення часу, що залишився до завершення доби. Перевантажити оператори: префіксний "+=" – для збільшення кількості хвилин на вказану величину, "-" – для знаходження тривалості часу між двома моментами часу. Створити три об’єкта часу (T1, T2, T3), використовуючи різні конструктори. Збільшити час T1 на 17 хвилин, а час T2 – на 34 хвилини. Визначити тривалість часу між моментами часу T1 і T2. Для часу T3 визначити час, що залишився до завершення доби.
4. Побудувати клас "Булевий вектор" розмірності n (складається з булевих констант – 0 і 1). Реалізувати для нього декілька конструкторів, геттери, метод знаходження його ваги (кількості компонент, рівних 1). Перевантажити оператори кон’юнкції ("&") та інверсії (“~”) компонент булевих векторів. Створити три булеві вектори (B1, B2, B3), використовуючи різні конструктори. Знайти доповнення вектора B1. Визначити вектор B3 як кон’юнкцію зміненого вектора B1 та вектора B2 (B3 = B1∧ B2). Обчислити вагу отриманого вектора B3.
5. Визначити клас "Квадратна матриця" розмірності n. Реалізувати для нього декілька конструкторів, геттери, метод обчислення норми матриці. Перевантажити оператори додавання "+" і множення "*"матриць. Створити три матриці (M1,

M2, M3), використовуючи різні конструктори. Визначити матрицю M3 як суму матриць M1 та M2. Отриману матрицю M3 піднести до квадрату. Знайти норму нової матриці M3.

6. Визначити клас "Відрізок", який задається координатами початку та кінця відрізка. Реалізувати для нього декілька конструкторів, геттери, метод перевірки приналежності заданої точки відрізку. Перевантажити оператори: +" - для додавання відрізків згідно правил додавання векторів, постфіксний "++" - для збільшення координат кінця відрізка на 1. Створити три відрізка (V1, V2, V3), використовуючи різні конструктори. Визначити відрізок V3 як суму відрізків V1 та V2. Збільшити координати кінця відрізка V3 на 1. Перевірити, чи належить задана точка відрізку V3.
7. Визначити клас "Дата" для роботи із датами в межах року. Членами класу є число, місяць та рік. Реалізувати для нього декілька конструкторів, геттери, метод визначення пори року, що відповідає вказаній даті. Перевантажити оператори: "+=" – для збільшення дати на вказану кількість днів, "-" – для знаходження інтервалу між двома датами. Створити три об’єкта-дати (D1, D2, D3), використовуючи різні конструктори. Збільшити дату D1 на 9 днів, а дату D2 – на 14 днів. Визначити тривалість інтервалу між датами D1 і D2. Для дати D3 визначити пору року, якій відповідає ця дата.
8. Визначити клас "Numeral_16", членом якого є шістнадцяткове число. Реалізувати для даного класу декілька конструкторів, геттери, метод перетворення числа у десяткове. Перевантажити оператори: постфіксний "++" – для інкрементації шістнадцяткового числа, "+=" – для збільшення його на вказану величину, "+" – для додавання двох шістнадцяткових чисел. Створити три шістнадцяткових числа (N1, N2, N3), використовуючи різні конструктори. Інкрементувати число N1, а число N2 збільшити на вказану величину. Знайти суму змінених чисел N1 та N2 і зберегти її в N3. Перевести отримане значення N3 у десятковий формат.
9. Визначити клас "Багаточлен" ступеня 3, членами якого є коефіцієнти полінома. Реалізувати для нього декілька конструкторів, геттери, метод обчислення значення поліному в заданій точці. Перевантажити оператори додавання "+" і

множення "*" поліномів. Створити три поліноми (P1, P2, P3), використовуючи різні конструктори. Визначити новий поліном P4 як суму поліномів P1 та P2 і новий поліном P5 як добуток поліномів P2 та P3. Обчислити значення поліномів P4 і P5 в заданій точці.

10. Розробити клас "Вектор на площині", який задається координатами його кінця. Реалізувати для нього декілька конструкторів, геттери, метод обчислення довжини вектору. Перевантажити оператори "-" та "*" для знаходження різниці векторів і "збільшення" вектору відповідно. Створити три вектори (V1, V2, V3), використовуючи різні конструктори. Вектор V1 "збільшити" у 2 рази. Визначити вектор V3 як різницю зміненого вектору V1 та вектору V2. Знайти довжину вектору V3.
11. Визначити клас "Точка в просторі", членами якого є координати точки в циліндричній системі координат. Реалізувати для нього декілька конструкторів, геттери, метод обчислення відстані від точки до початку системи координат. Перевантажити оператори: префіксний "++" – для збільшення азимутального кута на 1°, "+=" - для збільшення радіальної відстані ρ на вказану величину, "==" – для визначення рівності відповідних координат двох точок. Створити три точки (P1, P2, P3), використовуючи різні конструктори. Інкрементувати азимутальний кут точки P1, а радіальну відстань ρ точки P2 збільшити на вказану величину. З'ясувати, чи рівні між собою відповідні координати цих двох точок (P1 і P2). Визначити відстань від точки P3 до початку системи координат.
12. Визначити клас "Numeral_8", членом якого є вісімкове число. Реалізувати для даного класу декілька конструкторів, геттери, методи перетворення числа у десяткове. Перевантажити оператори: префіксний "++" – для інкрементації вісімкового числа, "+=" – для збільшення його на вказану величину, "+" – для додавання двох вісімкових чисел. Створити три вісімкових числа (N1, N2, N3), використовуючи різні конструктори. Інкрементувати число N1, а число N2 збільшити на вказану величину. Знайти суму змінених чисел N1 та N2 і зберегти її в N3. Перевести отримане значення N3 у десятковий формат.
13. Визначити клас "Число", членами якого є його розряди (кількість одиниць,

десятків, сотень та тисяч). Реалізувати для нього декілька конструкторів, геттери, метод обчислення числа (отримання десяткового еквівалента). Перевантажити оператори: префіксний "++" / префіксний "--" - для інкрементування / декрементування усіх розрядів числа, "+" – для додавання двох чисел, , заданих своїми розрядами, ">" - для знаходження більшого із двох таких чисел. Створити чотири числа (N1, N2, N3, N4), використовуючи різні конструктори. Інкрементувати число N1, а число N2 декрементувати. Знайти суму змінених чисел N1 та N2 і зберегти її в N3. Знайти більше із чисел N3 і N4. Отримати його десятковий еквівалент.

14. Розробити клас «Множина» для представлення множини цілих чисел. Реалізувати для нього декілька конструкторів, геттери, метод визначення приналежності заданого елемента множині. Перевантажити оператори "+", "*" та "-" для знаходження об'єднання, перетину і різниці множин відповідно. Створити дві множини (A, B), використовуючи різні конструктори. На їх основі побудувати множину C=(A∪B)\(A∩B).

15. Визначити клас "Коло", членами якого є радіус кола та координати його центру. Реалізувати для нього декілька конструкторів, геттери, метод обчислення довжини кола. Перевантажити оператори: префіксний "++" / постфіксний "++" - для інкрементування x-координати і y-координати центру кола відповідно, "*" – для збільшення радіусу кола у вказану кількість разів (ціле число). Створити три кола (C1, C2, C3), використовуючи різні конструктори. Інкрементувати x-координату кола C1 і y-координату кола C2. Збільшити радіус кола C3 у 3 рази. Серед кіл C1, C2, C3 визначити коло найбільшої довжини.

16. Побудувати клас BoolVector, що представляє булевий вектор (складається з булевих констант – 0 і 1). Реалізувати для нього декілька конструкторів, геттери, метод перевірки, чи є даний булевий вектор попередником іншого (булевий вектор $\alpha=a_1a_2\ldots a_n$ є попередником булевого вектора $\beta=b_1b_2\ldots b_n$, якщо для будь-якого i = 1, 2, …, n виконується умова $a_i < b_i$). Перевантажити оператори диз'юнкції ("|") та виключаюче АБО ("^") для компонент булевих векторів. Створити чотири булеві вектори (V1, V2, V3, V4), використовуючи різні конс-

труктори. Визначити вектор V3 як диз'юнкцію булевих векторів V1 та V2 (V3 = V1 ∨ V2), а вектор V4 – як результат застосування операції "^"до булевих векторів V1 і V3 (V4 =V3 ^ V1). З'ясувати, чи є булевий вектор V4 попередником булевого вектора V3.

