

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ
ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ім. О. ГОНЧАРА

ІНСТРУКТИВНО-МЕТОДИЧНІ МАТЕРІАЛИ

для виконання лабораторних робіт

з дисципліни

«Об'єктно-орієнтоване програмування»

для вищих навчальних закладів 1 та 2 рівня
акредитації

зі спеціальності 121

«ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Розробила викладач _____ Н.В.Гапоненко

Розглянуто та затверджено комісією
програмної інженерії

Протокол № _____ від _____ . 20 ____ р.

Голова ЦК ІІ: _____ С.С. Ланська

Дніпро,
2023

ЛАБОРАТОРНА РОБОТА №1-2

Знайомство з середовищем програмування. Створення простого проекту

Мета: ознайомитись з середовищем програмування, створити проект консольного додатку

Теоретична частина

Integrated Development Environment (інтегроване середовище розробки), або, скорочено, IDE — це програмний продукт, який об'єднує текстовий редактор, компілятор, відладчик і довідкову систему.

Запуск IDE здійснюється, наприклад, через меню Пуск>Програми>Visual Studio 2019, або клацанням мишею по відповідній піктограмі на робочому столі.

Після запуску з'являється головне вікно програми, в якому розміщені інші віконця.

Вікно Solution Explorer призначено для надання допомоги при написанні і супроводі великих багатофайлових програм. Спочатку воно закрите, але після створення нового проекту (або завантаження раніше створеного проекту) одна з вкладок цього вікна буде містити список файлів проекту.

Вікно Editor (вікно редактора) використовується для введення та перевірки вихідного коду.

Вікно Output (вікно виводу) служить для виведення повідомлень про хід компіляції, складання та виконання програми. Зокрема, повідомлення про помилки, які виникають, з'являються саме в цьому вікні.

Під заголовком головного вікна, як і у всіх Windows-додатках, знаходиться рядок меню. Для кнопок панелей інструментів передбачена зручна контекстна допомога: при наведенні курсора миші на кнопку й затриманні на декілька секунд спливе підказка з призначенням кнопки.

Середовище розробки дозволяє будувати проекти різних типів, орієнтовані на різні сфери застосування. При запуску консольного додатку операційна система створює так зване консольне вікно, через яке йде весь ввід-вивід програми. Цей тип додатків найбільше підходить для мети вивчення мови C/C++,

тому що компільовані програми не «покриваються» товстим шаром проміжного Windows-коду.

Частина I. Створення консольного додатку

Завдання. Створити консольний додаток з назвою Lab01. Ознайомитися зі складом робочого вікна, яке може мати орієнтовний вигляд, подібний до наведеного на рисунку 1. Описати різницю між компіляцією, побудовою і виконанням проекту, ознайомитись із окремими пунктами головного меню, наведеними на рисунку 2. Розібратись, де знаходиться Solution Explorer (Оглядач рішень) і знайти головну функцію. Описати процедуру додавання нового файлу з програмним кодом та додавання вже існуючого файлу.

Відкрити папку проекту на диску, навести, яке розширення має файл проекту, які ще є основні файли та папки, що відносяться до вихідних файлів (так звані "исходники").

Оформити звіт, в ході роботи навести індивідуальні кроки з виконання роботи і власні фрагменти вікон.

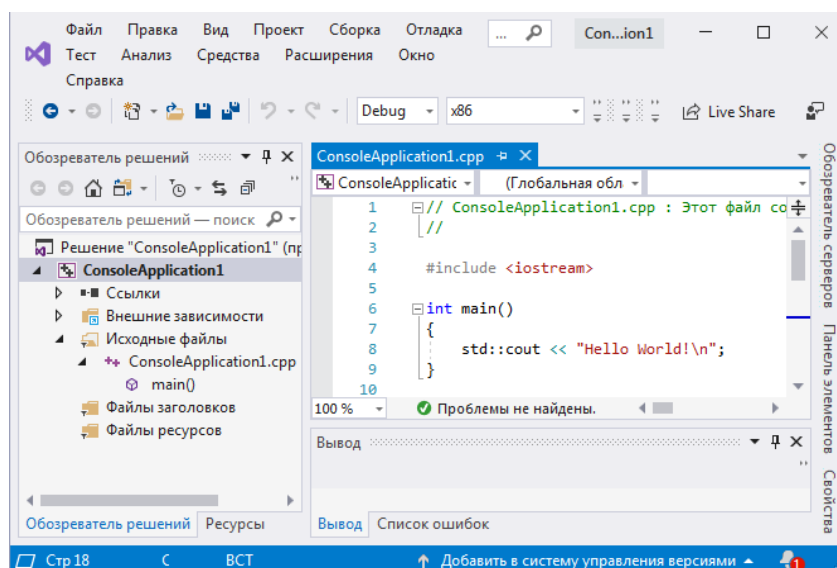


Рис. 1 – Вигляд вікна

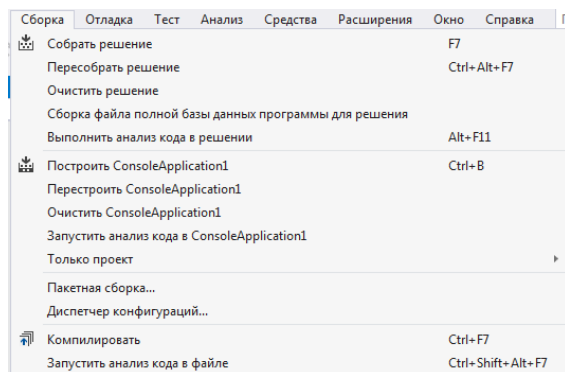


Рис. 2 – Фрагмент головного меню

Частина II

Завдання. В проєкті внесіть зміни в тіло головної функції, наведеної в лістингу 1, при необхідності допишіть підключення відповідних бібліотек, прокоментуйте рядки програмного коду, допишіть виведення власного прізвища на екран. Додайте в проєкт функції (відокремлюючи оголошення від визначення), які виконують дії, наведені в лістингу 2, індивідуально прокоментуйте рядки програмного коду. Додайте виклик цих функцій з головної функції. В четвертому фрагменті допишіть знаходження суми елементів масиву. В п'ятому фрагменті замість знаків питань поставте знаки необхідних операцій доступу до полів структур і поясніть. Допишіть звільнення пам'яті.

Оформіть звіт. Наведіть хід роботи, текст програми зі створеними функціями, детальними коментарями, результати роботи програми, висновок.

Лістинг 1 – Програмний код головної функції

```
int main ( )
{
    int x, sum = 0 ;
    bool flag = true ;
    while ( flag )
    {
        cout << "Enter x -> " ;
        cin >> x ;
        if ( x > 0 )
        {
            sum += x ;
            continue ;
        }
        flag = false ;
    }
    cout << sum << endl ; //вивести прізвище
    return 0 ;
}
```

Лістинг 1 – Фрагменти програмного коду для формування функцій

1)

```
int x = 5 ;
int * px = & x ;
cout << "Value x\t\t" << x << endl ;
cout << "Address x\t" << & x << endl ;
cout << "Pointer px\t" << px << endl ;
```

2)

```
double v1 = 0.05, v2 = 2.5e32 ;
double * pv ;
pv = & v1 ;
cout << "pv =\t" << pv << endl ;
cout << "v1 =\t" << v1 << endl ;
cout << "*pv = \t" ;
cout << * pv << endl ;
pv = & v2 ;
cout << "pv =\t" << pv << endl ;
cout << "v2 =\t" << v2 << endl ;
cout << "*pv = \t" ;
cout << * pv << endl ;
```

3)

```
int* pi ; unsigned short* pu ;
pi = new int ;
pu = new unsigned short ( 200 ) ;
double* pd = new double ;
float* pf = new float ( -3.15 ) ;
cout << "Pointers" ;
cout << "\npi\t" << pi << "\npu\t" << pu ;
cout << "\npd\t" << pd << "\npf\t" << pf ;
cout << "\n\nValues with initialization" ;
cout << "\n* pu\t" << * pu ;
cout << "\n* pf\t" << * pf ;
cout << "\n\nValues without initialization" ;
cout << "\n* pi\t" << * pi ;
cout << "\n* pd\t" << * pd << endl ;
* pi = - * pu ; * pd = - * pf ;
cout << "\n\nValues after initialization" ;
cout << "\n* pi\t" << * pi ;
cout << "\n* pd\t" << * pd << endl ;
```

