МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІГВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ
КАФЕДРА ІНФОРМАЦІЙНИХ ТА КОМП'ЮТЕРНИХ СИСТЕМ

ОСНОВИ ПРОГРАМУВАННЯ НА C/C++

Методичні вказівки
до виконання самостійної роботи
з дисципліни «Програмування»
для студентів спеціальності
123 "Комп'ютерна інженерія"

Затверджено
на засіданні кафедри
інформаційних і комп'ютерних систем

Протокол № 1 від «27серпня 2018 р.

Чернігів ЧНТУ 2018

Основи програмуванняна С/С++. Методичні вказівки до самостійної роботи з дисципліни «Програмування» для студентів напрямів підготовки 123 – “Комп’ютерна інженерія”. /Укл.: Риндич Є.В., Солдатов А.Ю. – Чернігів: ЧНТУ, 2018. – 53 с.

Укладачі: Риндич Євген Володимирович, кандидат технічних наук, доцент, доцент кафедри інформаційних та комп’ютерних систем;

Солдатов Артем Юрійович, асистенткафедри інформаційних та комп’ютерних систем.

Відповідальний за випуск: С.В. Зайцев, зав. кафедрою інформаційних та комп’ютерних систем, д-р. техн. наук, доцент.

Рецензент: Є. В. Нікітенко, кандидат фізико-математичних наук, доцент, доцент кафедри інформаційних та комп’ютерних систем

ЗМІСТ

ВСТУП

Мова С++ надає ефективні та гнучкі засоби визначення нових типів. Використовуючи визначення нових типів, що відповідають поняттям проблемної області (з додатками), програміст може розбивати розроблювану програму на частини, якими легко контролювати, (принцип «розділяй і володарюй»).

Новий тип створюється користувачем для визначення поняття, якому серед вбудованих типів немає відповідності. Набагато легше зрозуміти програму, в якій створюються типи, що відповідають поняттям проблемної області. Добре визначені типи роблять програму ясною і короткої. Крім того, компілятор може виявити неприпустиме використання даних.

Такий метод побудови програм називають абстракцією даних. Інформація про типи міститься в об'єктах, тип яких визначається користувачем. Програмування з застосуванням об'єктів називають об'єктно-орієнтованим ([1,2]).

Мова С ++ є мовою об'єктно-орієнтованого програмування (ООП). Автор мови створював його з метою підтримки абстракції даних і об'єктно-орієнтованого програмування.

Об'єктно-орієнтована мова - це мова програмування, в якій програма задається описом поведінки сукупності взаємопов'язаних об'єктів. Об'єктно-орієнтоване програмування має справу з об'єктами. Об'єктно-орієнтовані мови включають такі основні риси: інкапсуляція даних, поліморфізм, спадкування.

Об'єкти можуть включати закриті (private) дані і правила їх обробки, доступні тільки об'єкту, захищені (protected), доступні об'єкту і його спадкоємцям, а також загальні (public) дані і правила, які доступні об'єктам і модулям в інших частинах програми.

Методичні вказівки містять опис виконання п’яти лабораторних робіт. Студент виконує лабораторну роботу в QtCreator або MicrosoftVisualStudio в залежності від варінта, який визначається останньою цифрою залікової книжки.

По кожній роботі студент повинен оформити звіт. Звіти оформляються за допомогою текстового редактора Word на папері формату А4, у відповідності з вимогами стандартів на оформлення технічної документації. Звіт по роботі є розділом підсумкового документа. В кінці семестру звіти зшиваються в єдиний підсумковий документ з титульним листом, підписуються у керівника, після чого студент отримує допуск до іспиту.

ЛАБОРАТОРНА РОБОТА №1

НАЙПРОСТІШІ КЛАСИ

Мета роботи:

– Познайомитися з найпростішими класами на мові с++.
– Познайомитися з об’єктно-орієнтованим програмуванням.

1.1 Теоретичні відомості

Класи представляють абстрактні типи даних з відкритим інтерфейсом і прихованої внутрішньої реалізацією. У класах реалізовані базові принципи об'єктно орієнтованого програмування (ООП):

1) абстракція даних;

2) інкапсуляція - в класах об'єднуються дані і методи (функції) для роботи з ними, так як лише через методи можливий доступ до прихованим даними класу;

3) спадкування - в похідних класах успадковуються члени базового класу;

4) поліморфізм - можливість використання одних і тих же методів для роботи з різними об'єктами базового і порожден- них їм класів.У мові С ++ об'єкти описують за допомогою класів.

Клас являє собою сукупність даних, що описують об'єкти, та функції, які керують цими об'єктами.

Загальна схема опису класу:

```
class ім'я_класу
{
тіло класу (опис даних і функцій)
};
```

Тіло класу зазвичай складається з розділів, які мають різні рівні доступу до своїх елементів. Кожен розділ має свою мітку:

- private - закриті (приватні) елементи;
- protected - захищені елементи;
- public - відкриті (загальні) елементи.

Якщо для першого розділу рівень доступу не вказано, то за замовчуванням цей розділ має рівень доступу private.

Елементи даних класу оголошуються точно так само як звичайні змінні, за винятком тільки того, що вони не можуть бути явно ініціалізовані. Як правило, елементи даних закривають від прямого доступу.

Короткі функції, що не містять циклів, можна визначити усередині опису класу, компілятор, по можливості, зробить їх вбудованими (inline). Визначення більш складних функцій виносять з тіла класу, тоді визначення функції виглядає так:

```
Тип ім'я_класу :: ім’я_функції (параметри)
{
```

тіло функції
}

Після того як клас визначений, можна оголосити об'єкти, використовуючи ім'я класу як ім'я типу.

Опис об'єкта:

Ім’я_класа ім’я_об'єкта;

Доступ до елементів класу здійснюється за допомогою операції «точка». Якщо оголошують покажчик на об'єкт, то доступ до елементів здійснюється за допомогою операції "->":

Ім’я_об'єкта -> ім’я_елемента

При розробці програм, що використовують класи, рекомендується для класу створювати окремий заголовок, який потім підключають в текст головної програми.

1.2 Клас, що описує коло

```
#include <iostream>
#include <conio.h>
#include <windows.h>
using namespace std;
class Circle
{
// Закриті елементи даних:
float x, y, r;
// Відкриті функції (методи):
public:
// фукції, що забезпечують доступ до елементів даних:
void Set_x (float xx) {x = xx;}
void Set_y (float yy) {y = yy;}
void Set_r (float rr) {r = rr;}
float Get_x () {return x;}
float Get_y () {return y;}
float Get_r () {return r;}
// Функції вводу-виводу:
void input ();
void output ();
};
// Визначення функцій:
void Circle :: input ()
{
cout << "Задайте x:" << endl;
cin >> x;
cout << "Задайте y:" << endl;
cin >> y;
do
{
cout << "Задайте r> 0:" << endl;
cin >> r;
} While (r <0);
```

```
}
void Circle :: output ()
{
cout << "Значення x:" << x << endl;
cout << "Значення y:" << y << endl;
cout << "Значення r:" << r << endl;
}
// Приклад об'єктно-орієнтованого програмування:
int main ()
{
// Налаштування шрифтів і регіональних стандартів:
if (SetConsoleCP (1251) == 0)
// перевірка правильності установки кодування символів для введення
{
cerr << "Fialed to set codepage!" << endl;
/ * Якщо не вдалося встановити кодову сторінку, поява повідомлення про помилку * /
}
Circle a;
a.input ();
a.output ();
float xx, yy, rr;
cout << "Задайте нове значення x:" << endl;
cin >> xx;
cout << "Задайте нове значення y:" << endl;
cin >> yy;
cout << "Задайте нове значення r:" << endl;
cin >> rr;
a.Set_x (xx);
a.Set_y (yy);
a.Set_r (rr);
cout << "Значення x =" << a.Get_x () << endl;
cout << "Значення y =" << a.Get_y () << endl;
cout << "Значення r =" << a.Get_r () << endl;
a.output ();
_getch ();
return 0;
}
```