17. Визначити клас "Комплексне число", який задає комплексні числа в показниковій формі $\rho=e^{i\varphi}$. Членами класу є модуль ρ та аргумент φ комплексного числа. Реалізувати для нього декілька конструкторів, геттери, метод переведення числа у алгебраїчну форму. Перевантажити оператори: префіксний "++" / постфіксний "++" - для інкрементування дійсної і уявної частини числа відповідно, "+" – для обчислення суми двох комплексних чисел. Створити три комплексні числа (К1, К2, К3), використовуючи різні конструктори. Інкрементувати дійсну частину числа К1 і уявну частину числа К2. Знайти суму змінених чисел К1 і К2 та зберегти її в К3. Перевести число К3 у алгебраїчну форму.
18. Визначити клас "Numeral_8", членом якого є вісімкове число. Реалізувати для даного класу декілька конструкторів, геттери, методи перетворення числа у двійкове, у тому числі і скороченим способом. Перевантажити оператори: префіксний "++" – для інкрементації вісімкового числа, "+=" – для збільшення його на вказану величину, "+" – для додавання двох вісімкових чисел. Створити три вісімкових числа (N1, N2, N3), використовуючи різні конструктори. Інкрементувати число N1, а число N2 збільшити на вказану величину. Знайти суму змінених чисел N1 та N2 і зберегти її в N3. Перевести отримане значення N3 у двійковий формат двома способами (звичайним і скороченим).
19. Визначити клас "Квадратна матриця 3×3". Реалізувати для нього декілька конструкторів, геттери, метод обчислення визначника матриці. Перевантажити оператори множення "*" матриць та інкрементації її елементів "++". Створити три матриці (М1, М2, М3), використовуючи різні конструктори. Визначити матрицю М3 як добуток матриць М1 та М2. Інкрементувати елементи отриманої матриці М3. Знайти визначник зміненої матриці М3.
20. Визначити клас "Трикутник", членами якого є сторони трикутника в просторі. Реалізувати для нього декілька конструкторів, геттери, методи обчислення

площі трикутника. Перевантажити оператори: "++" / "--" - для інкрементування / декрементування довжин сторін трикутника відповідно, "+=" / "-=" – для збільшення / зменшення довжин сторін трикутника на вказану величину. Створити три трикутника (T1, T2, T3), використовуючи різні конструктори. Інкрементувати довжини сторін трикутника T1, а довжини сторін трикутника T2 декрементувати. Збільшити довжини сторін трикутника T3 на вказану величину. Серед трикутників T1, T2, T3 визначити трикутник, що має найбільшу площу.

21. Визначити клас "Поліном" розмірності 4, членами якого є коефіцієнти полінома. Реалізувати для нього декілька конструкторів, геттери, метод обчислення значення поліному в заданій точці. Перевантажити оператори: "+=" / "-=" - для збільшення / зменшення усіх коефіцієнтів полінома на вказану величину, "==" – для визначення рівності відповідних коефіцієнтів двох поліномів. Створити три полінома (P1, P2, P3), використовуючи різні конструктори. Коефіцієнти полінома P1 збільшити на вказану величину, а полінома P2 - зменшити на цю ж саму величину. З'ясувати, чи рівні між собою відповідні коефіцієнти поліномів P1 і P2. Обчислити значення поліному P3 в заданій точці.

22. Розробити клас "Вектор у просторі", який задається координатами його кінця. Реалізувати для нього декілька конструкторів, геттери, метод обчислення довжини вектора. Перевантажити оператори "+" та "*" для знаходження суми і скалярного добутку векторів відповідно. Створити три вектори (M1, M2, M3), використовуючи різні конструктори. Визначити вектор M3 як суму векторів M1 та M2. Знайти довжину вектору M3, а також скалярний добуток векторів M1 та M3.

23. Визначити клас "Дата" для роботи із датами в межах року. Членами класу є число, місяць та рік. Реалізувати для нього декілька конструкторів, геттери, метод визначення терміну, що залишився до кінця року. Перевантажити оператори: префіксний "++" – для збільшення кількості місяців на 1, постфіксний "++" - для збільшення кількості днів на 1; ">" – для порівняння дат. Створити три об'єкта-дати (D1, D2, D3), використовуючи різні конструктори. Збільшити дату D1 на 1 місяць, а дату D2 – на 1 день. З'ясувати, яка із цих дат (D1 чи D2) більш

пізня. Для дати D3 визначити термін, що залишився до кінця року.

24. Визначити клас "Булева матриця" (BoolMatrix) розмірності n×m. Реалізувати для нього декілька конструкторів, геттери, метод підрахунку числа одиниць у матриці. Перевантажити оператори диз'юнкції ("|") та інверсії ("~") компонент матриць. Створити три булеві матриці (M1, M2, M3), використовуючи різні конструктори. Визначити матрицю M3 як диз'юнкцію булевих матриць M1 та M2 (M3 = M1 ∨ M2). Знайти інверсію матриці M3. У отриманій матриці M3 підрахувати число одиниць.

25. Визначити клас "Відрізок", який задається координатами початку та кінця відрізка. Реалізувати для нього декілька конструкторів, геттери, метод обчислення довжини відрізка. Перевантажити оператори: префіксний "++" - для збільшення координат початку відрізка на 1, "||" - для перевірки паралельності відрізків. Створити три відрізка (B1, B2, B3), використовуючи різні конструктори. Перевірити, чи паралельні між собою відрізки B1 та B2. Збільшити координати початку відрізка B3 на 1. Знайти довжину отриманого відрізка B3.

26. Визначити клас "Час" для роботи із часом в межах доби. Членами класу є години, хвилини та секунди. Реалізувати для нього декілька конструкторів, геттери, метод обчислення часу, що залишився до вказаного моменту часу. Перевантажити оператори: префіксний "++" – для збільшення кількості хвилин на 1, постфіксний "++" - для збільшення кількості секунд на 1; ">" – для порівняння моментів часу. Створити три об'єкта часу (T1, T2, T3), використовуючи різні конструктори. Збільшити час T1 на 1 хвилину, а час T2 – на 1 секунду. З'ясувати, який із цих моментів часу (T1 чи T2) є більшим. Для часу T3 визначити час, що залишився до заданого.

27. Визначити клас "Точка", який задається координатами точки на площині. Реалізувати для нього декілька конструкторів, геттери, метод визначення квадранта системи координат (його номера), в якому знаходиться дана точка. Перевантажити оператори: префіксний "++" – для збільшення x-координати точки на 1, постфіксний "++" – для збільшення y–координати точки на 1, "-" – для визначення відстані між двома точками. Створити три точки (T1, T2, T3), викори-

стовуючи різні конструктори. Інкрементувати x-координату точки T1 і y–координату точки T2. Визначити відстань між отриманими точками T1 і T2. З’ясувати, якому квадранту належить точка T3.

28. Визначити клас "Numeral_16", членом якого є шістнадцяткове число. Реалізувати для даного класу декілька конструкторів, геттери, методи перетворення числа у двійкове, у тому числі і скороченим способом. Перевантажити оператори: префіксний "++" – для інкрементації шістнадцяткового числа, "+=" – для збільшення його на вказану величину, "+" – для додавання двох шістнадцяткових чисел. Створити три шістнадцяткових числа (N1, N2, N3), використовуючи різні конструктори. Інкрементувати число N1, а число N2 збільшити на вказану величину. Знайти суму змінених чисел N1 та N2 і зберегти її в N3. Перевести отримане значення N3 у двійковий формат двома способами (звичайним і скороченим).

29. Розробити клас "Вектор", членами якого є сферичні координати вектора у просторі. Реалізувати для нього декілька конструкторів, геттери, метод обчислення його декартових координат. Перевантажити оператори: "+=" – для збільшення полярного кута на вказану величину (у градусах), "!=" – для перевірки неколінеарності векторів. Створити три вектори (B1, B2, B3), використовуючи різні конструктори. Збільшити полярний кут вектора B1 на вказану величину. Знайти декартові координати зміненого вектора B1. Перевірити колінеарність векторів B2 і B3.

30. Визначити клас "Roman_numerals", який задає число в римській формі запису (у вигляді рядка). Реалізувати для даного класу декілька конструкторів, геттери, методи перетворення числа у десяткове. Перевантажити оператори: префіксний "++" – для інкрементації даного числа, "+=" – для збільшення його на вказану величину (римське число), "+" – для додавання двох римських чисел. Створити три римських числа (R1, R2, R3), використовуючи різні конструктори. Інкрементувати число R1, а число R2 збільшити на вказану величину. Знайти суму змінених чисел R1 та R2 і зберегти її в R3. Перевести отримане значення N3 у десятковий формат.