4)

```
int array [ ] = { 10, 20, 30, 40, 50 } ;
int size = sizeof ( array ) / sizeof ( array [ 0 ] ) ;
for ( int i = 0; i < size; i++ )
    cout << * ( array + i ) << '\t' ;
cout << endl ;
int* p = array ;
int count = 0 ;
do
{
    cout << * p++ << '\t' ;
    count++ ;
} while ( count < size ) ;
cout << endl ;
```

5)

```
// прості структури
struct Address {
    char city [ 20 ] ; char street [ 30 ] ; int house ;
```

```

} ;
struct Person {
    char Fname [ 15 ] ; char Lname [ 20 ] ;
} ;
// ієрархічна структура
struct Employee {
    Person p ;          // вкладена структура
    Address addr ;      // вкладена структура
} ;
Employee *p = new Employee ;
cin >> p ? p ? Fname ; cin >> p ? p ? Lname ;
cin >> p ? addr ? city ;
cin >> p ? addr ? street ;
cin >> p ? addr ? house ;

```

ЛАБОРАТОРНА РОБОТА № 3-4

Проектування та створення класу. Відлагодження та тестування програми, що містить клас

Мета: навчитись проектувати та створювати класи, відлагоджувати та тестувати програми, що містять клас.

Частина I

Постановка задачі. Створити консольний додаток C++. Додати в проєкт клас (модулі *.h, *.cpp), який буде повідомляти про створення та знищення об'єктів цього класу та розраховувати факторіал, внести програмний код за зразком, відлагодити, прокоментувати.

Порядок виконання роботи.

Для додавання в проєкт нового класу потрібно виконати дії, подібні до наведених нижче.

З використанням головного меню середовища або в дереві проєкту або на вкладці ClassView браузера клацнути над назвою проєкту і у контекстному меню обрати пункти Add, Class, C++, C++ Class (назви можуть незначно відрізнятись), натиснути Add. Ввести назву нового класу — myClass, завершити.

Переглянути в дереві проєкту файли, що з'явилися. Звернути увагу, що для класу myClass одразу створився новий файл myClass.cpp, який містить визначення (реалізацію) класу, і новий заголовний файл myClass.h, який включає опис (специфікацію) класу. Перегляньте у редакторі файлів ці створені файли.

Виберіть заголовний файл myClass.h та внесіть в нього опис функції розрахунку факторіалу factorial, як наведено в лістингу 1.

Лістинг 1 - Фрагмент програмного коду файлу myClass.h

```
class myClass {  
public:  
    myClass();  
    ~myClass();  
    unsigned int factorial(unsigned int i);  
};
```

В файлі myClass.cpp внесіть зміни у конструктор та деструктор цього класу, які були згенеровані автоматично при створенні класу, а також додайте

визначення функції факторіалу `factorial` за зразком, наведеним в лістингу 2. (Конструктор та деструктор - функції, що належать класу і мають ім'я, що співпадає з іменем класу, деструктор має тильду ~).

Лістинг 2 - Фрагмент програмного коду файлу `myClass.cpp` (реалізація класу)

```
#include "myClass.h"

myClass::myClass() {
    cout << "myClass created!\n" << endl;
}

myClass::~~myClass() {
    cout << "myClass destroyed!\n" << endl;
}

unsigned int myClass :: factorial(unsigned int i)
{
    int res = i;
    while (i>1)
        res *= --i;
    return res;
}
```

В файл `*.cpp`, де визначена головна функція `main`, внесіть зміни, необхідні для виклику функції факторіалу `factorial` як методу двох об'єктів: `N` та об'єкту, який задається покажчиком `m` за зразком, наведеним в лістингу 3.

Лістинг 3 - Фрагмент програмного коду головної функції

```
#include "myClass.h"

int main(int argc, char* argv[])
{
    int d;
    MyClass *m = new MyClass( );
    MyClass N;
    cout << "\nRashet factorial dlja klassa m \n" << endl;
    cout << "\nVvedite d\n" << endl;
    cin >> d;
    cout << "\nFactorial m->factorial(d) = " << m->factorial(d) << endl;

    cout << "\nRashet factorial dlja klassa N\n" << endl;
    cout << "\nVvedite d\n" << endl;
    cin >> d;
    cout << "\nFactorial N.factorial(d) = " << N.factorial(d) << endl;

    return 0;
}
```

Відлагодьте програму, протестуйте для різних значень вхідних даних.

Опишіть у звіті хід та результати роботи, прокоментуйте програмний код.

Частина II

Спроектувати програму, у якій описується клас CR## (## - двозначний номер варіанту за списком в журналі). Створити два об'єкти цього класу, один з яких динамічний. Клас повинен містити дві функції (методи): metod1 та metod2, параметри методів і потрібні поля в класі спроектувати самостійно. Призначення методів задається у відповідності з варіантом за таблицею 1. Продемонструвати роботу методів класу для кожного з об'єктів. Дати можливість користувачу вводити необхідні дані, наприклад, N.

Таблиця 1 - Варіанти завдань

№ варіанту	Призначення методу	
1	metod1	Розрахунок добутку суми парних чисел від 1 до N на суму непарних
	metod2	Розрахунок значення функції $4x \cdot (x-2)$
2	metod1	Розрахунок результату ділення суми парних чисел від 1 до N на суму непарних
	metod2	Розрахунок значення функції $8(x-x^2)$
3	metod1	Розрахунок добутку суми непарних чисел від 1 до N на добуток чисел, кратних 3
	metod2	Розрахунок значення функції $x \cdot (x-2)^3$
4	metod1	Розрахунок добутку суми чисел від 1 до N на суму непарних
	metod2	Розрахунок значення функції $(1-x) \cdot (x-2)$
5	metod1	Розрахунок добутку чисел, кратних 3, від 1 до N на суму парних
	metod2	Розрахунок значення функції $2\sin(x) \cdot (x-2)$
6	metod1	Розрахунок суми чисел, кратних 7, від 1 до N, помноженої на добуток непарних
	metod2	Розрахунок значення функції $1 + \cos(x) \cdot (x-2)$
7	metod1	Розрахунок добутку сум парних та непарних чисел від 1 до N
	metod2	Розрахунок значення функції $\sin(x-1) \cdot (x-2)$
8	metod1	Розрахунок добутку парних чисел від 1 до N на добуток непарних
	metod2	Розрахунок значення функції $x^2 \cdot (x-2)$
9	metod1	Розрахунок добутку суми парних чисел від 1 до N на добуток кратних 3
	metod2	Розрахунок значення функції $\sin^2(x) \cdot x$
10	metod1	Розрахунок добутку суми парних чисел від 1 до N на суму кратних 5
	metod2	Розрахунок значення функції $2 - \cos^2(x)$
11	metod1	Розрахунок добутку суми парних чисел від 1 до N на добуток

		кратних 3
	metod2	Розрахунок значення функції $-\cos(x)*(x^2-2)$
12	metod1	Розрахунок добутку суми парних чисел від 1 до N, помноженого на суму кратних 7
	metod2	Розрахунок значення функції $(1+\cos(x))*(x-2)$
13	metod1	Розрахунок добутку суми парних чисел від 1 до N на добуток кратних 3
	metod2	Розрахунок значення функції $\sin^2(x)*x$
14	metod1	Розрахунок добутку суми парних чисел від 1 до N на суму кратних 5
	metod2	Розрахунок значення функції $2-\cos^2(x)$
15	metod1	Розрахунок добутку суми парних чисел від 1 до N на добуток кратних 3
	metod2	Розрахунок значення функції $-\cos(x)*(x^2-2)$
16	metod1	Розрахунок добутку суми парних чисел від 1 до N на суму непарних
	metod2	Розрахунок значення функції $5x*(x-2)$
17	metod1	Розрахунок результату ділення суми парних чисел від 1 до N на суму парних
	metod2	Розрахунок значення функції $6(x-x^2)$
18	metod1	Розрахунок добутку суми непарних чисел від 1 до N на добуток чисел, кратних 4
	metod2	Розрахунок значення функції $x*(x-3)^3$
19	metod1	Розрахунок добутку суми всіх чисел від 1 до N на суму непарних
	metod2	Розрахунок значення функції $(2-x)*(x-2)$
20	metod1	Розрахунок добутку чисел, кратних 3, від 1 до N на суму парних
	metod2	Розрахунок значення функції $2\sin(x)*(x-3)$
21	metod1	Розрахунок суми чисел, кратних 6, від 1 до N, помноженої на добуток непарних
	metod2	Розрахунок значення функції $1+\cos(x)*(x-3)$
22	metod1	Розрахунок добутку сум парних та непарних чисел від 1 до N
	metod2	Розрахунок значення функції $\sin(x-2)*(x-3)$

Протестуйте програму для різних значень вхідних даних, передбачте необхідні перевірки на припустимість.