1.3 Клас, що описує двовимірний масив

```
#include <iostream>
#include <iostream>
#include <conio.h>
#include <windows.h>
#include <iomanip>
using namespace std;
class DArray
{
int ** a; // Покажчик для створення двовимірного масиву
int x, y; // розмірмасиву
```

```
public:
// Функціїдоступу:
void Set_array (int **, int, int);
int ** Get_array ();
// Функціївводу-виводу:
void input ();
void output ();
};
// Визначенняфункцій:
/ * Функція, якарозміщуєвпам'ятімасив a ізаповнюєйогоданимизмасиву aa, переданогофункціїякпараметр * /
void DArray :: Set_array (int ** aa, int xx, int yy)
{
x = xx; y = yy; // Копіюєморозміримасиву aa
// Створюємо в пам'яті структуру, яка працює як таблиця:
a = new int * [x]; // Створюємо масив покажчиків на рядки
for (int i = 0; i <x; i ++)
a [i] = new int [y]; // виділяємомісцепідрядкитаблиці
// Заповнюємо масив даними, копіюючи елементи з масиву aa:
for (int i = 0; i <x; i ++)
for (int j = 0; j <y; j ++)
a [i] [j] = aa [i] [j];
}
// Функція, яка повертає покажчик на масив:
int ** DArray :: Get_array ()
{
return a;
}
// Функція для введення масиву з клавіатури:
void DArray :: input ()
{
cout << "Задайте кількість рядків матриці";
cin >> x;
cout << "Задайте кількість стовпців матриці";
cin >> y;
// Розміщення масиву в пам'яті:
a = new int * [x];
for (int i = 0; i <x; i ++)
a [i] = new int [y];
// Введенняелементівмасиву:
for (int i = 0; i <x; i ++)
{
for (int j = 0; j <y; j ++)
{
cout << "Задайте елемент"<< i << "-й рядка i" << j << "- го стовпця:";
cin >> a [i] [j];
}
cout << endl;
}
}
// Функція для виведення масиву на екран:
void DArray :: output ()
```

```
{
for (int i = 0; i <x; i ++)
{
for (int j = 0; j <y; j ++)
{
cout << setw (4) << a [i] [j];
}
cout << endl;
}
}
// Прикладпрограми, щовикористовуєклас DArray:
int main ()
{
// Налаштуванняшрифтівірегіональнихстандартів:
if (SetConsoleCP (1251) == 0)
cerr << "Fialed to set codepage!" << endl;
DArray A, B; // Оголошенняоб'єктів
int ** c; // Покажчик на масив-джерело
/ * Створення масиву з [2] [2], елементи якого дорівнюють сумі номера рядка і номера стовпчика * /
c = new int * [2];
for (int i = 0; i <2; i ++)
c [i] = new int [2];
for (int i = 0; i <2; i ++)
for (int j = 0; j <2; j ++)
c [i] [j] = i + j;
B.Set_array (c, 2,2); / * Визначаємооб'єкт B, копіюючиелементизмасиву-джерела * /
B.output (); // Виведення масиву B на екран
A.input (); // Введення масиву A с клавіатури
A.output (); // Виведення масиву A на екран
_getch ();
return 0;
}
```

1.4 Завдання на лабораторну роботу

Для захисту лабораторної роботи необхідно виконати 2 завдання із розділів 1.4.1 та 1.4.2. Завдання виконуються в залежності від варіанта. Варіант визначається за останньою цифрою залікової книжки.

1.4.1 Завдання 1

Таблиця 1.1 – Завдання 1 для самостійної роботи

Варіант	Завдання
0	Розробити клас Integer для роботи з цілими числами. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену класу. Крім цього визначити функцію-член, яка визначає суму цифр об'єкта класу Integer.

1	Розробити клас Real для роботи з речовими числами. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену класу. Крім цього визначити функцію-член, меняющуюместамі цілу і дробову частину об'єкта класу Real.
2	Розробитиклас Byte для роботи з беззнаковими цілими. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену класу. Крім цього визначити функцію-член, що виводить бінарний еквівалент об'єкта класу Byte.
3	Розробитиклас Word для роботи з беззнаковими цілими. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену класу. Крім цього визначити функцію-член, що виводить шістнадцятковий еквівалент об'єкта класу Word.
4	Розробитиклас String для роботи з рядками. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член, яка підраховує кількість символів об'єкта класу String.
5	Розробитиклас Date для роботи з датами. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член TodayDate (), яка повертає системну дату об'єкту класу Date.
6	Розробитиклас Time для роботи з тимчасовими параметрами. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член TodayTime (), яка повертає системний час об'єкту класу Time.
7	Розробитиклас Complex для роботи з комплексними числами. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член, меняющуюместамі речову і уявну частину об'єкта класу Complex.
8	Розробитиклас Point для роботи з об'єктами типу точка на площині. Визначити в цьому класі функції-члени, які забезпечують введення / виводелементов класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член, що визначає відстань між двома точками.
9	Розробитиклас String для роботи з рядками. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член, яка підраховує суму чисел, що

	зустрічаються в об'єкті класу String.

1.4.2Завдання 2

Таблиця 1.2 – Завдання 2 для самостійної роботи

Варіант	Завдання
0	Розробити клас Rectangle для роботи з плоскими прямокутниками, сторони якого паралельні осям координат. В якості членів-даних задати координати двох точок прямокутника (ліву верхню і нижню праву). Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу, що задають координати двох вершин прямокутника. Додатково визначити функцію-член цього класу, яка визначає приналежність точки з заданими координатами прямокутника.
1	Розробити клас ThreeAngle для роботи з плоскими трикутниками. В якості членів-даних задаються довжини трьох сторін трикутника. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, доступ до закритих членів класу і логічну функцію, що визначає можливість побудови трикутника. Додатково визначити функцію-член цього класу, яка визначає довжини радіусів вписаного і описаного кіл цього трикутника.
2	Створити клас Circle для роботи з плоскими колами. Як член даного задається довжина радіусу кола. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену цього класу. Додатково визначити функцію-член цього класу, яка обчислює площу круга.
3	Розробити клас Line для роботи з об'єктами типу відрізок. Члени-дані цього класу визначають координати кінців відрізка на площині. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Описати функцію-член класу, що визначає довжину відрізка.
4	Створити клас Circle для роботи з плоскими колами. Як член даного задається довжина радіусу кола. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену цього класу. Додатково визначити функцію-член цього класу, яка обчислює довжину кола.
5	Розробити клас ThreeAngle для роботи з плоскими трикутниками. В якості членів-даних задаються довжини трьох сторін трикутника. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, доступ до

	закритих членів класу і логічну функцію, що визначає можливість побудови трикутника. Додатково визначити функцію-член цього класу, яка обчислює довжини всіх медіан трикутника.
6	Розробити клас Polygonдля роботи справільнимі замкнутими багатокутниками. Члени-дані цього класу визначають число сторін багатокутника і довжину сторони. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Описати функцію-член класу, яка обчислює площу багатокутника. Вивести отримане значення на екран дисплея.
7	Розробити клас Cylinder для роботи з циліндром. Члени-дані цього класу визначають радіус основи циліндра і висоту циліндра. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Описати функцію-член класу, яка обчислює площу повної поверхні циліндра.
8	Розробити клас Cone для роботи з конусом. Члени-дані цього класу визначають радіус підстави конуса і висоту конуса. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Описати функцію-член класу, яка обчислює обсяг конуса.
9	Розробити клас Sphere для роботи з кулею. Єдиний член-дане цього класу визначає радіус кулі. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену класу. Описати функцію-член класу, яка обчислює об'єм кулі.

1.5 Вимоги до звіту

- Назва роботи.
- Мета роботи.
- Тексти програм з коментарями.
- Результати виконання програм у вигляді копій консолі.
- Висновки.

ЛАБОРАТОРНА РОБОТА №2

КОНСТРУКТОРИ ТА ДЕСТРУКТОРИ

Мета роботи:

- Познайомитися з конструкторами та деструкторами.
- Навчитися створювати класи з ініціалізацією даних при створенні об’єктів класу.

2.1Теоретичні відомості

Конструктор - це функція-член класу, яка автоматично викликається при оголошенні об'єкту цього класу. Конструктор призначений для ініціалізації всіх або тільки деяких членів-даних об'єкта. Конструктори визначаютьсятак, як і будь-які інші функції, за винятком двох відмінностей:

- Конструктор повинен мати те ж ім'я, що і сам клас.
- Конструктор не може повертати значення. Більш того, в прототипі або заголовку визначення конструктора ніякої тип (навіть void) не вказується.

Конструктор, оголошений без аргументів, називається конструктором за замовчуванням (default constructor).

Конструктор при ініціалізації об'єктів може викликатися або неявно, що робиться частіше, або явно, використовуючи ім'я конструктора:

Date today (15); // Неявний виклик конструктора
Date yesterday = Date (14, 07); // Явний виклик конструктора

Однією з найважливіших форм перевантажуваного конструктора є конструктор копіювання (copyconstructor), який служить для отримання точних копій вже існуючого об'єкта. Зокрема конструктор копіювання викликається, коли об'єкт передається функції за значенням, при побудові тимчасового об'єкта як значення, що повертається функції, а також при використанні об'єкта для ініціалізації іншого об'єкта.

Будь-конструктор копіювання має таку загальну форму:

Ім'я_класу (const ім'я_класу &obj);

Параметри конструктору можна передати за допомогою списку ініціалізації членів. Цей список слід відразу за списком формальних параметрів і відділяється від нього двокрапкою. Окремий елемент списку є ім'я ініціалізіруемих члена-даного, за яким в круглих дужках задається инициализирующее вираз.

Якщо член-дане є посиланням або має специфікацію const, то його можна форматувати тільки за допомогою списку ініціалізації.

Для будь-якого створюваного класу рекомендується створювати три конструктора: головний архітектор, конструктор за замовчуванням і конструктор копіювання.

Деструктори (destructor) - з тієї ж категорії, що і конструктори. Вони використовуються для виконання певних операцій при видаленні об'єкта. Деструкція викликається автоматично при виході об'єкта з області видимості. Деструкція має ім'я, завжди збігається з ім'ям класу, але передує операцією доповнення (~).

Деструкція в С ++ має такі особливості і підпорядковується наступним правилам:

- У деструкції не визначається тип значення, навіть тип void;
- Деструкція не повинен мати параметрів;

Кожен об'єкт даного класу зберігає свою копію членів-даних, але для всіх об'єктів викликається один і той же варіант функції-члена. Для того, щоб не плутати такі об'єкти, у кожної функції-члена є неявний прихований покажчик this. Покажчик this - це зумовлений компілятором покажчик, який завжди вказує на об'єкт класу, для якого викликається функція-член цього класу. Він оголошується компілятором наступним чином:

Ім'я_класу * const this;

Оскільки this - це ключове слово, то його не можна оголосити явно, але явно користуватися ним можна. Змінити значення цього покажчика також не можна, оскільки він оголошений як константний покажчик.