31. Розробити клас ««Мультимножина» для представлення множини символів. Реалізувати для нього декілька конструкторів, геттери, метод визначення приналежності заданого елемента множині. Перевантажити оператори "+", "*" та "-" для знаходження об'єднання, перетину і різниці множин відповідно. Створити три множини (A, B, C), використовуючи різні конструктори. На їх основі побудувати множину D=(A∪B)\ C∩B.
32. Визначити клас "Трикутник", членами якого є координати вершин трикутника в просторі. Реалізувати для нього декілька конструкторів, геттери, методи обчислення периметра трикутника. Перевантажити оператори: префіксний "++" / постфіксний "++" - для інкрементування усіх x-координат і усіх y-координат вершин трикутника відповідно, "+=" – для збільшення усіх координат вершин трикутника на вказану величину. Створити три трикутника (T1, T2, T3), використовуючи різні конструктори. Інкрементувати x-координати вершин трикутника T1 і y-координати вершин трикутника T2. Збільшити координати вершин трикутника T3 на вказану величину. Серед трикутників T1, T2, T3 визначити трикутник, що має найбільший периметр.
33. Визначити клас "Numeral_2", членом якого є двійкове число. Реалізувати для даного класу декілька конструкторів, геттери, методи перетворення числа у десяткове. Перевантажити оператори: префіксний "++" – для інкрементації двійкового числа, "+=" – для збільшення його на вказану величину, "+" – для додавання двох двійкових чисел. Створити три двійкових числа (N1, N2, N3), використовуючи різні конструктори. Інкрементувати число N1, а число N2 збільшити на вказану величину. Знайти суму змінених чисел N1 та N2 і зберегти її в N3. Перевести отримане значення N3 у десятковий формат.
34. Визначити клас "Кільце", членами якого є внутрішній і зовнішній радіуси кільця та координати його центру. Реалізувати для нього декілька конструкторів, геттери, метод обчислення товщини кільця. Перевантажити оператори: префіксний "++" / постфіксний "++" - для інкрементування величин внутрішнього і зовнішнього радіусів кільця відповідно, "*=" – для збільшення зовнішнього радіусу кільця у вказану кількість разів (ціле число). Створити три кільця

(C1, C2, C3), використовуючи різні конструктори. Інкрементувати величину внутрішнього радіусу кільця C1 і зовнішнього радіусу кільця C2. Збільшити зовнішній радіус кільця C3 у 2 рази. Серед кілець C1, C2, C3 визначити кільце найбільшої товщини.

35. Визначити клас "Квадратна матриця 3×3". Реалізувати для нього декілька конструкторів, геттери, метод обчислення визначника матриці. Перевантажити оператори множення "*" матриць та інкрементації її елементів "++". Створити три матриці (M1, M2, M3), використовуючи різні конструктори. Визначити матрицю M3 як добуток матриць M1 та M2. Інкрементувати елементи отриманої матриці M3. Знайти визначник зміненої матриці M3.

Контрольні питання

1. Що означає перевантажити оператор чи операцію?
2. Наведіть приклади перевантаження унарних операцій.
3. Наведіть приклади перевантаження бінарних операцій та операторів.
4. Покажіть, як використовуються перевантажені операції та оператори.
5. Поясніть, у чому відмінність перевантаження операцій у мовах C++ та C#.
6. Поясніть, шо таке дружня функція, її призначення та відмінність від звичайної функції.
7. Поясніть випадки перевантаження операцій за допомогою дружніх функцій.
8. Поясніть, завдяки чому у C# на відміну від C++ немає потреби у дружніх функціях для перевантаження операцій.
9. Поясніть на наведіть приклад перевантаження оператора приведення типів у мовах C++ та C#.
10. Наведіть обмеження на перевантаження операцій у мовах C++ та C#.

Лабораторна робота 5

УСПАДКУВАННЯ ТА ПОЛІМОРФІЗМ

Мета роботи - вивчити механізми створення і використання класів та об'єктів.

Теоретичні відомості

Поняття успадкування

Успадкування – це механізм, за допомогою якого один клас може успадковувати характеристики і функціональність іншого. Спадкування дозволяє будувати ієрархію класів, переходячи від загальних до більш спеціалізованих класів.

Клас, що лежить в основі ієрархії, називають *базовим (класом-предком)*; класи, що успадковують властивості базового класу, - *похідними* (*класами-нащадками*). Похідні класи, у свою чергу, можуть бути базовими для своїх спадкоємців, що в результаті приводить до ланцюжка успадкування.

Таке впорядкування дозволяє легко справлятися зі складністю розроблюваних програм, зробити логіку їх роботи більш простою та зрозумілою.

Механізм успадкуванням передбачає конструювання нових класів-нащадків з вже наявних класів-предків шляхом автоматичного успадкування членів класу-предка (окрім конструкторів) і додавання, за необхідності, власних атрибутів та методів.

При успадкуванні за конструювання (виділення пам’яті та ініціалізацію) успадкованої частини похідного класу відповідає конструктор класу-предка, а за власну частину − конструктор класу-нащадка. Тому будь-який конструктор класу-нащадка повинен забезпечити коректне конструювання як базової частини, так і власної. При цьому слід пам’ятати, що спочатку конструюється успадкована частина. Для цього за замовчуванням викликається або конструктор за замовчуванням класу-предка (якщо у класі-предка не визначено жодного конструктора) або конструктор без параметрів класу-предка (якщо такий описано у класі-предку). Якщо у класі-предка описано конструктори з параметрами, але при цьому не описано конструктор без параметрів, то явний ви-

клик конструктора класу-предка є обов’язковим.

Під час руйнування об'єкта класу-нащадка спочатку викликається його "рідний" конструктор, а за ним – конструктор класу-предка.

При розширеній ієрархії класів застосовується загальне правило: конструктори викликаються у порядку породження класів, а деструктори – у зворотному порядку.

Поняття поліморфізму

При успадкуванні класів може виявитися, що деяка функціональність об’єктів, зберігаючи назву, змінюється по суті. Наприклад, знаходження площі прямокутника (клас-предок Rectangle) і площі повної поверхні прямокутного паралелепіпеда (клас-нащадок Parallelepiped). Як у класі предка, так і у похідному класі їх доцільно назвати Square.

Сукупність механізмів, що дозволяють використовувати однойменні методи, яким може відповідати різний програмний код, називають ***поліморфізмом.***

Часто поліморфізм характеризують фразою "один інтерфейс — багато реалізацій". Це означає, що до групи логічно зв’язаних однойменних методів можна отримати доступ одним і тим самим способом, незважаючи на можливу відмінність у конкретних діях кожного із методів.

Поліморфізм допомагає спростити програму, дозволяючи використовувати один і той же інтерфейс для опису схожих дій.

В ООП розрізняють два основних види поліморфізму:

1) *Спеціальний (простий, ad hoc) поліморфізм* - передбачає для різних класів спеціальну (власну) реалізацію поліморфного методу з можливістю використання різних прототипів.

Підтримується через механізми *раннього* (*статичного*) *зв’язування* і *перевантаження* (*overloading*) методів.

Механізм *раннього зв’язування* передбачає, що зв'язок між об’єктом класу і поліморфним методом (його адресою) встановлюється під час компіляції програми, тобто, статично, і саме ця адреса використовується в подальшому при

виклику такого методу.

Перевантаження являє собою синтаксичний механізм, який дозволяє визначати декілька різних варіантів методів з одним і тим же ім'ям, але різними прототипами. Компілятор визначає, який саме метод потрібно викликати, за типом фактичних параметрів.

Відповідні методи називають *статичними поліморфними,* оскільки їх поведінка залежить від конкретних типів аргументів.

2) *Динамічний поліморфізм (складний, поліморфізм підтипів)* – передбачає використання єдиного **інтерфейсу**, **що надається базовим класом**, для виклику різних аспектів поліморфного методу у множині класів (типів), об’єднаних в ієрархію «супертип - підтип». Відповідно усі аспекти поліморфного методу повинні мати однакову сигнатуру. Ґрунтується на можливості неявного висхідного перетворення типів при успадкуванні, згідно якого змінній класу-предка (як покажчику або посиланню), можна надавати адресу об’єкта, який є екземпляром довільного класу-нащадка.

Підтримується через механізми *пізнього (динамічного) зв'язування* та *перевизначення (override)* методів.

Перевизначення (*перекриття*) *методу* – це механізм, який дозволяє похідному класу надавати власну реалізацію поліморфного методу, визначеного в базовому класі, за умови, що метод у похідному класі має таку ж сигнатуру. У цьому випадку кажуть, що для роботи з об'єктами похідних класів використовується інтерфейс їх базового класу.