Прокоментуйте програму. Опишіть хід та результати роботи, оформіть звіт.

ЛАБОРАТОРНА РОБОТА № 5-6

Створення та відлагодження програми з ініціалізацією полів в конструкторі класу

Мета: навчитись створювати та відлагоджувати програми з ініціалізацією полів в конструкторі класу

Частина I

Постановка задачі. Виконати модифікацію програми лабораторної роботи №3 наступним чином. Передбачити закрите поле `n` замість використання параметра функції розрахунку факторіала (параметр виключити). Ініціалізувати поле `n` в конструкторі за замовчанням. Додати відкритий метод (функцію класу) для введення значення `n`, додати перевірки на допустимість). Пояснити доцільність проведених змін в програмі. Вивести власне прізвище в головній функції.

Порядок виконання роботи.

Передбачити закрите поле `n` в оголошенні класу, виключити параметр в функції розрахунку факторіала, додати відкритий метод (функцію класу) для введення значення `n`, як наведено в лістингу 1.

Лістинг 1 - Фрагмент програмного коду файлу `myClass.h`

```
class myClass {
    int n;
public:
    myClass();
    ~myClass();
    int factorial();
    void set_n();
};
```

В файлі `myClass.cpp` (реалізація класу) додати ініціалізацію поля в конструктор і за необхідністю інші зміни.

Лістинг 2 - Фрагмент програмного коду файлу `myClass.cpp` (реалізація класу)

```
#include "myClass.h"

myClass::myClass() {
    cout << "myClass created!\n" << endl;
```

```

    n=1;
}

myClass::~myClass() {
cout << "myClass destroyed!\n" << endl;
}

int myClass :: factorial()
{
    int res = n;
    while (n>1)
        res *= --n;
    return res;
}

void myClass :: set_n()
{
    // ... дописати можливість введення n користувачем і перевірки
    return;
}

```

В файл *.cpp, де визначена головна функція main, внесіть зміни, пов'язані з викликом функції встановлення значення для поля за зразком, наведеним в лістингу 3.

Лістинг 3 - Фрагмент програмного коду головної функції

```

#include "myClass.h"

int main(int argc, char* argv[])
{
    MyClass *m = new MyClass( );
    MyClass N;
    cout << "\nRashet factorial dlja klassa m \n" << endl;
    cout << "\nFactorial m->factorial() = " << m->factorial() << endl;

    cout << "\nRashet factorial dlja klassa N\n" << endl;
    N.set_n();
    cout << "\nFactorial N.factorial(d) = " << N.factorial() << endl;

    return 0;
}

```

Відлагодьте програму, протестуйте для різних значень вхідних даних.

Опишіть у звіті хід та результати роботи, прокоментуйте програмний код, поясніть, як працює програма, якщо для одного з об'єктів не викликали метод set_n().

Постановка задачі. Виконати модифікацію програми лабораторної роботи №4 наступним чином. Передбачити закриті поля замість використання параметрів методів `metod1`, `metod2` (параметри виключити). Ініціалізувати поля в конструкторі за замовчанням. Передбачити конструктор з параметрами для ініціалізації полів значеннями параметрів (допускається як конструктор з двома параметрами, так і два конструктори з параметрами по одному параметру різних типів). Додати відкриті методи для встановлення значень полів, додати перевірки на допустимість.

Для даної лабораторної роботи допускається виведення значення поля в консоль під час ініціалізації для наглядності проведених розрахунків, в подальшому, як правило, для цього будуть передбачатися додаткові методи.

Пояснити модифікацію програми в звіті.

ЛАБОРАТОРНА РОБОТА № 7-8

Проектування функцій-членів класу. Відлагодження та тестування програми з використанням функцій-членів класу

Мета: спроектувати, відлагодити та протестувати розроблені функції-члени класу, використати розроблені методи для роботи з об'єктами класу

Частина I

Постановка завдання. Створити клас, який описує книгу, містить конструктори та деструктор для створення та знищення об'єктів цього класу, а також методи для роботи з закритими полями за наведеним зразком. Оголосити у програмі другий динамічний об'єкт класу `SBook`, встановити значення його полів, вивести значення його даних з використанням функції `view`. Відлагодити програму, оптимізувати ініціалізацію динамічного поля класу, дописати за потреби звільнення пам'яті, оформити звіт. Пояснити:

- як і які об'єкти створюються у програмі;
- як виводяться значення полів об'єктів;
- які кроки передбачені для відлагодження програми;
- як універсально використовується функція `view` для двох об'єктів.

Порядок виконання роботи описаний нижче.

Створити консольний додаток, додати до проєкту клас `SBook`, який описує книгу.

Передбачити закриті поля з ініціалізацією в конструкторі за замовчанням та в конструкторі з параметрами та методи для роботи з полями (введення/виведення значень). Скористатись запропонованим програмним кодом, дописати код, якого не вистачає.

Файл специфікації CBook.h містить оголошення типу класу CBook. Щоб компілятор включав оголошення типу тільки один раз при трансляції програми, використається директива препроцесора `#pragma once`.

У класі оголошені закриті дані, які описують характеристики книги: автора, назву й рік видання. Щоб можна було змінювати значення даних об'єктів класу й виводити їх, у класі оголошені відкриті методи. Функції-члени, призначені для встановлення значень полів класу, мають параметр, тип якого відповідає типу даного, та тип значення `void`, що повертається. Методи для повернення значень не мають потреби в параметрах. Тип кожного методу обумовлений типом змінної, що повертається.

Внести до файлу Book.h – файлу специфікації класу CBook – програмний код, наведений в лістингу 7.1.

Лістинг 3.1 – Файл CBook.h — специфікація класу CBook.

```
class CBook {
private:
    char m_author [ 50 ] ;    // автор
    char *m_pTitle ;          // вказівник на назву
    int m_year ;              // рік видання
public:
    CBook ( ) ;
    ~ CBook ( ) ;
    // методи встановлення значень полів
    void setAuthor ( const char* ) ;
    void setTitle ( const char* ) ;
    void setYear ( const int ) ;
    // методи повернення значень полів
    char* getAuthor ( ) ;
    char* getTitle ( ) ;
    int getYear ( ) ;
}
```

Реалізація класу в файлі CBook.cpp. Внести програмний код, призначений для визначення методів за зразком, наведеним у лістингу 7.2.

Лістинг 7.2 – Файл CBook.cpp — реалізація класу CBook.

```
// встановити автора
```

```

void CBook :: setAuthor ( const char* author ) {
    cout << "\n працює setAuthor  \n"<< author<< endl;
    m_author [ 0 ] = '\0' ;
    strcat(m_author, author);
    cout << m_author<<endl;
    cout << "\n відпрацював setAuthor\n"<<endl;
}

// встановити назву
void CBook :: setTitle ( const char* title ) {
    cout << "\n працює setTitle\n"<<endl;
    cout << "\nstrlen ( title ) = "<<strlen ( title )<<endl;
    cout << "\ntitle  = "<< title <<endl;
    m_pTitle = new char [strlen ( title )+ 1] ;
    m_pTitle [ 0 ] = '\0' ;
    strcat(m_pTitle, title);
    cout << m_pTitle<<endl;
    cout << "\n відпрацював setTitle\n"<<endl;
}

// встановити рік видання
void CBook :: setYear ( const int year ) {
    m_year = year ;
}

// повернути автора
char* CBook :: getAuthor ( ) { return m_author ; }

// повернути назву
char* CBook :: getTitle ( ) { return m_pTitle ; }

// повернути рік видання
int CBook :: getYear ( void ) { return m_year ; }


// конструктор за замовчанням
CBook :: CBook ( ) : m_year ( 0 ), m_pTitle ( " " )
{ m_author [ 0 ] = '\0' ;
cout << "default CONSTRUCTOR\nthis = " << this ;
}

// деструктор
CBook::~CBook() {
    delete [ ] m_pTitle ;
}

```

```

    cout << "DESTRUCTOR\t " << this << endl ;
}

```

Головна функція має містити програмний код за зразком, наведеним в лістингу 7.3.