2.2 Клас, що описує тварину

```
#include <iostream>
#include <conio.h>
#include <windows.h>
#include <string>
#include <stdlib.h>
using namespace std;
/ * Клас описує тварину * /
class Animal
{
string animalName; // Ім'я тварини
double weight; // Вага тварини
double age; // Вік тварини
public:
Animal (string name, double w, double a): animalName (name), weight (w), age (a)
//Основний конструктор
{if (weight <0)
weight = 0;
if (age <0)
age = 0;}
Animal (const Animal & obj) // Конструктор копіювання
{
animalName = obj.animalName;
```

```
weight = obj.weight;
age = obj.age;
}
string GetAnimalName () const
{
return animalName;
}
double GetAnimalWeight () const
{
return weight;
}
double GetAnimalAge () const
{
return age;
}
bool Eat (unsigned food) // Метод "годування" - ви годуєте тварина
{
int i = rand ();
i% = 2;
if (i) // воно може бути ситим і не буде їсти
{
Weight + = food;
cout << "Ваша тварина з'їла" << food << "кг їжі. \ n";
return true;
}
cout << "Ваша тварина ще не зголодніла. \ n";
return false;
}
void SetAnimalName (string s)
{
animalName = s;
}
bool SetAnimalAge (double a)
{
if (a> = 0)
{
Age = a;
return true;
}
return false;
}
void Output () const
{
cout << "Ім'я тварини:" << animalName <<
"\nВага:" << weight <<
"\nВік:" << age << endl;
}
};
int main ()
{
// Налаштування шрифтів і регіональних стандартів:
if (SetConsoleCP (1251) == 0)
```

```
// перевірка правильності установки кодування символів для введення
{
cerr << "Fialed to set codepage!" << endl;
/ * Якщо не вдалося встановити кодову сторінку, поява повідомлення про помилку * /
}
if (SetConsoleOutputCP (одна тисяча двісті п'ятьдесят одна) == 0)
{
cerr << "Failed to set OUTPUT page!" << endl;
}
Animal a ( "Кішка", 10, 0.5);
a.Output ();
for (int i = 0; i <10; i ++)
a.Eat (1);
a.Output ();
Animal b (a);
b.SetAnimalName ( "Собака");
b.Output ();
b.Eat (5);
b.Output ();
_getch ();
return 0;
}
```

2.3 Клас, що описує час

```
#include <iostream>
#include <conio.h>
#include <windows.h>
using namespace std;
class CTime
{// Елементиданих: години, хвилини, секунди:
WORD hour, min, sec;
// функції:
public:
/ * Завдяки значень за замовчуванням можна об'єднати два конструктора (за замовчуванням і основний) в один: * /
CTime (WORD _hour = 0, WORD _min = 0, WORD _sec = 0)
{
if ((_ hour <0 || _hour> 23) || (_min <0 || _min> 59) || (_sec <0 || _sec> 59))
{
SYSTEMTIME t;
GetLocalTime (& t);
_hour = t.wHour;
_min = t.wMinute;
_sec = t.wSecond;
}
hour = _hour; min = _min, sec = _sec;
}
// конструкторкопіювання:
CTime (const CTime & tm)
{
```

```
hour = tm.hour; min = tm.min, sec = tm.sec;
}
// функціївведення-виведення:
void Input ()
{
do
{
cout << "Задайте час (години хвилини секунди)";
cin >> hour >> min >> sec;
} While ((hour <0 || hour> 23) || (min <0 || min> 59) || (sec <0 || sec> 59));
}
void Output ()
{
cout << hour << ":" << min << ":" << sec << endl;
}
// функціїдоступу:
int GetHour () {return hour;}
int GetMin () {return min;}
int GetSec () {return sec;}
void SetHour (int newhour)
{
if (newhour <0 || newhour> 23)
{
SYSTEMTIME t;
GetLocalTime (& t);
hour = t.wHour;
}
else
hour = newhour;
}
void SetMin (int newmin)
{
if (newmin <0 || newmin> 59)
{
SYSTEMTIME t;
GetLocalTime (& t);
min = t.wMinute;
}
else
min = newmin;
}
void SetSec (int newsec)
{
if (newsec <0 || newsec> 59)
{
SYSTEMTIME t;
GetLocalTime (& t);
sec = t.wSecond;
}
else
sec = newsec;
}
```

```
// функція, яказчитуєсистемнийчасвпоточнийоб'єкт:
void TodayTime ()
{
SYSTEMTIME t; / * Структуратипу SYSTEMTIME визначенавзаголовочному файлі<windows.h> * /
GetLocalTime (& t);
/ * Зчитуємо системний час за допомогою функції, оголошеної в <windows.h>, при цьому заповнюється структура t * /
/ * Встановлюємо поточний час, використовуючи елементи структури t: * /
hour = t.wHour;
min = t.wMinute;
sec = t.wSecond;
}
/ * Функція, яка встановлює новий системний час, використовуючи елементи даних поточного об'єкта: * /
void SetTime ()
{
SYSTEMTIME t; / * Структура типу SYSTEMTIME, яка потрібна функції SetLocalTime (), яка встановлює системний час: * /
/ * Заповнюємо структуру t, використовуючи елементи даних поточного об'єкта і місцевий час: * /
GetLocalTime (& t);
t.wHour = hour;
t.wMinute = min;
t.wSecond = sec; // сотічасткисекунди
SetLocalTime (& t);
}
};
// прикладпрограми, щовикористовуєклас CTime:
int main ()
{
// Налаштуванняшрифтівірегіональнихстандартів:
if (SetConsoleCP (1251) == 0)
// перевірка правильності установки кодування символів для введення
{
cerr << "Fialed to set codepage!" << endl;
/ * Якщо не вдалося встановити кодову сторінку, поява повідомлення про помилку * /
}
if (SetConsoleOutputCP (одна тисяча двісті п'ятьдесят одна) == 0)
{
cerr << "Failed to set OUTPUT page!" << endl;
}
CTime tm1, tm2;
tm1.Input ();
cout << "Визначчас:";
tm1.Output ();
tm2.TodayTime ();
cout << "Системнийчас:";
tm2.Output ();
// Корегуємо кількість хвилин в системному часі:
WORD mn;
cout << "Задайте хвилини";
```

```
cin >> mn;
tm2.SetMin (mn);
tm2.SetTime ();
tm2.TodayTime ();
cout << "Новийсистемнийчас:";
tm2.Output ();
_getch ();
return 0;
}
```

2.4 Завдання на лабораторну роботу

Для захисту лабораторної роботи необхідно виконати 2 завдання із розділів 2.4.1 та 2.4.2. Завдання виконуються в залежності від варіанта. Варіант визначається за останньою цифрою залікової книжки.

2.4.1 Завдання 1

Таблиця 2.1 – Завдання 1 для самостійної роботи

Варіант	Завдання
0	Розробити клас Integer для роботи з цілими числами. Клас повинен містити основний конструктор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену класу. Крім цього визначити функцію-член, що визначає являється число простим чи ні.
1	Розробити клас Real для роботи з дійсними числами. Клас повинен містити основний конструктор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену класу. Крім цього визначити функцію-член, суму цифр цілої та дробової частини числа.
2	Розробити клас String для роботи з рядками. Клас повинен містити основний конструктор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член, яка з двох рядків створює третій рядок, що містить загальні символи першогота другого рядків без їх дублювання.
3	Розробити клас Byte для роботи з беззнаковими цілими числами. Клас повинен містити основний конструктор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену класу. Крім цього визначити функцію-член класу, яка обчислює восьмеричний еквівалент беззнакового

	цілого числа.
4	Розробити клас Complex для роботи з комплексними числами. Клас повинен містити основний конструктор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функції-члени класу, які задають основні арифметичні операції з комплексними числами (додавання, віднімання, множення і ділення комплексних чисел). Виконати всі арифметичні дії за допомогою цих функцій, відображаючи результат на екрані дисплея.
5	Розробити клас Date для роботи з датами. Клас повинен містити конструктор за замовчуванням та конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член TodayDate (), яка повертає системну дату об'єкту класу Date.
6	Розробити клас Point для роботи з об'єктами типу точка на площині. Клас повинен містити основний конструктор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член класу, що дозволяє вивести на екран дисплея координати середини лінії, що з'єднує дві задані точки.
7	Розробити клас String для роботи з рядками. Клас повинен містити основний конструктор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член, яка за двома даними рядках створює третій рядок, що є конкатенацією вихідних рядків.
8	Розробити клас Time для роботи з тимчасовими параметрами. Клас повинен містити основний конструктор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член TodayTime (), яка повертає системний час об'єкту класу Time.
9	Розробити клас String для роботи з рядками. Клас повинен містити основний конструктор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Крім цього визначити функцію-член, яка за двома даними рядках створює третій рядок, що є

	конкатенацієюінвертованих вихідних рядків.