Механізм *пізнього зв’язування* забезпечує зв’язування поліморфного методу із об’єктом під час виконання програми, коли клас об'єкта, для якого викликається метод точно відомий. Реалізується ця можливість за допомогою покажчика або посилання на базовий клас, проініціалізованого адресою (або іменем – для посилання) об’єкта похідного класу.

Механізм пізнього зв’язування застосовується до так званих *віртуальних* функцій. *Віртуальною* називають функцію, яка оголошується в базовому класі з використанням ключового слова *virtual* і перевизначається в одному або

декількох похідних класах. Якщо сигнатури функцій не збігаються, то механізм віртуальності для них не включається.

Особливості реалізації у C++

Для визначення похідного класу у C++ використовується наступна синтаксична конструкція:

class **<ім'я_класу_нащадка>: [<специфікатор_доступу] <ім'я_класу_предка>**

{

<тіло_класу_нащадка>

}

Хоча всі члени класу-предка автоматично стають членами класу-нащадка, спосіб доступу до цих членів із класу-нащадка визначається *видом успадкування*, який визначається специфікатором доступу при оголошенні класу-нащадка (табл. 4).

Таблиця 4. Способи доступу до членів класу-предка

Специфікатор доступу у базовому класі	Тип успадкування		
	public	protected	private
public	public	protected	private
protected	protected	protected	private
private	не доступні	не доступні	не доступні

Якщо специфікатор доступу не вказаний, то за замовчуванням використовується private.

Відповідно розрізняють *відкрите*, *закрите* і *захищене* успадкування.

Реалізація динамічного поліморфізму у C++ базується на використанні механізму *успадкування* і *віртуальних* функцій.

У C++, на відміну від інших мов програмування, зокрема, C#, при перевизначенні віртуальної функції у похідному класі ключове слово *virtual* повторювати не потрібно (хоча це не буде помилкою).

Особливості реалізації успадкування у C#

Синтаксис опису класу-нащадка у C#:

[<список_специфікаторів] class **<ім'я_класу_нащадка> : <ім'я_класу_предка>**

{

<тіло_класу_нащадка>

}

У похідному класі доступ можливий тільки до тих членів успадкованої частини, які описані у класі-предку із специфікаторами доступу public або protected. Доступ до захищених (protected) членів успадкованої частини можливий тільки в межах опису похідного класу. Закриті (private) члени класу-предка у похідному класі є недоступними.

Особливості реалізації успадкування у Python

Синтаксис оголошення похідного класу в Python:

class **<ім'я_класу_нащадка> (<ім'я_класу_предка>) :**

<тіло_класу_нащадка>

В Python всі класи явно чи неявно успадковуються від класу object. Якщо при створенні похідного класу клас-предок не вказується, то неявним чином цей клас буде успадковано від object. В Python допускається тільки відкрите успадкування. Для доступу до членів класу-предка у методах класу-нащадка служить функція super().

Вимоги до програми

1. Реалізація коду відповідно до модульного підходу.
2. Виведення усіх вхідних, проміжних і вихідних даних.
3. У кожному варіанті завдання при описі класів самостійно визначити необхідні поля та методи вводу/виводу. Деякі методи класу-предка можуть бути віртуальними і абстрактними. У програмі-клієнті для збереження сукупності об'єктів використати масив.

Варіанти завдань

1. Створити клас TSystemLinearEquation, який представляє систему лінійних алгебраїчних рівнянь і містить методи для знаходження коренів рівнянь та перевірки того, чи є деякий набір чисел розв'язком системи рівнянь. На основі цього класу створити класи-нащадки, які представляють системи двох та трьох лінійних рівнянь (відповідно з двома та трьома невідомими). Випадковим чином згенерувавши дані для декількох систем двох лінійних рівнянь та декількох систем трьох лінійних рівнянь. Знайти розв'язок даних систем лінійних алгебраїчних рівнянь (обох видів).
2. Створити клас TSeries, який представляє прогресію і містить методи для знаходження n-го члена прогресії і суми перших n членів цієї прогресії. На основі цього класу створити класи-нащадки, які представляють арифметичні та геометричні прогресії. Випадковим чином згенерувати дані для n прогресій (геометрична, арифметична, геометрична, арифметична, і т.д.). Знайти суму перших m членів прогресії, n-ий член якої є найбільшим.
3. Створити клас TLine, що представляє пряму і містить методи для визначення того, чи є інша пряма паралельною / перпендикулярною до неї, та, чи належить вказана точка прямій. На основі цього класу створити класи-нащадки, що представляють пряму на площині і в просторі. Випадковим чином згенерувати дані для створення n прямих у просторі та m прямих на площині. Визначити, чи належить вказана точка хоча б одній прямій на площині, серед тих, які є перпендикулярними до першої (в порядку створення) прямої на площині, та, чи є серед заданих прямих у просторі така, що є перпендикулярною до всіх інших прямих у просторі.
4. Створити клас TTriad, який представляє трійку цілих чисел і містить методи для їх збільшення / зменшення на вказану величину. На основі цього класу створити класи-нащадки TTime ("години:хвилини:секунди") та TDate ("число.місяць.рік"). Випадковим чином згенерувати n дат та m

об’єктів часу. Визначити, які із дат мають значення, що є допустимими, якщо їх трактувати як час. Всі інші дати зменшити на 5 днів, а весь інший час збільшити на 15 хвилин.

5. Створити клас TPair, який представляє пару чисел і містить методи для їх інкрементування / декрементування. Реалізувати класи нащадки TTime (“години:хвилини”) та TMoney (“гривні.копійки”). Згенерувати поступово випадковим чином n пар (час, гроші), де час – тривалість виконання певної роботи, а гроші – вартість однієї хвилини роботи працівника. Обчислити витрати на виконання кожної із робіт.
6. Створити клас TArray, який представляє одновимірний масив і містить методи збільшення / зменшення всіх елементів на вказану величину та знаходження їх середнього арифметичного. Реалізувати класи-нащадки, що представляють одновимірні масиви з елементами цілого та дійсного типів. Випадковим чином створити по m масивів кожного виду. Елементи цілочисельних масивів збільшити на вказану величину, а дійсних масивів зменшити на цю ж величину. Знайти масив, середнє арифметичне елементів якого є найбільшим.
7. Створити клас TVector, який представляє вектор і місить методи для обчислення довжини вектора та скалярного добутку векторів. На основі цього класу створити класи-нащадки, які представляють вектори з просторів R^2 та R^3. За допомогою цих класів обчислити значення виразу

$$S = \langle a, b \rangle + \langle c, d \rangle + |a|,$$

де $a,b \in R^3$, а $c,d \in R^2$.

8. Створити клас TVector, який представляє вектор і містить методи для визначення того, чи є інший вектор паралельним / перпендикулярним до нього та метод знаходження довжини вектора. На основі цього класу створити класи-нащадки, які представляють вектори з просторів R^2 та R^3. Створити 3 двовимірні та 4 тривимірні вектори. Знайти суму довжин векторів, паралельних до першого по порядку двовимірного вектора, та

суму векторів, перпендикулярних до першого по порядку тривимірного вектора.

9. Створити клас TMatrix, який представляє матрицю і містить методи для обчислення детермінанта та суми елементів матриці. На основі цього класу створити класи-нащадки, які представляють квадратні матриці 2-го та 3-го порядку. За допомогою цих класів обчислити вираз

$$S = \left(\sum_{i=1}^{3} \sum_{j=1}^{3} a_{ij} \right) + |A| + |B|,$$

де $A = \|a_{ij}\|_1^3$ – матриця 3-го порядку, а $B = \|b_{ij}\|_1^2$ – матриця 2-го порядку.