Лістинг 7.3 – Фрагмент програмного коду файлу з головною функцією.

```

void view ( char*, CBook& ) ;
int main(int argc, char* argv[]) {
    CBook book ;
    char stroka_vvoda[10];
    cout << "Vvedi avtora" << endl;
    cin  >> stroka_vvoda;
    book.setAuthor ( stroka_vvoda ) ;
    book.setAuthor ( "Robert Lafore" ) ;
    cout << "\n otrabotal setAuthor pole = " << book.getAuthor() <<
endl;
    book.setTitle ( "Object-Oriented Programming in C++" ) ;
    cout << "\n otrabotal setTitle pole = " << book.getTitle() << endl;
    book.setYear ( 2004 )
    view ( "book", book ) ;
    cout << "\n otrabotal view " <<  endl;
return 0;
}
void view ( char *s, CBook &o ) {
    cout << "\nState of object \' " << s << " \'\n" ;
    cout << "Author:\t" << o.getAuthor ( ) << endl ;
    cout << "Title:\t" << o.getTitle ( ) << endl ;
    cout << "Year:\t" << o.getYear ( ) << endl << endl ;
}

```

Частина II

Завдання. Спроектувати клас за варіантом, наведеним в таблиці 8.1 (навести завдання в звіті), його конструктори за замовчуванням та з параметрами і деструктор, методи встановлення та отримання значень полів, продемонструвати виклик методів в головній функції, ініціалізувати поля в конструкторах. У

програмі створити два об'єкти, один з яких динамічний. Відлагодити програму, протестувати для різних значень вхідних даних, оформити звіт.

Таблиця 8.1 – Завдання за варіантами

Варіант, найменування класу	Характеристики
1. Клас «Аудиторія»	Номер
	Предмет, який викладається
	Площа
2. Клас «Білет (на поїзд)»	До якого пункту призначення
	Вартість
	Купе/плацкарта
3. Клас «Ліки»	Назва
	Вартість
	Термін придатності
4. Клас «Путівка до санаторію»	Назва санаторію
	Тривалість
	Вартість
5. Клас «Хлібопродукт»	Найменування
	Вага
	Ціна
6. Клас «Студент»	Прізвище
	Група
	Рік народження
7. Клас «Квитанція сплати за електроенергію»	Прізвище
	Останні показники електрорахівника
	Сума до сплати
8. Клас «Квиток до театру»	Назва вистави
	Ряд
	Місце
9. Клас «Член футбольної команди»	Прізвище
	Кількість проведених матчів
	Номер
10. Клас «Рейс аерофлоту»	Номер
	Місце призначення
	Вартість білету
11. Клас «Автобус»	Номер
	Марка
	Кількість місць
12. Клас «Фрукти»	Найменування
	Вага
	Ціна
13. Клас «Мешканець гуртожитку»	Прізвище
	Група
	Номер блока
14. Клас «Марка автомобіля»	Назва
	Потужність двигуна
	Ціна
15. Клас «Кабінет коледжу»	Назва
	Кількість місць
	Номер
16. Клас «Аудиторія»	Номер

	Предмет, який викладається
	Площа
17. Клас «Білет (на поїзд)»	До якого пункту призначення
	Вартість
	Купе/плацкарта
18. Клас «Ліки»	Назва
	Вартість
	Термін придатності
19. Клас «Путівка до санаторію»	Назва санаторію
	Тривалість
	Вартість
20. Клас «Хлібопродукт»	Найменування
	Вага
	Ціна
21. Клас «Студент»	Прізвище
	Група
	Рік народження
22. Клас «Квитанція сплати за електроенергію»	Прізвище
	Останні показники електрорахівника
	Сума до сплати
23. Клас «Квиток до театру»	Назва вистави
	Ряд
	Місце
24. Клас «Член футбольної команди»	Прізвище
	Кількість проведених матчів
	Номер
25. Клас «Рейс аерофлоту»	Номер
	Місце призначення
	Вартість білету
26. Клас «Автобус»	Номер
	Марка
	Кількість місць
27. Клас «Фрукти»	Найменування
	Вага
	Ціна
28. Клас «Мешканець гуртожитку»	Прізвище
	Група
	Номер блока
29. Клас «Марка автомобіля»	Назва
	Потужність двигуна
	Ціна
30. Клас «Кабінет коледжу»	Назва

ЛАБОРАТОРНА РОБОТА № 9-10

Створення та відлагодження програми з використанням конструктора копіювання

Мета: спроектувати, відлагодити програму з використанням конструктора копіювання

Частина I

Постановка завдання. Клас CBook з лабораторної роботи 7, який описує книгу та містить поля "Назва книги", "Рік видання", динамічне поле "Автор", конструктори та деструктор для створення та знищення об'єктів цього класу, методи для роботи з закритими полями, доповнити оголошенням в класі конструктору копіювання. Передбачити в конструкторі копіювання ініціалізацію полів та виведення власного прізвища, створити в головній функції екземпляр класу з використанням конструктора копіювання, продемонструвати виклик функції view зі створеним об'єктом. Відлагодити програму, оптимізувати ініціалізацію динамічного поля класу, оформити звіт. Пояснити:

- як має бути оголошений конструктор копіювання;
- як створити об'єкт класу, щоб спрацював конструктор копіювання;
- коли потрібно писати реалізацію конструктора копіювання.

Частина II

Завдання. Клас з лабораторної роботи 8, спроектований за варіантом, доповнити оголошенням в класі одного з полів динамічним (за предметною областю) та оголошенням конструктору копіювання. Передбачити в конструкторі копіювання ініціалізацію полів та виведення this, створити в головній функції екземпляр класу з використанням конструктора копіювання, продемонструвати виклик функції виведення стану об'єкту view. Відлагодити програму, оптимізувати ініціалізацію динамічного поля класу, оформити звіт.

ЛАБОРАТОРНА РОБОТА № 11-12

Складання, відлагодження та тестування програми з перевантаженням імен функцій

Мета: навчитись складати, відлагоджувати та тестувати програми з перевантаженням імен функцій

Теоретична частина

Перевантаження функцій являють собою використання того самого імені для декількох функцій. При виклику перевантаженої функції компілятор визначає відповідну функцію шляхом аналізу кількості, типів і порядку проходження параметрів у виклику. Перевантаження функцій повинне розрізнятися або типами параметрів, або їхньою кількістю. Якщо при однаковому списку параметрів функції будуть розрізнятися тільки типом значення, що повертається, компілятор повідомить про помилку неоднозначності. Параметри за замовчуванням так само можуть стати джерелом помилок з тієї причини, що виклик однієї функції може виявитися аналогічним виклику іншої перевантаженої функції.

Частина 1

Постановка задачі. Розробити перевантажені функції, які здійснюють реверс, тобто розміщення у зворотному порядку елементів масиву, причому функції будуть мати одне й те ж ім'я `reverseArray` і будуть виконувати реверс як для всього масиву, так і для його частини, а масив може бути як масивом цілих чисел, так масивом символів, тобто, рядком. Для перестановки елементів масиву буде використатися перевантажена функція `swapR`, причому вона буде використатися як для перестановки чисел, так і для перестановки символів. Скористатися зразком програмного коду, наведеним у лістигнах 11.1. Відлагодити, прокоментувати програму, оформити звіт, пояснити принцип перевантаження.