2.4.2Завдання 2

Таблиця 2.2 – Завдання 2 для самостійної роботи

Варіант	Завдання
0	Створити клас Circle для роботи з плоскими колами. Як член даного задається довжина радіусу кола. Клас повинен містити основний конструктор, конструктор-архітектор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену цього класу. Додатково визначити функцію-член цього класу, яка обчислює площу круга.
1	Розробити клас ThreeAngle для роботи з плоскими трикутниками. В якості членів-даних задаються координати вершин трикутника. Клас повинен містити основний конструктор, конструктор-архітнектор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, доступ до закритих членів класу і логічну функцію, що визначає можливість побудови трикутника. Додатково визначити функцію-член цього класу, яка визначає приналежність точки з заданими координатами трикутнику.
2	Розробити клас Line для роботи з об'єктами типу відрізок. Члени-дані цього класу визначають координати кінців відрізка на площині. Клас повинен містити основний конструктор, коструктор-архітектор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Описати функцію-член класу, що визначає довжину відрізка.
3	Створити клас Circle для роботи з плоскими колами. Як член-дане задається довжина радіусу кола. Клас повинен містити основний конструктор, конструктор-архітектор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену цього класу. Додатково визначити функцію-член цього класу, яка обчислює довжину кола.
4	Розробити клас Sphere для роботи з кулею. Єдиний член-дане цього класу визначає радіус кулі. Клас повинен містити основний конструктор, коструктор-архітектор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену класу. Описати функцію-член класу, яка

	обчислює площу кульової поверхні.
5	Розробити клас Cylinder для роботи з циліндром. Члени-дані цього класу визначають радіус основи циліндра і висоту циліндра. Клас повинен містити основний конструктор, конструктор-архітектор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Описати функцію-член класу, яка обчислює об’єм циліндра.
6	Розробити клас Cone для роботи з конусом. Члени-дані цього класу визначають радіус підстави конуса і висоту конуса. Клас повинен містити основний конструктор, конструктор-архітектор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Описати функцію-член класу, яка обчислює об’єм конуса.
7	Розробити клас Pyramid для роботи з правильною пірамідою. Члени-дані цього класу визначають число сторін підстави піраміди, довжину одного боку і висоту піраміди. Клас повинен містити основний конструктор, конструктор-архітектор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритих членів класу. Описати функцію-член класу, яка обчислює обсяг піраміди.
8	Розробити клас Sphere для роботи з кулею. Єдиний член-дане цього класу визначає радіус кулі. Клас повинен містити основний конструктор, конструктор-архітектор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену класу. Описати функцію-член класу, яка обчислює об'єм кулі.
9	Створити клас Circle для роботи з плоскими колами. Як член-дане задається довжина радіусу кола. Клас повинен містити основний конструктор, конструктор-архітектор, конструктор копіювання. Визначити в цьому класі функції-члени, які забезпечують введення / виведення елементів класу, а також доступ до закритого члену цього класу. Додатково визначити функцію-член цього класу, яка обчислює центральний кут для хорди, довжина якої вводиться з клавіатури.

2.5 Вимоги до звіту

- Назва роботи.
- Мета роботи.
- Тексти програм з коментарями.
- Результати виконання програм у вигляді копій консолі.
- Висновки.

ЛАБОРАТОРНА РОБОТА №3

СПАДКУВАННЯ

Мета роботи:
- Познайомитися з одним із принципів ООП: спадкування.
- Навчитися на практиці застосовувати спадкування класів.

3.1 Теоретичні відомості

Спадкування - один з чотирьох найважливіших механізмів об'єктно-орієнтованого програмування (поряд з інкапсуляцією, поліморфізмом і абстракцією), дозволяють описати новий клас на основі вже існуючого (батьківського), при цьому властивості і функціональність батьківського класу запозичуються новим класом.

Класи найчастіше будуються поступово, починаючи від базових класів і закінчуючи класами будь-якої складності. Кожен раз, коли з попереднього класу проводиться наступний, то похідний клас успадковує якісь або всі батьківські якості, додаючи до них нові.

Мова С ++ відрізняється від інших об'єктно-орієнтованих мов ще й тим, що допускає не тільки просте спадкування, але і множинне спадкування, коли даний клас породжується відразу від декількох батьківських класів, наслідуючи поведінку всіх своїх предків.

Маючи добре розроблений базовий клас, потрібно пам'ятати і налагоджувати тільки ті зміни, які були внесені в похідні класи. Наслідуючи характеристики базового класу, можна змусити породжений клас розширити, звузити, змінити, знищити або використовувати ці характеристики без зміни.

3.1.1 Просте спадкування

Перш ніж будувати похідний клас, слід визначити, чи можна в якості базового класу використовувати вже існуючий клас? Часто вже є класи, які мають майже всі необхідні параметри. Саме вони і є кращими кандидатами в батьки.

При створенні екземпляра похідного класу автоматично викликається конструктор базового класу. Після того як об'єкт сконструйований, конструктор базового класу стає недоступним. Точно так само і деструктори призначені тільки для автоматичного виклику при видаленні об'єкта. Програма явно не дозволено явно виклткають деструктор.

Ім'я базового класу записується після імені породженого класу, відділяючись від нього двокрапкою.

Коли клас породжується від базового, все імена базового класу в похідному класі автоматично стають приватними. Саме така поведінка в С++ за умовчанням. Але його легко змінити, вказавши наступні специфікатори доступу для базового класу:

- private - (специфікатор за замовчуванням, якщо іншого при оголошенні класу не вказано). Всі успадковані імена базового класу з будь-якою формою доступу стають приватними в похідному класі;

- public - все загальнодоступні імена базового класу будуть загальнодоступними в похідному класі, і всі захищені імена базового класу будуть захищеними і в похідному класі.

- protected - все загальнодоступні члени базового класу стають захищеними в похідному класі. Захищені і закриті члени зберігають свої початкові специфікації доступу в похідному класі.

З огляду на, що в програмуванні на мову С ++ особлива увага приділяється насле¬дованію і повторного використання, програміст витрачає багато часу на разра¬ботку класів, які є базовими для інших класів. Привілеїв доступу private і public, які були встановлені для перших версій мови, виявилося явно недостатньо, і була введена ще одна привілей - protected. Будь-ідентифікатор, що з'явився в секції protected базового класу, доступний функцій класу (але не поль¬зователю). Цей привілей поширюється вниз по ієрархічному дереву не обмежена.

Базовий клас, похідний клас або обидва разом можуть мати конструктори і / або деструктори. Якщо і у базового, і у похідного класу є конструктори і де¬структори, то конструктори виконуються в порядку спадкування, а деструк¬тори - в зворотному порядку.

При ініціалізації об'єктів похідного класу часто виникає необхідність передати аргументи конструктору базового класу. В цьому випадку всенеобходімие аргументи базового і похідного класу передаються конструктору похідного класу, при цьому базовому класу аргументи передаються через список ініціалізації даних.

3.1.2 Множинне успадкування

Клас може мати багато батьків, наслідуючи властивості кожного з них. Природно, такий вид спадкування пов'язаний з певним ускладненням мови і компілятора.

Специфікатор доступу може бути різним для кожного базового класу, але його необхідно вказувати для кожного імені. Коли успадковується безліч базових класів, конструктори базових класів виконуються зліва направо в порядку, який задається в оголошенні похідного класу. Деструктори виконуються в зворотному порядку.

3.2 Класи, що описують точку, коло та конус

```
// файл point.h:
#include <iostream>
using namespace std;
// Клас, що описує точку:
class Point
{
```

```
protected:
double x, y;
private:
friend istream & operator >> (istream & is, Point & p)
{
cout << "Задайте координати точки";
return is >> p.x >> p.y;
}
friend ostream & operator << (ostream & os, const Point & p)
{
return os << "кооринати точки: [" <<
p.x << "," << p.y << "] \ n";
}
// ---------------------------------------------- ----------------
public:
Point (double _x = 0, double _y = 0)
{X = _x; y = _y;}
};
// файл circle.h:
// Клас, що описує коло:
#include "point.h"
#define _USE_MATH_DEFINES
#include <math.h>
class Circle: public Point // Клас Point є базовим для класу Circle
{
friend istream & operator >> (istream & is, Circle & c)
{
cout << "Задайте координати центру";
return is >> c.x >> c.y;
cout << "Задайте радіус";
is >> c.r;
}
friend ostream & operator << (ostream & os, const Circle & c)
{
return os << "Радіус:" << c.r << "," << Point (c.x, c.y);
// Приклад використання функції базового класу
}
protected:
double r; // радіус кола
public:
// конструктор:
Circle (double _x = 0, double _y = 0, double _r = 0): Point (_x, _y)
{R = _r;}
double Area ()
{Return M_PI * r * r;} // площа кола
};
// файл cone.h:
// клас, що описує конус
#define _USE_MATH_DEFINES
#include <math.h>
#include "circle.h"
class Cone: public Circle // клас Cone є похідним від класу Circle
```

```
{
double h; // висота конуса
friend istream & operator >> (istream &, Cone &);
friend ostream & operator << (ostream &, const Cone &);
public:
// Основний конструктор:
Cone (double _x = 0, double _y = 0, double _r = 0, double _h = 0):
Circle (_x, _y, _r) {h = _h; }
// Конструктор копіювання:
Cone (const Cone & cn) {h = cn.h;}
// Функції доступу до закритих членам:
double Get_h () const {return h;}
// Функції установки значень закритих членів:
void Set_h (double _h) {h = _h;}
double Volume () {return h * Area () / 3; } // обсяг конуса
};
// Визначення функцій:
istream & operator >> (istream & is, Cone & cn)
{
cout << "Задайте координати центру підстави і радіус підстави";
is >> cn.x >> cn.y >> cn.r;
cout << "Задайте висоту конуса";
return is >> cn.h;
}
ostream & operator << (ostream & os, const Cone & cn)
{
os << "Висота конуса:" << cn.h << "";
return os << Circle (cn.r, cn.x, cn.y) << endl;
}
// приклад використання класу:
#include <iostream>
#include <conio.h>
#include <windows.h>
#include <iomanip>
#include "cone.h"
using namespace std;
int main ()
{
// Налаштування шрифтів і регіональних стандартів:
if (SetConsoleCP (1251) == 0)
// перевірка правильності установки кодування символів для введення
{
cerr << "Fialed to set codepage!" << endl;
/ * Якщо не вдалося встановити кодову сторінку, поява повідомлення про помилку * /
}
if (SetConsoleOutputCP (одна тисяча двісті п'ятьдесят одна) == 0)
{
cerr << "Failed to set OUTPUT page!" << endl;
}
Point pt (200,200);
cout << pt;
cin >> pt;
```

```
cout << pt;
Circle cr (300,200,20);
cout << cr;
cin >> cr;
cout << cr << "площа кола" << cr.Area () << endl;
Cone cn (450, 200, 50, 90);
cout << "Об'єм конуса:" <<
setiosflags (ios :: fixed) << setprecision (2) << cn.Volume () << endl;
_getch ();
return 0;
}
```

3.3 Завдання на лабораторну роботу

Для захисту лабораторної роботи необхідно виконати 2 завдання із розділів 3.3.1 та 3.3.2. Завдання виконуються в залежності від варіанта. Варіант визначається за останньою цифрою залікової книжки.