10. Створити клас TNumber, який представляє число і містить методи для знаходження суми цифр та першої / останньої цифри числа. На основі цього класу створити класи-нащадки TIntNumber та TRealNumber. Створити m об’єктів цілих чисел та n об’єктів дійсних чисел (дані згенерувати випадковим чином). Знайти суму перших цифр цілих чисел та суму останніх цифр дійсних чисел.
11. Спроектувати клас "Число", який представляє число і містить методи збільшення / зменшення числа у вказану кількість разів та знаходження модуля числа. На основі цього класу створити класи-нащадки "Раціональне число" та "Комплексне число". Створити m об’єктів раціональних чисел та n об’єктів комплексних чисел (дані згенерувати випадковим чином). Раціональні числа зменшити у 2 рази, а комплексні – збільшити у 3 рази. Знайти суму модулів усіх чисел.
12. Спроектувати клас TEquation, який представляє рівняння і містить віртуальні методи для знаходження коренів рівняння та перевірки, чи є деяке значення коренем рівняння. На основі цього класу створити класи-нащадки, які представляють лінійні та квадратні рівняння. Створити n лінійних рівнянь та m квадратних рівнянь, згенерувавши дані для них випадковим чином. Знайти суму коренів для кожного із видів рівнянь (за умови, що вони існують). Перевірити, чи є задане значення коренем вка-

заного рівняння

13. Спроектувати клас “Рухома матеріальна точка”, який представляє точку, що рухається в певному напрямку і містить методи для визначення координати точки в заданий момент часу *t* та обчислення відстані від неї до іншої точки. На основі цього класу створити класи-нащадки “Рухома матеріальна точка *x*”, яка рухається по прямій і її координата визначається як $x=x_0 + a_1 sin(t), y=0$, та “Рухома матеріальна точка (*x*, *y*)”, яка рухається по площині і її координати визначається як $x=x_0 + a_1 sin(t), y=y_0 + a_2 cos(t)$. Створити k об’єктів класу “Рухома матеріальна точка *x*” та n об’єктів класу “Рухома матеріальна точка (*x*, *y*)” (дані згенерувати випадковим чином). Визначити найбільшу відстань між рухомими матеріальним точками у заданий користувачем момент часу *t*.
14. Спроектувати клас “Рухома матеріальна точка”, який представляє точку, що рухається в певному напрямку і містить методи для визначення координати точки в заданий момент часу *t* та обчислення відстані від неї до іншої точки. На основі цього класу створити класи-нащадки “Рухома матеріальна точка (*x*, *y*, *z*)”, яка рухається у просторі і її координата визначається як $x=x_0 + a_1 sin(t), y=y_0 + a_2 cos(t), z=z_0 + a_3 t^2$ та “Рухома матеріальна точка (*x*, *y*)”, яка рухається по площині і її координати визначається як $x=x_0 + a_1 sin(t), y=y_0 + a_2 cos(t), z=0$. Створити q об’єктів класу “Рухома матеріальна точка (*x*, *y*, *z*)” та k об’єктів класу “Рухома матеріальна точка (*x*, *y*)” (дані згенерувати випадковим чином). Визначити найменшу відстань між рухомими матеріальним точками у заданий користувачем момент часу *t*.
15. Спроектувати клас TFunction, який представляє функцію і містить методи збільшення / зменшення всіх коефіцієнтів функції на вказану величину та обчислення значення функції в заданій точці. На основі цього класу створити класи-нащадки “Лінійна функція” (виду a_1x+a_0) та “Квадратична функція” (виду $b_2x^2 + b_1x + b_0$). Створити n лінійних функцій та m квадратичних функцій, згенерувавши дані для них випадковим чином. Кое-

фіцієнти лінійних функцій збільшити на 3, а квадратичних - зменшити на 2. Визначити функцію, яка має найбільше значення у введеній користувачем точці.

16. Створити клас TTime, який представляє час в межах доби і містить методи для його збільшення / зменшення на вказану величину та обчислення часу, що залишився до завершення доби. На основі цього класу створити класи-нащадки TTime12 (“ГГ:ХХ:СС”) та TTime24 (“ГГ:ХХ:СС”). які задають час в 12-годинному та 24-годинному форматі. Створити n об’єктів часу в 12-годинному форматі та m об’єктів часу в 24-годинному форматі. Усі об’єктів часу в 12-годинному форматі зменшити на 5 секунд, а інші - збільшити на 15 хвилин. Визначити найменший час, що залишився до завершення доби.
17. Створити клас TIntNumber, який представляє ціле число у будь-якій системі числення і містить методи для додавання іншого числа, порівняння чисел та переведення числа у вказану систему числення (із заданого переліку). На основі цього класу створити класи-нащадки TIntNumber2 та TIntNumber16, що представляють двійкові та шістнадцяткові числа. Створити m двійкових та n шістнадцяткових чисел. Знайти суму двійкових і суму шістнадцяткових чисел. Перевести отримане значення суми шістнадцяткових чисел у двійковий формат. Визначити, сума яких чисел більша (двійкових чи шістнадцяткових).
18. Створити клас TIntNumber, який представляє ціле число у будь-якій системі числення і містить методи для додавання іншого числа, порівняння чисел та переведення числа у вказану систему числення (із заданого переліку). На основі цього класу створити класи-нащадки TIntNumber2 та TIntNumber8, що представляють двійкові та вісімкові числа. Створити m двійкових та n вісімкових чисел. Знайти суму двійкових і суму вісімкових чисел. Перевести отримане значення суми вісімкових чисел у двійковий формат. Визначити, сума яких чисел менша (двійкових чи вісімкових).

19. Створити клас TMatrix, який представляє матрицю і містить методи збільшення / зменшення всіх її елементів на вказану величину та обчислення їх середнього арифметичного. Реалізувати класи-нащадки, що представляють матриці з елементами цілого та дійсного типів. Випадковим чином створити по m матриць кожного виду. Елементи цілочисельних матриць збільшити на 9, а дійсних матриць зменшити на 5. Знайти матрицю, середнє арифметичне елементів якої є найменшим.
20. Створити клас TIntNumber, який представляє ціле число у будь-якій системі числення і містить методи для інкрементування / декрементування числа та переведення числа у десяткову систему числення. На основі цього класу створити класи-нащадки TIntNumber2 та TIntNumber16, що представляють двійкові та шістнадцяткові числа. Створити m двійкових та n шістнадцяткових чисел. Двійкові числа інкрементувати, шістнадцяткові – декрементувати. Перевести числа у десяткові і знайти найбільше із них.
21. Створити клас TVector, який представляє вектор і містить методи для визначення того, чи є інший вектор паралельним / перпендикулярним до нього та метод знаходження довжини вектора. На основі цього класу створити класи-нащадки, які представляють вектори з просторів R^2 та R^3. Створити 3 двовимірні та 4 тривимірні вектори. Знайти суму довжин векторів, паралельних до першого по порядку двовимірного вектора, та суму векторів, перпендикулярних до першого по порядку тривимірного вектора.
22. Створити клас “Товар”, який містить назву, дату виготовлення, ціну, кількість одиниць, а також методи порівняння дат та обчислення сумарної вартості товару. На його основі створити класи-нащадки “Промисловий товар”, що додатково зберігає умови транспортування, місце знаходження товару (на складі, в торговому залі) та “Харчовий продукт”, який додатково містить термін зберігання дату продукту. Створити n номенклатур промислових товарів та m номенклатур харчових продуктів. Визна-

чити загальну вартість харчових продуктів, термін зберігання яких закінчився, і загальну вартість промислових товарів, які знаходяться на складі.

23. Створити клас TDate, який містить трійку цілих чисел, що представляють число, місяць та рік, і методи для порівняння дат, заданих різними форматами, їх збільшення / зменшення на вказану величину. На основі цього класу створити класи-нащадки TDate1 та TDate2, що представляють дати в форматі “ЧЧ.ММ.РРРР” та “ММ-ЧЧ-РРРР” відповідно. Створити n об’єктів TDate1 та m об’єктів TDate2. Визначити саму пізню дату, а також дати, що належать заданому періоду дат.
24. Створити клас “Банківський рахунок”, який містить назву банку, номер рахунку, статус (діє, закінчився) та методи додавання / зняття коштів. На основі цього класу створити класи-нащадки “Депозитний рахунок”, який додатково містить дату його відкриття, період, ставку, суму коштів, та “Розрахунковий рахунок”, який додатково містить дату останньої операції, залишок коштів на рахунку. Створити n пар депозитних та розрахункових рахунків. В межах вказаного періоду передбачити щомісячне перерахування процентів по депозиту на відповідний розрахунковий рахунок. У випадку закінчення терміну дії депозиту, перерахувати на відповідний розрахунковий рахунок усі кошти (разом з процентами) та закрити відповідний депозитний рахунок.
25. Створити клас “Подія”, який містить дату і час певної події, а також методи обчислення часу, що залишився до початку події. На основі цього класу створити класи-нащадки “День народження”, який містить ПІБ іменинника, його вік, місце проведення свята, та “Зустріч”, який містить ПІБ людини, з якою призначена зустріч, і місце зустрічі. Створити розклад активностей особи на конкретну дату, який включає n зустрічей і одне святкування дня народження. Визначити останню заплановану зустріч в цей день і інтервал часу від її закінчення до початку святкування дня народження.