Лістинг 11.1

```

// головна функція

void swapR ( int&, int& ) ;
void swapR ( char&, char& ) ;
void reverseArray ( int*, int = 10 ) ;
int* reverseArray ( const int*, int, int ) ;
char* reverseArray ( const char* ) ;
void view ( const int*, int = 5 ) ;
int main ( )
{
    int ar [ 10 ] = { 1, 2, 3, 4, 5, 6 } ;
    cout << "Initial array\n" ;    view ( ar, 10 ) ;
    // реверс усього масиву
    reverseArray ( ar ) ;
    cout << "\nReverse array\n" ; view ( ar, 10 ) ;
    // реверс частини масиву
    int* p = reverse Array ( ar, 5, 9 ) ;
    cout << "\nReverse part array\n" ; view ( p ) ;
    // реверс стрічки
    char* s = "NOMEL ON NOLEM ON" ;
    char* pr = reverseArray ( s ) ;
    cout << "\nInitial string\n" ;    cout << s << endl ;
    cout << "\nReverse string\n" ;    cout << pr << endl ;
    return 0 ;
}

// перестановка за допомогою посилань (ссылки)
void swapR ( int& rx, int& ry )
{    int t = rx ; rx = ry ; ry = t ;    }
void swapR ( char& rx, char& ry )
{    char t = rx ; rx = ry ; ry = t ;    }
// реверс масиву в самому собі
void reverseArray ( int* p, int size ) {
    for ( int i = 0, j = size - 1; i < j; i++, j-- )
        swapR ( p [ i ], p [ j ] ) ;
}

// формування зворотнього масиву із частини масиву
int* reverseArray ( const int* pIn, int top, int end ) {
    if ( end <= top ) return 0 ;
    int size = end - top + 1 ; int* pRes = new int [ size ] ;
    int i, j, ir, jr ;
    for ( i = top, j = end, ir = 0, jr = size - 1; i < j; )
    {
        pRes [ ir++ ] = pIn [ j-- ]
        pRes [ jr-- ] = pIn [ i++ ] ;
    }
    if ( i == j ) pRes [ ir ] = pIn [ i ] ;
    return pRes ;
}

// реверс стрічки
char* reverseArray ( const char* p ) {

```

```

char* pOut = new char [ strlen ( p ) + 1 ] ;
strncpy ( pOut, p, strlen ( p ) + 1 ) ;
char *pTop = pOut, *pEnd = pOut + strlen ( pOut ) - 1 ;
while ( pTop < pEnd )
    swapR ( *pTop++, *pEnd-- ) ;
return pOut ;
}

void view ( const int* arr, int count )
{
    for ( int i = 0; i < count; i++ ) {
        cout << '\n' << "\ti=" << i << "    " << arr[i] << endl ;
    }
}

```

Порядок виконання роботи

Створити консольний додаток із запропонованим програмним кодом.

У лістингу є дві перевантажені функції: `swapR ( )` і `reverseArray( )`. Функція `swapR()` перестановки значень перевантажена для параметрів посилань на цілий і символьний тип. Таким чином, функції розрізняються типом параметрів зі списку параметрів. Версія з посиланнями на цілий тип `swapR ( p [ i ], p [ j ] )` викликається усередині функції реверса масиву цілих чисел, а з посиланнями на символьний тип `swapR ( *pTop++, *pEnd-- )` - у функції реверса рядка. Другий виклик вимагає розіменування, тому що `pTop` і `pEnd` є покажчиками.

Перевантажена функція `reverseArray ( )` має три варіанти реалізації, які розрізняються типом значення, що повертається, типом параметрів; а також їхньою кількістю.

Прототип `void reverseArray ( int*, int = 10 )`; визначає функцію, яка використовується для реверса масиву цілих чисел у самому собі. У головній функції `main ( )` вона викликається як `reverseArray ( ar )`, використовуючи значення за замовчуванням 10 для кількості елементів.

Прототип `int* reverseArray ( const int*, int, int )`; відрізняється від попереднього типом значення, що повертається, і не використовує значень за замовчуванням. Функція призначена для створення нового зворотного масиву із частини вихідного. Адреса вихідного масиву задається першим параметром. Два останніх параметри визначають початковий і кінцевий індекси елементів у вихідному масиві, які повинні бути переписані в новий масив у зворотному

порядку. Функція повертає покажчик на новий масив. Перший параметр є покажчиком на константу, що забороняє зміну елементів у вихідному масиві. У попередньому варіанті таке рішення неможливо тому, що реверс усередині масиву вимагає перестановки існуючих значень. Два останніх параметри не можна оголосити параметрами за замовчуванням, тому що виникне невизначеність, і компілятор не зможе розрізнити виклики цих функцій.

Обробка масиву починається з перевірки заданих у виклику індексів. Повертається 0, якщо кінцевий індекс менше або дорівнює початковому. Інакше обчислюється розмір необхідної пам'яті, динамічно виділяється блок пам'яті відповідно до обчисленого значення й запускається цикл `for`, у якому формується новий масив. Велика кількість змінних у циклі обумовлена тим, що необхідно два індекси `i` й `j` для переміщення по вихідному масиві й ще два індекси `ig`, `jr` для переміщення по створюваному масиві. Індекс `i` змінюється від значення, заданого параметром `top`, убік збільшення на 1 доти, поки він буде менше індексу `j`. Індекс `j` змінюється від значення, заданого параметром `end`, убік зменшення на 1. Індекс `ig` змінюється від 0, тому що індексація в масиві починається саме із цього значення, і збільшується на 1 для нової ітерації циклу. Індекс `jr` зменшується на 1 наприкінці кожної ітерації, починаючи від значення індексу останнього елемента нового масиву `size - 1`. Коли кількість підлягаючих реверсу елементів непарна, на виході із циклу індекси `i` й `j` будуть рівні. При істинності відношення `i == j` у новий масив переписується центральний елемент заданого інтервалу. Функція повертає покажчик на сформований масив. У головній функції оператор

```
Int* p = reverseArray ( ar, 5, 9 );
```

призначений для виклику реверса частини масиву `ar`, обмеженого індексами 5 і 9. У результаті покажчик `p` одержить повернуте функцією значення, а `main ( )` - доступ через цей покажчик до елементів зворотної частини масиву `ig`. Сам масив `ar` не міняється. Варто пам'ятати, що операцію `delete` не можна використати для покажчика `p` у головній функції, тому що динамічного виділення пам'яті вона не робила.

Прототип `char* reverseArray ( const char* )`; оголошує перевантажену функцію для реверса рядка. Функція не змінює вихідний рядок, покажчик на яку є

її параметром. Довжину рядка передавати немає необхідності, тому що вона може бути обчислена. Функція повертає покажчик на динамічно виділений блок пам'яті, у якому розташований результат. На початку функції виділяється необхідний простір оперативної пам'яті, куди копіюється вихідний рядок. Реверс виконується з рядком, розташованому по виділеній адресі pOut, для чого в циклі while багаторазово викликається перевантажена функція перестановки елементів swapR(). У головній функції оголошується покажчик pr, якому привласнюється значення, повернуте викликом функції reverseArray (s). Стан вихідного рядка s і результуючої pr виводиться після виклику функції, причому рядок s не змінює свого значення після виклику.

Відлагодити та протестувати програму для різних значень вхідних даних.

Оформити звіт.

Частина 2

Створити клас, у відповідності з варіантом розробити перевантажені методи власного класу, призначення яких приведено в таблиці. Розрахунок виконувати **по сторонах**, формули взяти з відкритих джерел, передбачити необхідні перевірки.