3.3.1 Завдання 1

Таблиця 3.1 – Завдання 1 для самостійної роботи

Варіант	Завдання
0	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функції, які повертають довжину кола і площу круга.
1	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає відстань між центрами двох кіл.
2	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає відстань між центром окружності і початком координат.
3	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає площа трикутника, вершинами якого служать центри трьох заданих кіл.
4	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, що повертає довжину радіусу кола описаної навколо трикутника, вершинами якого служать центри трьох заданих кіл.

5	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає мінімальне відстань від початку координат до окружності.
6	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає довжини медіан трикутника, вершинами якого служать центри трьох заданих кіл.
7	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає довжини биссектрис трикутника, вершинами якого служать центри трьох заданих кіл.
8	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, що повертає довжину радіусу кола вписаною в трикутник, вершинами якого служать центри трьох заданих кіл.
9	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає площа трикутника, вершинами якого служать центри трьох заданих кіл.

3.3.2Завдання 2

Таблиця 3.2 – Завдання 2 для самостійної роботи

Варіант	Завдання
0	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Новий клас використовувати в якості базового для розробки класу Cylinder, що визначає різні циліндри. Визначити в цьому класі функцію, яка повертає площу повної поверхні циліндра.
1	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Новий клас використовувати в якості базового для розробки класу Cylinder, що визначає різні циліндри. Визначити в цьому класі функцію, яка повертає площа прямокутника, отриманого при осьовому перерізі циліндра.

2	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового класу, розробити похідний клас Polygon, що визначає правильні багатокутники. Визначити в цьому класі функцію, що визначає радіус кола, описаного навколо даного багатокутника.
3	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Новий клас використовувати в якості базового для розробки класу Cone, що визначає різні конуси. Визначити в цьому класі функцію, яка повертає площа трикутника, одержуваного при осьовому перерізі конуса.
4	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Новий клас використовувати в якості базового для розробки класу Cone, що визначає різні конуси. Визначити в цьому класі функцію, яка повертає обсяг конуса.
5	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового класу, розробити похідний клас Polygon, що визначає правильні багатокутники. Новий клас використовувати в якості базового для розробки класу Prizm, що визначає правильні призми. Визначити в цьому класі функцію, що визначає площу повної поверхні призми.
6	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового класу, розробити похідний клас Polygon, що визначає правильні багатокутники. Новий клас використовувати в якості базового для розробки класу Pyramid, що визначає правильні піраміди. Визначити в цьому класі функцію, що визначає обсяг піраміди.
7	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового класу, розробити похідний клас Polygon, що визначає правильні багатокутники. Новий клас використовувати в якості базового для розробки класу Prizm, що визначає правильні призми. Визначити в цьому класі функцію, що визначає обсяг призми.
8	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового класу, розробити похідний клас Polygon, що визначає правильні багатокутники. Визначити в цьому класі функцію, що визначає радіус кола, вписаного в даний багатокутник.
9	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового класу, розробити похідний клас Polygon, що визначає правильні багатокутники. Новий клас

	використовувати в якості базового для розробки класу Pyramid, що визначає правильні піраміди. Визначити в цьому класі функцію, що визначає площу повної поверхні піраміди.

3.4 Вимоги до звіту

- Назва роботи.
- Мета роботи.
- Тексти програм з коментарями.
- Результати виконання програм у вигляді копій консолі.
- Висновки.

ЛАБОРАТОРНА РОБОТА №4

ПОЛІМОРФІЗМ ТА ВІРТУАЛЬНІ ФУНКЦІЇ

Мета роботи:

- Познайомитися з одним із принципів ООП: поліморфізмом.
- Навчитися на практиці застосовувати поліморфізм.

4.1Теоретичні відомості

Поліморфізм це механізм, що дозволяє використовувати один і той же інтерфейс для об'єктів базових і похідних класів. В С ++ поліморфізм реалізують за допомогою віртуальних функцій.

Віртуальна функція (virtual function) є функцією-членом класу. Вона оголошується всередині базового класу і переопределяется в похідному класі.

Віртуальна функція оголошується за допомогою специфікатор virtual, зазначених вище перед типом повертається функцією значення. Після того, як Ви оголосили функцію віртуальної, її можна перевизначити тільки в похідному класі. Заміщають функції повинні мати той же список параметрів.

Коли в віртуальної функції базового класу відсутній значущу дію, в реалізації будь-якого похідного класу ця функція обов'язково повинна перевизначатися. Для реалізації цього положення в С ++ підтримуються чисто віртуальні функції (pure virtual function). Для чисто віртуальних функцій використовується така загальна форма прототипу:

virtual type Name (parameters) = 0;

Ключовим моментом цього оголошення є прирівнювання функції нулю. Це повідомляє компілятору, що в базовому класі не існує тіла функції.

Якщо клас містить хоча б одну чисто віртуальну функцію, то про нього говорять як про абстрактне класі. Абстрактний клас може бути тільки спадкоємною, оскільки в ньому міститься, принаймні, одна функція, у якій відсутнє тіло, тобто не можна створювати об'єкти такого класу.

Якщо оголошується покажчик на об'єкт, то при використанні перевантаження функцій, тіпуказателя визначає, з якого класу викликається функція. При використанні віртуальнихфункцій, клас викликається функції визначається не по типу покажчика, а по типу об'єкта, адреса якого зберігає цей покажчик.

4.2 Класи, що описують точку та коло

```
#include <iostream>
#include <conio.h>
#include <windows.h>
#define _USE_MATH_DEFINES
#include <cmath>
using namespace std;
```

```
// Клас, що описує точку:
class CPoint
{
protected:
int x, y; // координати точки
public:
// Конструктор:
CPoint (int _x = 320, int _y = 240)
{X = _x; y = _y; }
int GetX () {return x;}
virtual double Length () // Віртуальна функція повертає довжину.
// Для точки - довжина радіус вектора:
{
return sqrt ((double) (x * x + y * y));
}
};
// Клас, що описує коло (похідний від класу CPoint):
class CCircle: public CPoint
{
int r; // радіус кола
public:
CCircle () {r = 2;}
// Основний конструктор:
CCircle (int _x, int _y, int _r):
CPoint (_x, _y) // Явний виклик конструктора базового класу
{R = _r;}
double Length () {return 2 * M_PI * r;} // довжина кола
};
// Приклад використання класів:
int main ()
{
// Налаштування шрифтів і регіональних стандартів:
if (SetConsoleCP (1251) == 0)
// перевірка правильності установки кодування символів для введення
{
cerr << "Fialed to set codepage!" << endl;
/ * Якщо не вдалося встановити кодову сторінку, поява повідомлення про помилку * /
}
if (SetConsoleOutputCP (одна тисяча двісті п'ятьдесят одна) == 0)
{
cerr << "Failed to set OUTPUT page!" << endl;
}
CPoint ** ptr = new CPoint * [5]; / * Пам'ятайте, що покажчик на об'єкт похідного класу є
одночасно і покажчиком на об'єкт базового класу * /
// створюємо кілька об'єктів:
ptr [0] = new CPoint (100, 100);
ptr [1] = new CCircle (200,150, 10);
ptr [2] = new CPoint (100,200);
ptr [3] = new CCircle (200,250,10);
ptr [4] = new CPoint (100,300);
/ * Такого типу цикл можна написати тільки при використанні віртуальних функцій: * /
for (int i = 0; i <5; i ++)
```

```
cout << "Довжина об'єкта" << ptr [i] -> Length () << endl;
// одним оператором обробляємо об'єкти різних типів
// ----------------------------------
for (int i = 0; i <5; i ++)
delete ptr [i];
delete ptr;
_getch ();
return 0;
}
```

4.3 Завдання на лабораторну роботу

Для захисту лабораторної роботи необхідно виконати одне завдання із розділу4.3.1. Завдання виконуються в залежності від варіанта. Варіант визначається за останньою цифрою залікової книжки.