26. Визначити клас TPrism, який представляє правильну призму і містить методи для знаходження площі її поверхні та об’єму. На основі цього класу створити класи-нащадки TPrism3 та TPrism4, які представляють правильну трикутну та чотирикутну призми. Дані для створення правильної трикутної та чотирикутної призми вводяться з клавіатури. Створити послідовно m правильних призм (трикутних та чотирикутних), об’єм кожної з яких на 5 більше попередньої. Для трикутних призм знайти сумарний об’єм, а для чотирикутних – суму площ поверхні.
27. Створити клас TTriangle, який містить координати вершин і методи обчислення його площі та периметру. На основі цього класу створити класи-нащадки, які представляють рівносторонні, прямокутні та рівнобедрені трикутники. Створити певну кількість трикутників кожного виду, щоб їх сумарна кількість дорівнювала n. Для рівносторонніх та прямокутних трикутників обчислити суму їх площ, а для рівнобедрених – суму всіх периметрів.
28. Створити клас TQuadrangle, який містить координати вершин і методи обчислення площі та периметру. На основі цього класу створити класи-нащадки, які представляють прямокутник, квадрат, паралелограм (квадрат створити на основі прямокутника). Створити певну кількість чотирикутників кожного виду, щоб їх сумарна кількість дорівнювала n. Обчислити суму площ прямокутників та квадратів і суму периметрів паралелограмів.
29. Спроектувати клас “Особа”, який містить ПІБ, дата народження і методи визначення її віку та обчислення місячного заробітку. На основі цього класу створити класи-нащадки “Студент”, який містить додатково номер академічної групи студента, середній рейтинговий бал за результатами останньої сесії, статус отримання стипендії у новому семестрі (підвищена, звичайна, немає), та “Викладач”, який містить додатково назви дисциплін, які викладає даний викладач, планову кількість годин, які він повинен провести по кожній дисципліні, за місяць. Створити n студентів і

m викладачів. Для студентів розрахувати розмір місячної стипендії (підвищена, якщо середній рейтинговий бал більше 95, і звичайна – якщо середній рейтинговий бал більше 85, але менше 95), для викладачів – місячну заробітну плату. Визначити вік викладача, що має найбільшу місячну заробітну плату.

30. Спроектувати клас “Особа”, який містить ПІБ, дата народження і методи визначення її віку та обчислення рейтингу. На основі цього класу створити класи-нащадки “Студент”, який містить додатково номер академічної групи студента, дисципліни, що вивчалися ним у останньому семестрі, рейтинговий бал, отриманий за результатами здачі цих дисциплін, та “Викладач”, який містить додатково назви дисциплін, які викладає даний викладач, рейтинговий бал популярності по кожній дисципліні (від 1 до 10). Створити n студентів і m викладачів. Для студентів розрахувати їх середній рейтинговий бал, для викладачів – середній рейтинговий бал популярності. Визначити кількість неповнолітніх студентів, що мають академічні заборгованості (менше 60 балів хоча б по одній дисципліні).
31. Створити клас TBody, який представляє просторову геометричну фігуру з методами обчислення площі її поверхні та об’єму. На основі цього класу створити класи-нащадки TParallelepiped та TBall. Створити певну кількість паралелепіпедів та куль, щоб їх сумарна кількість дорівнювала n. Для куль знайти сумарний об’єм, а для паралелепіпедів – суму площ поверхні.
32. Спроектувати клас TFigure, який представляє просторову геометричну фігуру з методами обчислення площі її поверхні та об’єму. На основі цього класу створити класи-нащадки TPyramid та TCylinder. Створити n пірамід і m циліндрів. Знайти циліндр з найбільшим об’ємом і піраміду – з найменшою площею поверхні.
33. Створити клас TFigure, який представляє геометричну фігуру на площині і методи обчислення її площі та периметру. На основі цього класу ство-

рити класи-нащадки THexagon та TPentagon, які задаються своїми координатами. Створити n п’ятикутників і m шестикутників. Знайти п’ятикутник з найбільшим периметром і шестикутник – з найменшою площею.

34. Створити клас TIntNumber, який представляє ціле число у будь-якій системі числення і містить методи для інкрементування / декрементування числа та переведення числа у десяткову систему числення. На основі цього класу створити класи-нащадки TIntNumber2 та TIntNumber8, що представляють двійкові та вісімкові числа. Створити m двійкових та n вісімкових чисел. Двійкові числа інкрементувати, вісімкові – декрементувати. Перевести числа у десяткові і знайти найменше із них.
35. Створити клас TLine, що представляє пряму і містить методи для визначення того, чи є інша пряма паралельною / перпендикулярною до неї, та, чи належить вказана точка прямій. На основі цього класу створити класи-нащадки, що представляють пряму на площині і в просторі. Випадковим чином згенерувати дані для створення n прямих у просторі та m прямих на площині. Визначити, чи належить вказана точка хоча б одній прямій на площині, серед тих, які є перпендикулярними до першої (в порядку створення) прямої на площині, та, чи є серед заданих прямих у просторі така, що є перпендикулярною до всіх інших прямих у просторі.

Контрольні питання

1. У чому полягає сутність механізму успадкування?
2. Поясніть роль специфікатора доступу в успадкуванні.
3. Поясніть, чому в C# немає множинного спадкування.
4. Поясніть різницю між прямим базовим класом та непрямим.
5. Поясніть, як керувати викликом конструкторів базового класу у конструкторі похідного класу, наведіть приклад.

6. У чому полягає сутність поліморфізму?
7. Яка функція (метод) називається віртуальною?
8. Яка функція називається чистою віртуальною? Поясніть її призначення
9. Який метод називається абстрактним? Поясніть його призначення
10. Який клас називається абстрактним? Навіщо створювати абстрактні класи?
11. Наведіть приклади та поясніть відмінності раннього та пізнього зв’язування.

Лабораторна робота 6
СТРУКТУРИ ДАНИХ

Мета роботи - вивчити особливості організації і обробки дерев.

Теоретичні відомості

Деревом називають структуру, кожен елемент (вершина, вузол) якої пов'язаний з будь-якою кількістю інших її елементів. Дерево характеризується наступними властивостями:

– має єдину вершину, на яку не посилається ніяка інша вершина, і яка називається *коренем дерева*;

– починаючи з кореня і слідуючи по певному ланцюжку посилань (покажчиків), що містяться у вершинах, можна здійснити доступ до будь-якого елемента дерева;

– на кожну вершину, окрім кореня, є єдине посилання.

Вершини дерева (за виключенням кореня) розподілені серед m непересічних множин, кожна із яких в свою чергу є деревом. Такі дерева називають *піддеревами* даного кореня. Для кожної вершини k існує послідовність вершин k_0, k_1, ..., k_n що утворює гілку довжини n.

Серед будь-якої пари безпосередньо зв'язаних вершин дерева можна виділити *предка та нащадка*. *Нащадок* є коренем певного піддерева предка. Якщо вершина не має нащадків, то її називають *листом* дерева.

Степінь вершини k - це кількість її нащадків (піддерев). Максимальну степінь вершин у певному дереві називають *степенем дерева*.

Якщо в дереві порядок слідування вершин є фіксованим, то такі дерева називають *впорядкованими*.

За кількістю можливих нащадків у вершин розрізняють *бінарні* (*двійкові*) та *недвійкові* дерева. *Бінарне дерево* - це впорядковане дерево, кожна вершина якого має не більше двох нащадків.

Вершини дерева у мовах C/C++ оголошують наступним чином:

```
struct TNode
{   char  inf;            // інформаційне поле вершини
TNode *left,   // покажчик на ліве піддерево
*right;           // покажчик на праве піддерево
};
```

Для листків покажчики left та right мають значення NULL. Адреса кореневої вершини зберігається у спеціальному покажчику, наприклад, TNode *root.

Формується дерево шляхом включення в нього нових вершин. Для цього треба мати адресу батьківської вершини (parent), а також тип піддерева, що створюється (ліве або праве). Наприклад, якщо

```
enum tag_type {RIGHT, LEFT} type;
```

то функція створення бінарного дерева може мати вид:

```
TNode *curr = new TNode;        // створення нової вершини дерева
cin>> curr -> inf;              // заповнення її інформаційного поля
curr ->left = NULL;             // обнулення її покажчика на ліве піддерево
curr -> right = NULL;           // обнулення її покажчика на праве піддерево
if (parent)                     // parent – покажчик на предка
{   if (type == LEFT)
   parent ->left = curr;       // створена вершина стає лівим нащадком предка
else parent->right = curr;     // створена вершина стає правим нащадком предка
else  root = curr;              // створена вершина стає коренем дерева
}
```

Над деревами виконують такі дії:

- обхід дерева (всіх його вершин),
- пошук у дереві (заданої вершини),
- включення вершини в дерево,
- видалення вершини з дерева тощо.