№ вар	Призначення функцій
1	Розрахунок площ: А) трикутника по трьом сторонам; Б) прямокутника; В) кола.
2	Розрахунок об'ємів: А) куба; Б) конуса; В) призми з правильним трикутником в основі.
3	Розрахунок площ поверхонь тіл: А) кулі; Б) призми з прямокутником в основі; В) конусу.
4	Розрахунок площ: А) трикутника по основі і висоті; Б) трикутника по трьом сторонам; В) кола.
5	Розрахунок площ: А) трикутника по трьом сторонам; Б) трапеції;

	В) ромбу.
6	Розрахунок об'ємів: А) призми з прямокутником в основі; Б) циліндра; В) кулі.
7	Розрахунок площ бокових поверхонь тіл: А) піраміди з правильним трикутником в основі; Б) призми з прямокутником в основі; В) конусу.
8	Розрахунок площ: А) кільця; Б) трапеції; В) квадрату.
9	Розрахунок об'ємів: А) куба; Б) конуса; В) призми з правильним трикутником в основі.
10	Розрахунок площ бокових поверхонь тіл: А) циліндру; Б) піраміди з прямокутником в основі; В) призми з правильним трикутником в основі.
11	Розрахунок площ: А) трикутника; Б) прямокутника; В) кола.
12	Розрахунок площ поверхонь тіл: А) кулі; Б) призми з прямокутником в основі; В) конусу.
13	Розрахунок площ: А) кільця; Б) трапеції; В) квадрату.
14	Розрахунок площ: А) трикутника по трьом сторонам; Б) трапеції; В) ромбу.
15	Розрахунок площ бокових поверхонь тіл: А) піраміди з правильним трикутником в основі; Б) призми з прямокутником в основі; В) конусу.
16	Розрахунок площ: А) трикутника по трьом сторонам; Б) прямокутника; В) кола.
17	Розрахунок об'ємів:

	А) куба; Б) конуса; В) призми з правильним трикутником в основі.
18	Розрахунок площ поверхонь тіл: А) кулі; Б) призми з прямокутником в основі; В) конусу.
19	Розрахунок площ: А) трикутника по основі і висоті; Б) трикутника по трьом сторонам; В) кола.
20	Розрахунок площ: А) трикутника по трьом сторонам; Б) трапеції; В) ромбу.
21	Розрахунок об'ємів: А) призми з прямокутником в основі; Б) циліндра; В) кулі.
22	Розрахунок площ бокових поверхонь тіл: А) піраміди з правильним трикутником в основі; Б) призми з прямокутником в основі; В) конусу.
23	Розрахунок площ: А) кільця; Б) трапеції; В) квадрату.
24	Розрахунок об'ємів: А) куба; Б) конуса; В) призми з правильним трикутником в основі.
25	Розрахунок площ бокових поверхонь тіл: А) циліндру; Б) піраміди з прямокутником в основі; В) призми з правильним трикутником в основі.
26	Розрахунок площ: А) трикутника; Б) прямокутника; В) кола.
27	Розрахунок площ поверхонь тіл: А) кулі; Б) призми з прямокутником в основі; В) конусу.
28	Розрахунок площ: А) кільця; Б) трапеції;

	В) квадрату.
29	Розрахунок площ: А) трикутника по трьом сторонам; Б) трапеції; В) ромбу.
30	Розрахунок площ бокових поверхонь тіл: А) піраміди з правильним трикутником в основі; Б) призми з прямокутником в основі; В) конусу.

Відлагодити, прокоментувати прораму, продемонструвати її роботу, оформити звіт.

ЛАБОРАТОРНА РОБОТА № 13-14

Проектування класу з дружніми функціями. Відлагодження та тестування
програми з дружніми функціями

Мета: навчитись проектувати класи з дружніми функціями, закріпити навички з відлагодження та тестування програми з дружніми функціями

Теоретична частина

Зовнішня функція, визначена як "друг" класу, має необмежений доступ до всіх елементів класу, у тому числі й до закритих. Для того щоб зовнішня функція стала другом класу, необхідно в інтерфейсі класу задати її прототип, додавши слово `friend`. При реалізації функції слово `friend` не пишеться.

Необхідно пам'ятати, що:

- дружня функція не отримує значення вказівника `this`;
- специфікатор доступу не впливає на оголошення друзів класу;
- відносини дружби не успадковуються.

Частина 1

Постановка задачі. Оголосити клас з ім'ям `DF##` (`##` - двозначний номер студента за журналом), який містить поля і методи та дружні функції за наведеним в лістингах 13.1-13.3 зразком, продемонструвати роботу з об'єктами класу. Використати дружню функцію для виведення стану об'єкту.

Провести модифікацію програми, щоб користувач міг задавати довільний розмір масиву, і модифікацію дружніх функцій, щоб можна було задавати довільні межі для обчислення суми та добутку елементів масиву. У дружніх функціях передбачити перевірку введених даних на коректність (при виході з діапазону видати повідомлення і завершити програму, або повторити введення).

Доповнити програму необхідними ініціалізаціями, звільненням пам'яті, протестувати для різних значень вхідних даних. Оформити звіт, пояснити, необхідність використання дружніх функцій. Прокоментувати програму.

Лістинг 13.1 – Специфікація класу `DF`

```

class DF
{
    int * mas;
    int  raz_mas;
public:
    DF(int);
    DF();
    virtual ~DF();
    void SetMas();
    void PrintMas();
    friend int SumMas(DF *, int, int);
    friend int ProizMas(DF &, int, int);
};

```

Лістинг 13.2 – Реалізація класу DF

```

DF::DF()
{
}
DF::DF(int razmer): raz_mas(razmer)
{
    cout<<"raz_mas="<< raz_mas <<endl;
    mas = new int [raz_mas];
    cout<<"Constructor s param"<<endl;
}
DF::~~DF()
{
    delete [] mas;
    cout<<"Destructor "<<endl;
}

void DF::SetMas()
{
    cout<<"raz_mas="<< raz_mas <<endl;
    for(int i =0; i<raz_mas; i++)
    {
        cout<<"Vvod"<<i<<endl;
        cin>>mas[i];
    }
    cout<<"Massiv vveden"<<endl;
}
void DF::PrintMas()
{
    cout<<"\tElementy massiva"<<endl;
    for(int i =0; i<raz_mas; i++)
    {
        if (i%5 ==0) cout<<" "<<endl;
        cout<<"\t"<<mas[i];
    }
    cout<<" "<<endl;
}

```

Лістинг 13.2 – Головна функція

```

int SumMas(DF * Obj1, int n1, int n2)
{
    int sum_mas(0);
    for (int i = n1; i<=n2; i++)
    {
        sum_mas +=Obj1->mas[i];
    }
}

```

```

        cout<<"\tOtrabotala SumMas. sum_mas="<<sum_mas<<endl;
        return sum_mas;
    }
    int ProizMas(DF & Obj2, int n1, int n2)
    {
        int proiz_mas(1);
        for (int i = n1; i<=n2; i++)
        {
            proiz_mas *=Obj2.mas[i];
        }
        cout<<"\tOtrabotala ProizMas. proiz_mas="<<proiz_mas<<endl;
        return proiz_mas;
    }
    int main(int argc, char* argv[])
    {
        int S1, P1;
        DF Ob1(5);
        Ob1.SetMas();
        Ob1.PrintMas();
        S1=SumMas(&Ob1, 2, 4);
        cout<<"\tS1="<<S1<<endl;
        P1=ProizMas(Ob1, 1, 3);
        cout<<"\tP1="<<P1<<endl;
        return 0;
    }

```

Частина 2

Створити програму, яка працює з об'єктами класу, в якому містяться необхідні елементи-дані для рішення задачі у відповідності з варіантом. Клас повинен мати методи для встановлення значень елементів-даних а також дружні функції для розрахунків Run() і для виведення значень даних print();

Програма повинна включати різні способи створення об'єктів, як за іменем, так і через вказівник. Передбачити введення довільних значень користувачем, а також демонстрацію розрахунку для заданих у варіанті значень змінних, обидва варіанти продемонструвати у скріншоті виконання у звіті. Передбачити обробку помилкових ситуацій (ділення на 0 тощо). Оформити звіт, пояснити хід роботи, прокоментувати програму.