4.3.1 Завдання 1

Таблиця 4.1 – Завдання 1 для самостійної роботи

Варіант	Завдання
0	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функції повертають довжину кола і площа круга. В обох класах оголосити віртуальну функцію Length, яка повертає довжину відповідного об'єкта. У головній програмі оголосити масив покажчиків на об'єкти точок і кіл. Вивести середнє арифметичне довжин кіл і середнє арифметичне їх площ.
1	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає відстань між центрами двох кіл. В обох класах оголосити віртуальну функцію Length, яка повертає довжину відповідного об'єкта. У головній програмі оголосити масив покажчиків на об'єкти точок і кіл. Вивести середня відстань між центрами кіл.
2	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає відстань між центром окружності і початком координат. В обох класах оголосити віртуальну функцію Length, яка повертає довжину відповідного об'єкта. У головній програмі оголосити масив покажчиків на об'єкти і точок і кіл. Вивести режимі середня відстань від центрів кіл до початку координат.
3	Розробити клас Point для завдання координати точки на площині.

	Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає площа трикутника, вершинами якого служать центри трьох заданих кіл. В обох класах оголосити віртуальну функцію Length, яка повертає довжину відповідного об'єкта. У головній програмі оголосити масив покажчиків на об'єкти і оголосити кілька точок і 3 кола. Вивести площу трикутника, вершинами якого є центри заданих кіл.
4	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, що повертає довжину радіуса кола, описаного навколо трикутника, вершинами якого служать центри трьох заданих кіл. В обох класах оголосити віртуальну функцію Length, яка повертає довжину відповідного об'єкта. У головній програмі оголосити масив покажчиків на об'єкти і намалювати кілька точок і 3 кола. Вивести довжину радіуса кола, описаного навколо трикутника, вершинами якого є центри заданих кіл.
5	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає мінімальне відстань від початку координат до окружності. В обох класах оголосити віртуальну функцію Length, яка повертає довжину відповідного об'єкта. У головній програмі оголосити масив покажчиків на об'єкти і намалювати кілька точок і кілька кіл. Вивести мінімальне значення відстані від початку координат до кіл.
6	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. В обох класах оголосити віртуальну функцію Length, яка повертає довжину відповідного об'єкта. У головній програмі оголосити масив покажчиків на кілька точок і 3 кола. Визначити в цьому класі функцію, яка повертає довжини медіан трикутника, вершинами якого служать центри трьох заданих кіл і вивести ці довжини в головній програмі.
7	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, яка повертає довжини биссектрис трикутника, вершинами якого служать центри трьох заданих кіл. В обох класах оголосити віртуальну функцію Length, яка повертає довжину відповідного об'єкта. У головній програмі оголосити масив покажчиків на кілька точок і 3 кола. Вивести довжини

	биссектрис.
8	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового, розробити похідний клас Circle, що визначає кола різного радіусу. Визначити в цьому класі функцію, що повертає довжину радіусу кола вписаною в трикутник, вершинами якого служать центри трьох заданих кіл. В обох класах оголосити віртуальну функцію Length, яка повертає довжину відповідного об'єкта. У головній програмі оголосити масив покажчиків на кілька точок і 3 кола. Вивести довжину вписаного кола.
9	Розробити клас Point для завдання координати точки на площині. Вибираючи цей клас в якості базового класу, розробити похідний клас Rectangle, що визначає різні прямокутники зі сторонами, паралельними осям координат. Визначити в цьому класі функцію, яка повертає координати всіх вершин прямокутника. В обох класах оголосити віртуальну функцію Length, яка повертає довжину відповідного об'єкта. У головній програмі оголосити масив покажчиків на кілька точок і кілька прямокутників. Вивести центр мас всіх вершин прямокутників.

4.4 Вимоги до звіту

- Назва роботи.
- Мета роботи.
- Тексти програм з коментарями.
- Результати виконання програм у вигляді копій консолі.
- Висновки.

ЛАБОРАТОРНА РОБОТА №5

ПОТОКИ, ОБРОБКА ВИНЯТКОВИХ СИТУАЦІЙ

Мета роботи:

– Познайомитися з потоками: cout, cin, cerr, clog.
– Навчитися на практиці оброблювати виняткові ситуації.

5.1Теоретичні відомості

Виняток - це непередбачене або аварійна подія. В С ++ виняток це об'єкт, який система повинна генерувати при виникненні виняткової ситуації. Генерація такого об'єкта і створює виняткову ситуацію. Винятки дозволяють розділити обчислювальний процес на 2 частини:

1) виявлення аварійної ситуації (невідомо як обробляти);

2) обробка аварійної ситуації (невідомо, де вона виникла).

Переваги такого підходу:

1) зручно використовувати в програмі, яка складається з декількох модулів;

2) не потрібно повертати значення в зухвалу функцію.

Загальна схема:

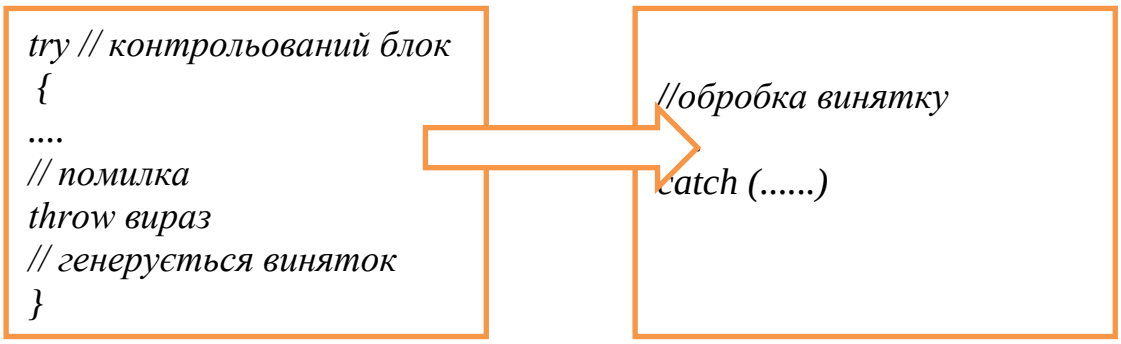

Рисунок 5.1 – Загальна схема виняткової ситуації

Виняток генерується оператором throw<вираз>, де <вираз>:

- або константа;
- або змінна деякого типу;
- або вираз деякого типу. Т

Тип об'єкта-виключення може бути як вбудованим, так і визначеним користувачем. Для уявлення винятків часто використовують порожній клас: *class ZeroDevide; class NegativeArg;*

Генерація виключення буде виглядати:

throwZeroDevide (); // викликається конструктор без параметрів

або

thrownewZeroDevide ();

Виняток треба перехопити і обробити. Для перевірки виникнення виключення використовується контрольований блок try, з яким пов'язана одна або кілька секцій-пасток catch. Всі змінні, оголошені в цьому блоці, є локальними для цього блоку.

Форма запису секції-пастки наступна: catch (специфікація виключення), де специфікація виключення може мати три форми:

1) (тип ім'я);

2) (тип);

3) ().

Тип - це вбудований тип або тип, визначений програмістом. Форми 1 і 2 обробляють конкретні винятки, а форма 3 перехоплює все виключення, таку пастку треба поміщати останньої, тоді вона буде обробляти всі винятки, які ще не були оброблені. Форма 1 означає, що об'єкт передається в блок обробки, щоб його якимось чином там використовувати, наприклад, для виведення інформації в повідомленні про помилку. Приклади:

catch (exception e) // за значенням

catch (exception & e) // за посиланням

catch (const exception & e) // за константним посиланням

*catch (exception * e) // за вказівником*

Найкраще передавати об'єкт за посиланням, при цьому не створюється тимчасовий об'єктісключеніе. Для кожної функції, методу, конструктора або деструктора можна в заголовку вказати специфікацію винятків. Якщо в заголовку специфікація винятків не вказана, вважається, що функція може породжувати будь-виключення, якщо вказана, то вважається, що функція генерує ті винятки, які явно зазначені у цьому списку. Приклади:

void f1 () throw (int, double);

void f2 () throw (zerodivide);

Якщо специфікація має вигляд:

void f () throw ();

то вважається, що функція винятків не генерує. Наявність специфікацій виключення не є обмеженням при реальній генерації виключень. Але якщо функція генерує неспеціфікований виняток, то запускається стандартна функція unexpected (), яка викликає функцію terminate (), що призводить до аварійного завершення програми. При відсутності підходящої секції-пастки здійснюється виклик стандартної функції завершення terminate (). Ця функція викликається з функції unexpected () при порушенні специфікації завершення. Обидві функції можна підмінити власними реалізаціями. Для цього необхідно:

1. Підключити заголовок #include <expection>

2. Визначити власну функцію з прототипом void F () для підміни стандартної функції terminate ().

3. Вказати ім'я цієї функції у виклику функції set_terminate (f); Після цього замість terminate () буде викликатися наша функція F (). Така функція не повинна повертати управління оператором return або генерувати виняток throw

(), вона може тільки завершити програму функцією exit () або abort (). Аналогічно реалізується підміна стандартної функції unexpected (): set_ unexpected (f); Функція може згенерувати неспеціфіцірованное виняток, в цьому випадку, якщо в специфікації винятків не вказано виняток bad_exception, викликається функція terminate (), в іншому випадку сгенерированное виняток підміняється на bad_exception і починається пошук його обробника.