Існує декілька схем обходу дерева.

Обхід, при якому спочатку відвідують корінь, а потім всі його піддерева у

відповідності з їхнім упорядкуванням називають обходом дерева в *прямому порядку* (*зверху вниз*). Алгоритм обходу дерева в прямому порядку (рис. 1):

1) обробити кореневу вершину поточного піддерева;
2) перейти до обробки лівого піддерева (в прямому порядку);
3) обробити праве піддерево (в прямому порядку).

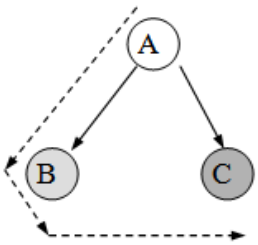


Рис. 1 Обхід дерева в прямому порядку

Обхід, при якому спочатку проглядаються піддерева у відповідності з їхнім упорядкуванням, а потім корінь піддерева, називають обходом дерева у *зворотному порядку* (*знизу вверх*). Алгоритм обходу дерева у зворотному порядку (рис. 2):

1) рекурсивно обробити ліве піддерево поточного піддерева;
2) рекурсивно обробити праве піддерево;
3) потім - вершину поточного піддерева.

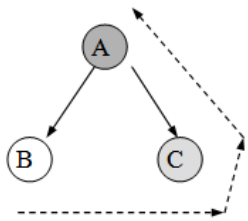


Рис. 2 Обхід дерева у зворотному порядку

Обхід, при якому відвідується спочатку ліве піддерево, потім вузол, потім - праве піддерево називають *симетричним обходом* дерева (*обходом зліва направо*). Алгоритм симетричного обходу дерева (рис. 3):

1) рекурсивно обробити ліве піддерево поточного піддерева;
2) обробити вершину поточного піддерева;
3) рекурсивно обробити праве піддерево.

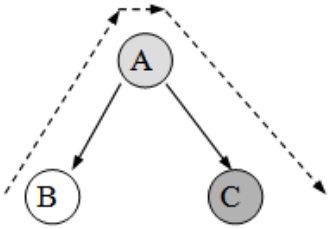


Рис. 3 Симетричний обхід дерева

Обхід дерева слід проводити за рахунок послідовного виділення в дереві подібних найпростіших піддерев і застосуванням до кожного з них відповідного правила обходу. Виділення починається з кореневої вершини дерева. Як приклад, розглянемо обхід різними способами наступного дерева з числовими компонентами (рис. 4):

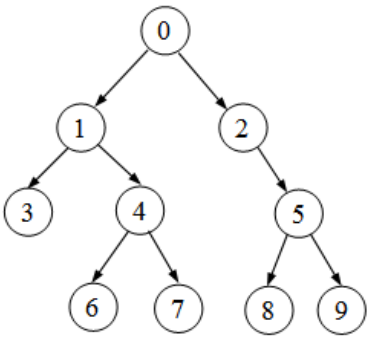


Рис. 4 Обхід різними способами

При прямому обході отримуємо наступний порядок обходу вершин: 0-1-3-4-6-7-2-5-8-9, при зворотному - 3-6-7-4-1-8-9-5-2-0, при симетричному - 3-1-6-4-7-0-2-8-5-9.

У результаті обходу дерева зверху вниз утворюється *префіксна* форма виразу, при обході знизу вверх - *постфіксна* форма, а при обході зліва направо - *інфіксна* форма. Будь-який спосіб обходу дерева можна реалізувати рекурсивною процедурою.

Бінарні дерева часто використовуються для зображення множин, елементи яких потрібно знаходити за заданим значенням ключа. Такі дерева називаються *деревами бінарного пошуку.* Особливість подібного дерева полягає в тому, що для будь-якої її вершини всі ключі в її лівому піддереві менші ключа цієї вершини, а всі ключі в її правому піддереві більші ключа вершини. Приклад дерева пошуку з цілими ключами (рис. 5):

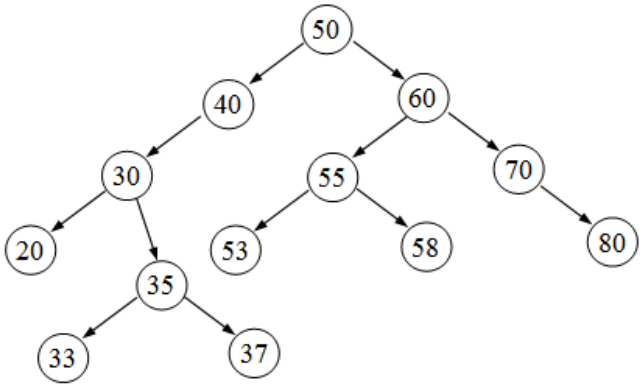


Рис. 5 Дерево пошуку з цілими ключами

Вершина із заданим значенням ключа буде знайдена доволі швидко, якщо спускатися від кореня дерева бінарного пошуку за таким правилом: ліве піддерево обирається тоді, коли значення вершини, що розглядається, більше за шукане, а праве піддерево — тоді, коли вказане значення менше за шукане. Якщо значення вершини дорівнює шуканому, пошук слід завершити. Якщо пошук привів до порожньої гілки дерева, то ключового значення в дереві немає.

Дерева змінної структури - дерева, кількість елементів яких під час роботи програми може збільшуватися або зменшуватися.

Включення вершини в дерево має здійснюватися так, щоб не порушувалася властивість упорядкованості ключів. Це правило буде дотримане, якщо застосувати алгоритм знаходження ключового значення у бінарному дереві пошуку, а включення нового елемента здійснювати тоді, коли пошук завершився безуспішно.

Видалення вершини бінарного дерева пошуку можливе трьома способами:

- вершина, що видаляється, не має нащадків (є листком дерева);
- вершина, що видаляється, має одного нащадка;
- вершина, що видаляється, має двох нащадків.

Розглянемо фрагмент дерева пошуку з цілими ключами (рис. 6).

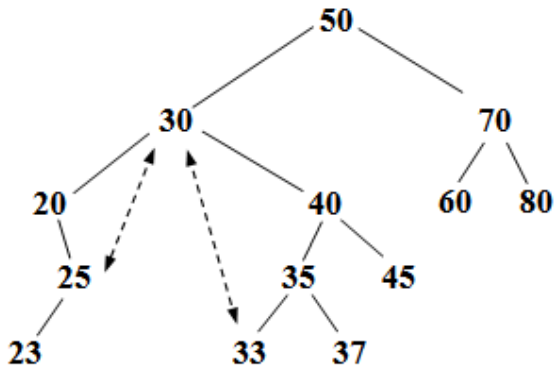


Рис. 6 Фрагмент дерева пошуку з цілими ключами

Задача розв'язується найпростішим способом тоді, коли вершина, що видаляється, є листком дерева. У такому випадку слід звільнити область динамічної пам'яті, яку ця вершина займала, та обнулити покажчик на дану вершину у її предка. Наприклад, для видалення вершини з ключем 23 (рис. 6) досить встановити 25 -> left = NULL.

Якщо видаляється вершина**,** яка має одного нащадка, то покажчику на цю вершину слід присвоїти адресу її нащадка і звільнити пам'ять, яку займала дана вершина. Наприклад, для видалення вершини з ключем 20 (рис. 6) досить встановити 30 ->left = 20 ->right = 25.

Найскладнішим є випадок, коли вершина, що видаляється, має двох нащадків. У цьому випадку на її місце слід переставити іншу вершину дерева так, щоб не порушувалася властивість впорядкованості ключів.

Правило визначення вершини, яка повинна замінити ту, що видаляється, складається з двох альтернативних дій:

1) або увійти в ліве піддерево вершини, що видаляється, і в цьому піддереві спуститися якомога глибше, дотримуючись тільки правих нащадків; це дозволяє знайти в дереві найближчу меншу вершину (наприклад, для вершини 30 це буде вершина 25);
2) або увійти в праве піддерево вершини, що видаляється, і спуститися в ньому якомога глибше дотримуючись тільки лівих нащадків; це дозволяє знайти найближчу більшу вершину (наприклад, для тієї ж вершини 30 це буде вершина 33).

Тобто, ключ вершини, що видаляється, замінюється на значення мінімального (максимального) ключа серед вершин правого (лівого) піддерева даної вершини. Таке видалення називають правим (лівим).