Варіанти завдань

$$1. \quad t = \frac{2 \cos\left(x - \frac{\pi}{6}\right)}{0.5 + \sin^2 y} \left(1 + \frac{z^2}{3 - z^2/5}\right).$$

При $x=14.26$, $y=-1.22$, $z=3.5 \times 10^{-2}$ **$t=0.564849$** .

$$2. \quad u = \frac{\sqrt[3]{8 + |x - y|^2 + 1}}{x^2 + y^2 + 2} - e^{|x-y|} (tg^2 z + 1)^x.$$

При $x=-4.5$, $y=0.75 \times 10^{-4}$, $z=0.845 \times 10^2$ **$u=-55.6848$.**

$$3. \quad v = \frac{1 + \sin^2(x + y)}{\left| x - \frac{2y}{1 + x^2 y^2} \right|} x^{|y|} + \cos^2 \left(\arctg \frac{1}{z} \right).$$

При $x=3.74 \times 10^{-2}$, $y=-0.825$, $z=0.16 \times 10^2$, **$v=1.0553$.**

$$4. \quad w = |\cos x - \cos y|^{(1+2\sin^2 y)} \left(1 + z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4} \right).$$

При $x=0.4 \times 10^4$, $y=-0.875$, $z=-0.475 \times 10^{-3}$ **$w=1.9873$.**

$$5. \quad \alpha = \ln \left(y^{-\sqrt{|x|}} \right) \left(x - \frac{y}{2} \right) + \sin^2 \arctg(z).$$

При $x=-15.246$, $y=4.642 \times 10^{-2}$, $z=20.001 \times 10^2$ **$\alpha=-182.036$.**

$$6. \quad \beta = \sqrt{10(\sqrt[3]{x} + x^{y+2})} (\arcsin^2 z - |x - y|)$$

При $x=16.55 \times 10^{-3}$, $y=-2.75$, $z=0.15$ **$\beta=-40.630$.**

$$7. \quad \gamma = 5 \arctg(x) - \frac{1}{4} \arccos(x) \frac{x + 3|x - y| + x^2}{|x - y|z + x^2}.$$

При $x=0.1722$, $y=6.33$, $z=3.25 \times 10^{-4}$ **$\gamma=-205.305$.**

$$8. \quad \varphi = \frac{e^{|x-y|} |x-y|^{x+y}}{\arctg(x) + \arctg(z)} + \sqrt[3]{x^6 + \ln^2 y}.$$

При $x = -2.235 \times 10^{-2}$, $y = 2.23$, $z = 15.221$ $\varphi = 39.374$.

$$9. \quad \psi = \left| x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}} \right| + (y-x) \frac{\cos y - \frac{z}{(y-x)}}{1 + (y-x)^2}.$$

При $x = 1.825 \times 10^2$, $y = 18.225$, $z = -3.298 \times 10^{-2}$ $\psi = 1.2131$.

$$10. \quad b = y^{\sqrt[3]{|x|}} + \cos^3(y) \frac{|x-y| \left(1 + \frac{\sin^2 z}{\sqrt{x+y}} \right)}{e^{|x-y|} + \frac{x}{2}}.$$

При $x = 6.251$, $y = 0.827$, $z = 25.001$ $b = 0.7121$.

$$11. \quad c = 2^{(y^x)} + (3^x)^y - \frac{y \left(\arctg z - \frac{\pi}{6} \right)}{|x| + \frac{1}{y^2 + 1}}.$$

При $x = 3.251$, $y = 0.325$, $z = 0.466 \times 10^{-4}$ $c = 4.25$.

$$12. \quad f = \frac{\sqrt[4]{y + \sqrt[3]{x-1}}}{|x-y| (\sin^2 z + \tg z)}$$

При $x = 17.421$, $y = 10.365 \times 10^{-3}$, $z = 0.828 \times 10^5$ $f = 0.33056$.

$$13. \quad g = \frac{y^{x+1}}{\sqrt[3]{|y-2|} + 3} + \frac{x + \frac{y}{2}}{2|x+y|} (x+1)^{-1/\sin z}$$

При $x = 12.3 \times 10^{-1}$, $y = 15.4$, $z = 0.252 \times 10^3$ $g = 82.8257$.

$$14. \quad h = \frac{x^{y+1} + e^{y-1}}{1 + x|y - \operatorname{tg} z|} (1 + |y - x|) + \frac{|y - x|^2}{2} - \frac{|y - x|^3}{3}.$$

При $x=2.444$, $y=0.869 \times 10^{-2}$, $z=-0.13 \times 10^3$ $h = -0.49871$.

$$15. \quad \beta = \sqrt{10(\sqrt[3]{x} + x^{y+2})} (\arcsin^2 z - |x - y|)$$

При $x=16.55 \times 10^{-3}$, $y=-2.75$, $z=0.15$ $\beta = -40.630$

$$16. \quad f = \frac{\sqrt[4]{y + \sqrt[3]{x-1}}}{|x - y|(\sin^2 z + \operatorname{tg} z)}$$

При $x=17.421$, $y=10.365 \times 10^{-3}$, $z=0.828 \times 10^5$ $f=0.33056$.

$$17. \quad t = \frac{2 \cos\left(x - \frac{\pi}{6}\right)}{0.5 + \sin^2 y} \left(1 + \frac{z^2}{3 - z^2/5}\right).$$

При $x=14.26$, $y=-1.22$, $z=3.5 \times 10^{-2}$ $t=0.564849$.

$$18. \quad u = \frac{\sqrt[3]{8 + |x - y|^2} + 1}{x^2 + y^2 + 2} - e^{|x-y|} (\operatorname{tg}^2 z + 1)^x.$$

При $x=-4.5$, $y=0.75 \times 10^{-4}$, $z=0.845 \times 10^2$ $u=-55.6848$.

$$19. \quad v = \frac{1 + \sin^2(x + y)}{\left|x - \frac{2y}{1 + x^2 y^2}\right|} x^{|y|} + \cos^2\left(\operatorname{arctg} \frac{1}{z}\right).$$

При $x=3.74 \times 10^{-2}$, $y=-0.825$, $z=0.16 \times 10^2$, $v=1.0553$.

20.
$$w = |\cos x - \cos y|^{(1+2\sin^2 y)} \left(1 + z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4} \right).$$

При $x=0.4 \times 10^4$, $y=-0.875$, $z=-0.475 \times 10^{-3}$ **$w=1.9873$.**

21.
$$\alpha = \ln \left(y^{-\sqrt{|x|}} \right) \left(x - \frac{y}{2} \right) + \sin^2 \arctg(z).$$

При $x=-15.246$, $y=4.642 \times 10^{-2}$, $z=20.001 \times 10^2$ **$\alpha=-182.036$.**

22.
$$\beta = \sqrt{10 \left(\sqrt[3]{x} + x^{y+2} \right)} \left(\arcsin^2 z - |x - y| \right)$$

При $x=16.55 \times 10^{-3}$, $y=-2.75$, $z=0.15$ **$\beta=-40.630$.**

23.
$$\gamma = 5 \arctg(x) - \frac{1}{4} \arccos(x) \frac{x + 3|x - y| + x^2}{|x - y|z + x^2}.$$

При $x=0.1722$, $y=6.33$, $z=3.25 \times 10^{-4}$ **$\gamma=-205.305$.**

24.
$$\varphi = \frac{e^{|x-y|} |x-y|^{x+y}}{\arctg(x) + \arctg(z)} + \sqrt[3]{x^6 + \ln^2 y}.$$

При $x=-2.235 \times 10^{-2}$, $y=2.23$, $z=15.221$ **$\varphi=39.374$.**

25.
$$\psi = \left| \frac{y}{x^x} - \sqrt[3]{\frac{y}{x}} \right| + (y-x) \frac{\cos y - \frac{z}{(y-x)}}{1 + (y-x)^2}.$$

При $x=1.825 \times 10^2$, $y=18.225$, $z=-3.298 \times 10^{-2}$ **$\psi=1.2131$.**

$$b = y^{\sqrt[3]{|x|}} + \cos^3(y) \frac{|x-y| \left(1 + \frac{\sin^2 z}{\sqrt{x+y}} \right)}{e^{|x-y|} + \frac{x}{2}}.$$

26.

При $x=6.251$, $y=0.827$, $z=25.001$ **$b=0.7121$** .

$$c = 2^{(y^x)} + (3^x)^y - \frac{y \left(\operatorname{arctgz} - \frac{\pi}{6} \right)}{|x| + \frac{1}{y^2 + 1}}.$$

27.

При $x=3.251$, $y=0.325$, $z=0.466 \times 10^{-4}$ **$c=4.25$** .

$$f = \frac{\sqrt[4]{y + \sqrt[3]{x-1}}}{|x-y|(\sin^2 z + \operatorname{tg} z)}$$

28.

При $x=17.421$, $y=10.365 \times 10^{-3}$, $z=0.828 \times 10^5$ **$f=0.33056$** .

$$g = \frac{y^{x+1}}{\sqrt[3]{|y-2|} + 3} + \frac{x + \frac{y}{2}}{2|x+y|} (x+1)^{-1/\sin z}$$

29.

При $x=12.3 \times 10^{-1}$, $y=15.4$, $z=0.252 \times 10^3$ **$g=82.8257$** .

$$h = \frac{x^{y+1} + e^{y-1}}{1 + x|y - \operatorname{tg} z|} (1 + |y-x|) + \frac{|y-x|^2}{2} - \frac{|y-x|^3}{3}$$

30.