У складі стандартної бібліотеки С ++ реалізований ряд стандартних винятків, які організовані в ієрархію класів. Ця ієрархія може служити основою для створення власних класів винятків і ієрархії виключень. Можна визначати власні виключення, успадкувавши їх від класу exception. Клас exception визначено в стандартній бібліотеці наступним чином:

class exception
{
public:
exception () throw (); // конструктор без параметрів
exception (const exception &) throw (); // конструктор копіювання
exception & operator = (const exception &) throw (); // оператор = virtual ~ exception () throw (); // деструктор
*virtual const char * what () const throw (); // генерує повідомлення про помилку;*
};

Всі конструктори і методи мають специфікацію, яка забороняє генерацію винятків. Передбачається, що виключення типу - logic_error - помилки в логіці програми, наприклад, невиконання якого-небудь умови; - runtime_error помилки виникають в результаті непередбачених обставин при виконанні програми, наприклад, переповнення при операціях з дробовими числами; Ці виключення програма повинна генерувати самостійно оператором throw.

5.2 Хід роботи

Реалізувати клас Вектор. Розмір вектора обмежений значенням MAX_SIZE =30.

Перевантажити для нього операції:

- доступ за індексом([inti]),
- додавання елемента(+ int),
- видалення елемента з початку вектора (--).

Передбачити генерацію виняткових ситуацій.

Виняткові ситуації генеруються:

1- в конструкторі з параметром при спробі створити вектор більше максимальногорозміру;

2, 3- в операції []- при спробі звернутися до елементу з індексом менше 0 або більшепоточного розміру вектора;

4- в операції +- при спробі додати елемент з індексом більше максимального

розміру;
5- в операції- при спробі видалити елемент з пустого вектора.

5.2.1 ВАРІАНТ РЕАЛІЗАЦІЇ 1

Інформація про виняткові ситуації передається за допомогою стандартного типуданих.

1.Створити порожній проект.

2.Додати в нього класVector.

3.У файлVector.h додатиопис класуVector:

```
#pragma once
#include <iostream>
usingnamespace std;
constintMAX_SIZE = 30; // максимальний розмір вектора
class Vector
{
intsize; // поточний розмір
int * beg; // покажчик на початок динамічного массиву
public:
Vector () {size = 0; beg = 0; } // конструктор без параметрів
Vector (ints); // конструктор з параметрами
Vector (ints, int * mas); // конструктор з параметром
Vector (constVector & v); // конструктор копіювання
~Vector (); // деструктор
const Vector & operator = (const Vector & v ); // операція присвоювання
int operator [] (int i); // доступ за індексом
Vector operator+ (inta); // додавання елемента
Vectoroperator-- (); // видалення елемента
// дружні функції введення-виведення
friend ostream & operator << (ostream & out, const Vector & v);
friend istream & operator >> (istream & in, Vector & v);};
```

4.У файл Vector.cpp додати визначення методів класу Vector:

```
Vector::Vector(int s)
{
// якщо поточний розмір більше максимального, то генерується виняток
if(s>MAX_SIZE) throw1;
size=s;
beg=newint[s];
for(int i=0;i<size;i++)
beg[i]=0;
}
Vector::Vector(const Vector &v)
{
size=v.size;
beg=newint [size];
for(int i=0;i<size;i++)
```

```
beg[i]=v.beg[i];
}
Vector::~Vector()
{
if (beg!=0) delete[]beg;
}
Vector::Vector(int s,int *mas)
{
// якщо поточний розмір більше максимального, то генерується виняток
if(s>MAX_SIZE) throw1;
size=s;
beg=newint[size];
for(int i=0;i<size;i++)
beg[i]=mas[i];
}
const Vector& Vector::operator =(const Vector &v)
{
if(this==&v)return *this;
if(beg!=0) delete []beg;
size=v.size;
beg=newint [size];
for(int i=0;i<size;i++)
beg[i]=v.beg[i];
return*this;
}
ostream& operator<<(ostream&out, const Vector&v)
{
if(v.size==0) out<<"Empty\n";
else
{
for(int i=0;i<v.size;i++)
out<<v.beg[i]<<" ";
out<<endl;
}
return out;
}
istream& operator >>(istream&in, Vector&v)
{
for(int i=0;i<v.size;i++)
{
cout<<">";
in>>v.beg[i];
}
return in;
}
int Vector::operator [](int i)
{
if(i<0)throw2;
// якщо індекс негативний, то генерується виняток
// якщо індекс більше розмір вектора, то генерується виняток
if(i>=size)throwerror("Vector length more than MAXSIZE\n");
return beg[i];
```

```
}
Vector Vector::operator+(int a)
{
// якщо при додаванні елемента розмір вектора побільшає
// максимального,
// то генерується виняток
if(size+1==MAX_SIZE) throw4;
Vector temp(size+1,beg);
temp.beg[size]=a;
return temp;
}
Vector Vector::operator--()
{
// якщо вектор порожній, то видалити елемент не можна і генерується виняток
if(size==0) throw5;
if (size==1)
{
size=0;
delete[]beg;
beg=0;
return *this;
};
Vector temp(size,beg);
delete[]beg;
size--;
beg=newint[size];
for(int i=0;i<size;i++)
beg[i]=temp.beg[i];
return*this;
}
```

5. Додати в проект файл lab5_main.cpp. У файл записати функцію main(), що створює об'єкти класаVector і дозволяє генерувати виняткові ситуації.

```
#include "Vector.h"
#include <iostream>
Using namespace std;
int main ()
{
// контрольований блок
try
{
Vectorx (2);
// вектор з двох елементів
Vectory;
// порожній вектор
cout<<x;
// друк вектора x
cout << "Nomer?";
int i;
cin >> i;
```

```
// вивід елемента з номером i, якщо номер більше 2 або менше 0, то
// генерується виняткова ситуація
cout<<x [i] <<endl;
// додавання елемента в вектор, якщо MAX_SIZE = 2, то генерується
// виняткова ситуація
y = x + 3;
cout<<y;
// видалити один елемент з вектора
--x;
cout<<x;
// видалити один елемент з вектора
--x;
cout<<x;
// вектор порожній
// видалити один елемент з вектора
// генерується виняткова ситуація
--x;
}
// обробник виключення
catch (int)
{Cout<< "ERROR !!!" <<endl;} // повідомлення про помилку
return 0;
}
```

6.Виконати тестування програми з генерацією різних виняткових ситуацій.

5.2.2 ВАРІАНТ РЕАЛІЗАЦІЇ 2

Інформація про виняткові ситуації передається за допомогою призначеного для користувача класу.

1.Створити порожній проект.

2.Додати в нього файлerror.h.

3.У файлerror.h додатиопис класуerror:

```
#pragma once
#include <string>
#include <iostream>
using namespace std;
class error //классошибка
{
  string str;
public:
//конструктор, ініціюєатрибут str повідомленнямпропомилку
  error(string s){str=s;}
  void what(){cout<<str<<endl;} //виводитьзначенняатрибута str
};
```

4. Додати клас Vector. У файл Vector.h додати опис класу Vector:

```
#pragma once
```

```
#include <iostream>
using namespace std;
const int MAX_SIZE = 20;
class Vector
{
    int size;
    int * beg;
public:
    Vector () {size = 0; beg = 0;}
    Vector (int s);
    Vector (int s , int * mas);
    Vector (const Vector & v);
    ~Vector ();
    const Vector & operator = (const Vector & v);
    int operator [] (int i);
    Vector operator+ (int a);
    Vector operator - ();
friend ostream & operator << (ostream & out, constVector & v);
friend istream & operator >> (istream & in, Vector & v);};
```

5. У файл Vector.c ++ додати визначення методів класу Vector:

```
include "Vector.h"
#include "Error.h"
#include <iostream>
using namespace std;
Vector::Vector(int s)
{
    if(s>MAX_SIZE) throw error("Vector length more than MAXSIZE\n");
    size=s;
    beg=new int[s];
    for(int i=0;i<size;i++)
       beg[i]=0;
}
Vector::Vector(const Vector &v)
{
    size=v.size;
    beg=new int[size];
    for(int i=0;i<size;i++)
       beg[i]=v.beg[i];
}
Vector::~Vector()
{
    if(beg!=0) delete []beg;
}
Vector::Vector(int s,int *mas)
{
    if(s>MAX_SIZE) throw error("Vector length more than MAXSIZE\n");
    size=s;
    beg=new int[size];
    for(int i=0;i<size;i++)
```

```
        beg[i]=mas[i];
}
const Vector& Vector::operator =(const Vector &v)
{
    if(this==&v)return *this;
    if(beg!=0) delete []beg;
    size=v.size;
    beg=new int[size];
    for(int i=0;i<size;i++)
        beg[i]=v.beg[i];
    return *this;
}
ostream& operator<<(ostream&out, const Vector&v)
{
    if(v.size==0) out<<"Empty\n";
    else
    {
      for(int i=0;i<v.size;i++)
      out<<v.beg[i]<<" ";
      out<<endl;
    }
    return out;
}
istream& operator >>(istream&in, Vector&v)
{
    for(int i=0;i<v.size;i++)
    {
        cout<<">";
        in>>v.beg[i];
    }
    return in;
}
int Vector::operator [](int i)
{
    if(i<0) throw error("index <0");
    if(i>=size) throw error("index>size");
    return beg[i];
}
Vector Vector::operator+(int a)
{
if(size+1==MAX_SIZE) throw 4;
Vector temp(size+1,beg);
temp.beg[size]=a;
return temp;
}
Vector Vector::operator--()
{
if(size==0) throw error("Vector is empty");
if(size==1)
{
size=0;
delete[]beg;
```

```
beg=0;
return *this;
};
Vector temp(size,beg);
delete[]beg;
size--;
beg=newint[size];
for(int i=0;i<size;i++)
beg[i]=temp.beg[i];
return*this;
}
```

6. Додати в проект файл lab5_main.cpp. У файл записати функцію main (), що створює об'єкти класу Vector і дозволяє генерувати виняткові ситуації.

```
#include "Vector.h"
#include "Error.h"
#include <iostream>
using namespace std;
int main()
{
    try
    {
    Vector x(2);
    Vector y;
    cout<<x;
    cout<<"Nomer?";
    int i;
    cin>>i;
    cout<<x[i]<<endl;
    y=x+3;
    cout<<y;
    --x;
    cout<<x;
    --x;
    cout<<x;
    --x;
   }
   catch(error e)
   {e.what();}
   return 0;
}
```

7. Виконатитестуванняпрограмизгенерацієюрізнихвинятковихситуацій.