Варіанти завдань

1. Текстовий файл містить програму мовою C/C++. Надрукувати в алфавітному порядку всі ідентифікатори цієї програми, вказавши для кожного з них число входжень у текст програми. Для збереження ідентифікаторів використати структуру типу дерева, елементами якого є ідентифікатор і число його входжень у текст.
2. Побудувати дерево, елементами якого є дійсні числа. Обчислити середнє арифметичне усіх його елементів.
3. Побудувати дерево, що відображає формулу (a*(b+c))/d, де коренем дерева та його підкоренями є операції "*, +, -, /", а листками - змінні a, b, c, d. Надрукувати дерево переліком своїх вершин на рівнях, які містять відповідні вершини.
4. Побудувати дерево, елементами якого є числа. Надрукувати дерево. Визначити кількість від'ємних та додатних елементів дерева.
5. Побудувати двійкове дерево, елементами якого є символи. Визначити, чи знаходиться у цьому дереві елемент, значення якого вводиться з клавіатури. Якщо елемент знайдений, то підрахувати число його входжень.
6. Побудувати двійкове дерево пошуку з літер заданого рядка. Видалити з дерева літери, що зустрічаються більше одного разу. Вивести елементи дерева, що залишилися, при його постфіксному обході.
7. Заданий текст. Підрахувати кількість повторень кожного слова. Побудувати дерево із слів тексту. Слова, що зустрічаються найчастіше розмістити на верхньому рівні, на інших рівнях дерева розмістити слова з меншою кількістю повторень.
8. Побудувати бінарне дерево, елементами якого є дійсні числа. Знайти зна-

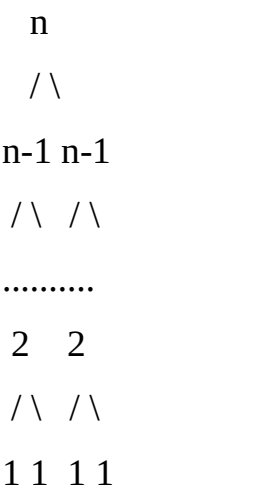
чення найбільшого елемента цього дерева та надрукувати його.

9. Заданий рядок символів латинського алфавіту. Побудувати дерево, в якому значеннями вершин є символи, що розміщуються на рівнях відповідно до кількості їх повторень у рядку.

10. Побудувати дерево, елементами якого є цілі числа. Визначити кількість вершин на n-му його рівні та кількість рівнів.

11. Побудувати дерево, що відображає формулу ((a+b)*c-d), де коренем дерева та його підкоренями є операції " +, -, *, /", а листками є змінні a, b, c, d . Вивести значення дерева-формули. Надрукувати відповідні піддерева y1=a+b, y2=y1*c, y3=y2-d.

12. Побудувати дерево наступного виду:



, де n -додатне ціле число

13. Побудувати бінарне дерево для зберігання даних виду: деталь, її кількість, постачальник. Забезпечити виконання операцій додавання нового елемента у дерево в діалоговому режимі та визначення постачальника найбільшої кількості деталей.

14. Побудувати дерево, елементами якого є символи. Визначити максимальну глибину дерева (число гілок на найбільшому з маршрутів від кореня дерева до листків).

15. Текстовий файл містить програму мовою C/C++. Для збереження ідентифікаторів програми використати структуру типу дерева, елементами якого є ідентифікатори. Номер рядка, в якому оголошений ідентифікатор,

визначає рівень дерева. Ліва гілка дерева визначає змінні, права гілка - константи.

16. Побудувати бінарне дерево, елементами якого є цілі числа. Підрахувати кількість вершин на n-му рівні цього дерева (нульовий рівень - корінь цього дерева) та надрукувати ці елементи.

17. Побудувати дерево, що відображає формулу ((a+b)/c)*d, де коренем дерева та його підкоренями є операції, а листками - змінні. Ввести значення змінних та визначити значення дерева-формули. Надрукувати відповідні піддерева, наприклад: y1=a+b, y2:= y1/c, y3=y2*d.

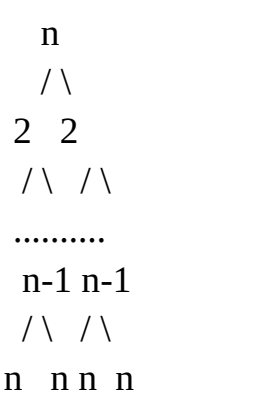
18. Відповідно до виразу, що зчитується з текстового файлу, побудувати дерево-формулу та обчислити значення цієї формули.

19. Побудувати бінарне дерево, елементами якого є слова. Знайти у ньому значення слова, введеного з клавіатури, визначивши номер відповідного рівня.

20. Текстовий файл містить програму мовою C/C++. Надрукувати в порядку зростання номери всіх рядків, що містять кожний ідентифікатор програми. Для збереження ідентифікаторів використати структуру типу дерева, елементами якого є ідентифікатор і номер рядка, де вони зустрічаються.

21. Побудувати дерево, елементами якого є символи. Знайти довжину шляху (число гілок) від кореня до значення символу, введеного з клавіатури. Різновид дерева вибрати самостійно.

22. Побудувати дерево наступного типу:



, де n -додатне ціле число

23. Побудувати два бінарних дерева, елементами якого є цілі числа. Об'єднати їх, уникаючи дублювання елементів в сумарному дереві.
24. Побудувати дерево, елементами якого є дійсні числа. Поміняти місцями найбільше та найменше значення дерева.
25. Побудувати бінарне дерево для зберігання даних виду: найменування товару, його кількість, вартість одиниці. Забезпечити виконання операцій додавання нового елемента у дерево в діалоговому режимі та підрахунку загальної вартості вказаного товару.
26. Побудувати двійкове дерево пошуку, в вершинах якого знаходяться слова з текстового файлу. Визначити кількість вершин дерева, що містять слова, які починаються на зазначену букву.
27. Побудувати дерево, елементами якого є цілі числа. Визначити кількість вузлових вершин даного дерева та надрукувати їх координати (номер рівня та номер гілки).
28. Написати програму, що будує дерево-формулу та перетворює в ньому всі піддерева, що відповідають формулам $((f_1 \pm f_2)*f_3)$, на піддерева виду $((f_1*f_3)\pm(f_2*f_3))$.
29. Побудувати дерево, елементами якого є символи. Визначити і вивести на друк усі термінальні вершини (листя) цього дерева.
30. Побудувати і вивести на екран бінарне дерево наступного виразу: 9 + 8 * (7 + (6 * (5 + 4) – (3 – 2)) +1)). Реалізувати постфіксний, інфіксний та префіксний обходи дерева і вивести відповідні вирази на екран.
31. Побудувати дерево, елементами якого є символи. Знайти у ньому символ, введений з клавіатури, визначивши номер відповідного рівня.
32. Побудувати дерево, що відображає формулу (a*(b+c))/d, де коренем дерева та його підкоренями є операції "*, +, -, /", а листками - змінні a, b, c, d. Надрукувати дерево переліком своїх вершин на рівнях, які містять відповідні вершини.
33. Текстовий файл містить програму мовою C/C++. Для збереження ідентифікаторів програми використати структуру типу де¬рева, елементами

якого є ідентифікатори. Номер рядка, в якому оголошений ідентифікатор, визначає рівень дерева. Ліва гілка дерева визначає змінні, права гілка - константи.

34. Побудувати двійкове дерево пошуку з літер заданого рядка. Видалити з дерева літери, що зустрічаються більше одного разу. Вивести елементи дерева, що залишилися, при його постфіксному обході.

35. Відповідно до виразу, що зчитується з текстового файлу, побудувати дерево-формулу та обчислити значення цієї формули.

Контрольні питання

1) Чим дерево відрізняється від лінійних списків?
2) Дати означення листка та кореня дерева.
3) У чому полягає особливість бінарних дерев?
4) Як включити вершини у дерево?
5) Як видалити вершини із дерева?

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.*Стивен Прата*. Язык программирования C++. Лекции и упражнения — М.: Издательский дом «Вильямс», 2007. — 1184 с.

2. Курс лекцій з дисципліни «Основи програмування -2. Методології програмування».І.П.Муха. [Електронний ресурс]. – Режим доступу: https://ipo.kpi.ua/study/i-semestr/

3. *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на C++. Пер. с англ. – М.: СПб.: “Издательство БИНОМ”-“Невский Диалект”, 2001.-560 с.

4.*Шилдт Г.* Теория и практика C++: пер. с англ.-СПб.: BHV-Санкт-Петербург,1999.-416 с.