5.2.3 ВАРІАНТ РЕАЛІЗАЦІЇ 3

Інформація про виняткові ситуації передається за допомогою ієрархії призначених для користувача класів.

1.Створити порожній проект.

2.Додати в нього файл error.h.

3.У файл error.h додати опис ієрархії призначених для користувача класів для визначення виняткових ситуацій.

```
#pragma once
#include <string>
#include <iostream>
using namespace std;
class Error //основний клас
{
public:
virtualvoidwhat(){};
};
classIndexError:publicError // помилка в індексі вектора
{
protected:
stringmsg;
public:
IndexError(){msg = "IndexError \ n";}
Virtualvoidwhat(){cout<<msg;}
};
classSizeError:publicError //помилка в розмірі вектора
{
protected:
stringmsg;
public:
SizeError () {msg = "sizeerror \ n ";}
Virtualvoidwhat(){cout<<msg;}};
classMaxSizeError:publicSizeError //перевищено максимальний розмір
{
protected:
stringmsg_;
public:
MaxSizeError(){SizeError();msg_="size>MAXSIZE\n";}
virtualvoidwhat(){cout<<msg<<msg_;}
};
classEmptySizeError:publicSizeError//видалення з пустого вектора
{
protected:
stringmsg_;
public: EmptySizeError(){SizeError();msg_="Vectorisempty\n";}
virtualvoidwhat(){cout<<msg<<msg_;}
};
classIndexError1:publicIndexError //індекс менше нуля
{
protected:
stringmsg_;
public:
IndexError1(){IndexError();msg_="index<0\n";}
    virtualvoidwhat(){cout<<msg<<msg_;}
};
```

```
classIndexError2: publicIndexError // індекс більше заданого розміру //вектора
{
protected:
stringmsg_;
public:
IndexError2(){IndexError();msg_="index>size\n";}
virtualvoidwhat(){cout<<msg<<msg_;}
};
```

4. Додати клас Vector. У файл Vector.h додати опис класу Vector:

```
const int MAX_SIZE = 20;
class Vector
{
    int size;
    int * beg;
public:
    Vector () {size = 0; beg = 0;}
    Vector (int s);
    Vector (int s, int * mas);
    Vector (const Vector & v );
    ~Vector ();
    const Vector & operator = (const Vector & v);
    int operator [] (int i);
    Vector operator + (int a);
     Vector operator- ();
     friend ostream & operator << (ostream & out, const Vector & v);
     friend istream & operator >> (istream & in, Vector & v);};
```

5. У файл Vector.c ++ додати визначення методів класу Vector:

```
#include "Vector.h"
#include "Error.h"
#include <iostream>
using namespace std;
Vector::Vector(int s)
{
    if(s>MAX_SIZE) throw MaxSizeError();
    size=s;
    beg=new int[s];
    for(int i=0;i<size;i++)
       beg[i]=0;
}
Vector::Vector(const Vector &v)
{
    size=v.size;
    beg=new int[size];
    for(int i=0;i<size;i++)
       beg[i]=v.beg[i];
}
Vector::~Vector()
```

```
{
    if(beg!=0) delete[]beg;
}
Vector::Vector(int s,int *mas)
{
    size=s;
    beg=new int[size];
    for(int i=0;i<size;i++)
       beg[i]=mas[i];
}
const Vector& Vector::operator =(const Vector &v)
{
    if(this==&v)return *this;
    if(beg!=0) delete[]beg;
    size=v.size;
    beg=new int[size];
    for(int i=0;i<size;i++)
       beg[i]=v.beg[i];
    return *this;
}
ostream& operator<<(ostream&out, const Vector&v)
{
    if(v.size==0) out<<"Empty\n";
    else
    {
    for(int i=0;i<v.size;i++)
       out<<v.beg[i]<<" ";
    out<<endl;
    }
    return out;
}
istream& operator >>(istream&in, Vector&v)
{
    for(int i=0;i<v.size;i++)
    {
       cout<<">";
       in>>v.beg[i];
    }
    return in;
}
int Vector::operator [](int i)
{
    if(i<0)throw IndexError1();
    if(i>=size) throw IndexError2();
    return beg[i];
}
Vector Vector::operator+(int a)
{
    if(size+1==MAX_SIZE) throw MaxSizeError();
    Vector temp(size+1,beg);
    temp.beg[size]=a;
    return temp;
```

```
}
Vector Vector::operator--()
{
    if(size==0) throw EmptySizeError();
    if(size==1)
    {
      size=0;
      delete[]beg;
      beg=0;
      return *this;
    };
    Vector temp(size,beg);
    delete[]beg;
    size--;
    beg=new int[size];
    for(int i=0;i<size;i++)
       beg[i]=temp.beg[i];
    return *this;
}
```

6. Додати в проект файл lab5_main.cpp. У файл записати функцію main (), що створює об'єкти класу Vector і дозволяє генерувати виняткові ситуації.

```
#include"Vector.h"
#include"Error.h"
#include<iostream>
usingnamespace std;
int main()
{
try
{
Vector x(2);
Vector y;
cout<<x;
cout<<"Nomer?";
int i;
cin>>i;
cout<<x[i]<<endl;
y=x+3;
cout<<y;
--x;
cout<<x;
--x;
cout<<x;
--x;
}
catch(Error &e)
{
e.what();
}
return 0;
```

}

5.3 Завдання на лабораторну роботу

Таблиця 5.1 – Завдання на лабораторну роботу

Варіант	Завдання	Варіант реалізації
0	Клас-контейнер ВЕКТОР з елементами типу int. Реалізувати операції: [] - доступу за індексом; () - визначення розміру вектора; + число - додає константу до всіх елементів вектора; -n - видаляє n елементів з кінця вектора.	1, 2
1	Клас-контейнер ВЕКТОР з елементами типу int. Реалізувати операції: [] - доступу за індексом; int() - визначення розміру вектора; -n - видаляє n елементів з кінця вектора; +n-додає n елементів вкрай вектора.	2, 3
2	Клас-контейнер ВЕКТОР з елементами типу int. Реалізувати операції: [] - доступу за індексом; ++ - додає елемент в вектор (постфіксна операція додає елемент в кінець, префіксна в початок)	3, 1
3	Клас-контейнер ВЕКТОР з елементами типу int. Реалізувати операції: [] - доступу за індексом; () - визначення розміру вектора; --- видаляє елемент з вектора (постфіксна операція видаляє елемент в кінець вектора, префіксна - до початку)	1, 2
4	Клас-контейнер ВЕКТОР з елементами типу int. Реалізувати операції: [] - доступу за індексом; int() - визначення розміру вектора; *вектор - множення елементів векторів a[i] * b[i]; +n - перехід вправо до елементу з номером n.	2, 3
5	Клас-контейнер МНОЖИНА з елементами типу int. Реалізувати операції: [] - доступу за індексом; () - визначення розміру множини; + - об'єднання множин; ++ - додавання елемента в множину	3, 1
	Клас-контейнер МНОЖИНА з елементами типу int.	

6	Реалізувати операції: [] - доступу за індексом; int() - визначення розміру вектора; * - перетин множин; --- видалення елемента з множини.	1, 2
7	Клас-контейнер МНОЖИНА з елементами типу int. Реалізувати операції: [] - доступу за індексом; == - перевірка на рівність; >число - приналежність числа множини; - n - перехід вліво до елементу з номером n.	2, 3
8	Клас-контейнер МНОЖИНА з елементами типу int. Реалізувати операції: [] - доступу за індексом; ! = - перевірка на нерівність; <число - приналежність числа множині; +n- перехід вправо до елементу з номером n.	3, 1
9	Клас-контейнер МНОЖИНА з елементами типу int. Реалізувати операції: [] - доступу за індексом; () - визначення розміру вектора; - - різниця множин; --- видалення елемента з множини.	1, 2

5.4 Вимоги до звіту

- Назва роботи.
- Мета роботи.
- Тексти програм з коментарями.
- Результати виконання програм у вигляді копій консолі.
- Висновки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Берн Страуструп. Язык программирования С++. Второе дополненное издание. – М: Бином-Пресс, 2008. – 369 с.
2. Страуструп Б. Язык программирования С++. Пер. с англ. М.: Радио и связь. 1991, 1993 (второе издание).
3. Прата Стивен. Язык программирования С++. Лекции и упражнения. Учебник: Пер. с англ./Стивен Прата – СПб.:ООО «ДиаСофтЮП», 2003. –1104 с.
4. Шилдт Герберт. Полный справ очник по С++. Пер. с англ. – М: Вильямс, 2004. 783 с.
5. Шпак З.Я. Програмування мовою С. – Львів: Оріяна-Нова, 2012. – 43 с.