При $x=2.444$, $y=0.869 \times 10^{-2}$, $z=-0.13 \times 10^3$ **$h=-0.49871$** .

ЛАБОРАТОРНА РОБОТА №15-16

Складання програми з перевантаженням операторів методами класу та зовнішніми функціями. Відлагодження та тестування програми.

Мета: навчитись складати, відлагоджувати та тестувати програми з перевантаженням операторів методами класу та зовнішніми функціями.

Частина 1

Постановка задачі

Створити консольний додаток C++ та клас з ім'ям `D##`, де `##` — двозначний порядковий номер студента в групі (номер варіанту). Передбачити файли специфікації та реалізації для класу. У класі оголошений вказівник на масив, є конструктори, деструктор та інші методи. Перевантажити бінарні оператори `+` і `-` для складання об'єктів, отримання суми об'єкту та числа, а також отримання різниці об'єктів за допомогою функцій-членів. Використати дружню функцію для виведення стану об'єкту. Використати за основу запропонований нижче програмний код, доповнити його, проініціалізувати поля, додати необхідні перевірки та звільнення пам'яті, вихід за межі кількості елементів масиву тощо (у випадках різної кількості елементів в двох об'єктах). Відлагодити, протестувати та прокоментувати програму. У звіті навести індивідуальний хід роботи.

Порядок виконання роботи

У зразку файл специфікації `D.h` містить оголошення типу класу `D`. У класі оголошені закриті дані.

Забезпечено клас `D` двома конструкторами: конструктором за замовчанням та конструктором з параметрами, а також деструктором.

Конструктор за замовчанням та деструктор вже присутні в специфікації класу `D`. Для оголошення конструктору з параметрами потрібно описати прототип цієї функції. Нижче приведено лістинг файлу специфікації класу `D`.

```
// d.h - специфікація класу D
#pragma once
```

```

#endif // _MSC_VER > 1000
class D
{
    int* m_p ; int m_n ;
public:
    D ( int = 10 ) ;
    D ( D& ) ;
    ~ D ( ) ;
    int getN ( void ) const ;

    int& operator [ ] ( const unsigned int ) const ;
    D& operator = ( D& ) ;
    D operator + ( const D& ) const ;
    D operator + ( const int& ) const ;
    D operator - ( const D& ) const ;

    friend void view ( const D& ) ;
} ;

```

Файл реалізації класу має бути відредагований у відповідності з текстом:

```

// d.cpp - реалізація D
#include "d.h"
D :: D ( int n ) : m_n ( n )
{
    m_p = new int [ m_n ] ;
    for ( int i = 0; i < m_n; i++ ) m_p [ i ] = 0 ;
}
D :: D ( D& o ) : m_n ( o.m_n )
{
    m_p = new int [ m_n ] ;
    for ( int i = 0; i < m_n; i++ ) m_p [ i ] = o.m_p [ i ] ;
}
D :: ~ D ( ) { delete [ ] m_p ; }
int D :: getN ( void ) const { return m_n ; }

int& D :: operator [ ] ( const unsigned int i ) const
{
    return m_p [ i ] ; }

D& D :: operator = ( D& right )
{
    if ( m_n != right.m_n )
    { delete [ ] m_p ; m_n = right.m_n ; m_p = new int [ m_n ] ; }
    for ( int i = 0; i < m_n; i++ ) m_p [ i ] = right.m_p [ i ] ;
    return *this ;
}

D D :: operator + ( const D& right ) const
{
    D tmp ( m_n ) ;
    for ( int i = 0; i < tmp.m_n ; i++ )
        tmp.m_p [ i ] = m_p [ i ] + right.m_p [ i ] ;
    return tmp ;
}

```

```

D D :: operator + ( const int& right ) const
{
    D tmp ( m_n ) ;
    for ( int i = 0; i < tmp.m_n ; i++ )
        tmp.m_p [ i ] = m_p [ i ] + right ;
    return tmp ;
}

D D :: operator - ( const D& right ) const
{
    D tmp ( m_n ) ;
    for ( int i = 0; i < tmp.m_n ; i++ )
        tmp.m_p [ i ] = m_p [ i ] - right.m_p [ i ] ;
    return tmp ;
}

```

Головна функція проекту має бути відредагована за зразком:

```

#include "d.h"
// прототип функції
void view ( const D& ) ;           // виведення стану об'єкту
// main.cpp - главная функция
int main(int argc, char* argv[])
{
    D o1 ( 5 ), o2 ( 5 ) ;
    for ( int i = 0; i < o1.getN ( ); i++ )
    {
        o1 [ i ] = ( i + 1 ) * 2 ; o2 [ i ] = i ;
        cout << "\nol:\t\t" ;      view ( o1 ) ;
        cout << "o2:\t\t" ; view ( o2 ) ;

        D res ;
        res = o1 + o2 ;
        cout << "\nres = o1 + o2:\t" ; view ( res ) ;

        int x ( 5 ) ; D o3 ( 5 ) ;
        cout << "\no3:\t\t" ;      view ( o3 ) ;
        res = o3 + x ;
        cout << "res = o3 + 5:\t" ; view ( res ) ;

        res = o3 - o2 ;
        cout << "\nres = o3 - o2:\t" ; view ( res ) ;
        res = o2 - o3 ;
        cout << "res = o2 - o3:\t" ; view ( res ) ;

        cout << "\nol:\t\t" ;      view ( o1 ) ;
        cout << "o2:\t\t" ;      view ( o2 ) ;
        cout << "o3:\t\t" ;      view ( o3 ) ;
        return 0 ;
    }
}

```

```
// функція виведення стану об'єкту
void view ( const D& o )
{   for ( int i = 0; i < o.m_n; i++ ) cout << o [ i ] << '\t' ;
    cout << endl ;
}
```

Оформити звіт, пояснити в ході роботи використання перевантаження операторних функцій. Прокоментувати в програмному коді, де яка операція перевантажується, де сторонньою функцією, де методом класу.

Частина 2

Постановка задачі

1 Забезпечити ініціалізацію об'єктів o1, o2 та o3 значеннями, які вводяться користувачем (кількість елементів масиву для кожного об'єкту може бути своя).

2 Забезпечити для об'єктів класу D####, де ## — двозначний порядковий номер студента в групі, перевантаження оператора * не методом класу, а зовнішньою функцією відповідно варіанту:

- непарні варіанти: множення об'єкта на число;
- парні варіанти: множення об'єкта на об'єкт.

Протестувати програму для введеної користувачем кількості елементів масиву:

- для випадку, що кількість в o1 менша, ніж в o2;
- для випадку, що кількість в o2 менша, ніж в o1;
- для випадку однакової кількості в o1 та o2.

3 Прокоментувати програму, оформити звіт, пояснити у звіті відмінності у перевантаженні операції методом та сторонньою функцією.

ЛАБОРАТОРНА РОБОТА № 17-18

Проектування класу з статичними членами-даними та статичними членами-функціями. Відлагодження та тестування програми.

Мета роботи: спроектувати клас з статичними даними та з статичними методами, створити об'єкти класу з статичними даними та з статичними методами, закріпити навички з відлагодження та тестування програми.

Частина I

Завдання. Оголосити клас, який містить статичні поля, проініціалізувати їх, пояснити використання в програмі, створити два об'єкти класу. З'ясувати, як впливає зміна значення статичного поля одним об'єктом класу на значення цього статичного поля для іншого об'єкту класу.

Порядок виконання роботи.

Оголосити клас `Stat_d` та внести специфікацію та реалізацію за зразком, наведеним в лістингах 17.1-17.2.

Лістинг 17.1 – Специфікація класу

```
class Stat_d {
public:
    static double A;
    int m_x, m_y;
private:
    static int B;
public:
    // Stat_d();
    Stat_d(int,int );
    virtual ~Stat_d();
void setB(int);
int getB ();
}
```

Лістинг 17.2 – Реалізація класу

```
Stat_d::Stat_d(): m_x(0), m_y(0) { }
Stat_d::Stat_d(int x,int y): m_x(x), m_y(y) { }
Stat_d::~~Stat_d() { }

void Stat_d::setB(int _b) {
    B = _b;
```

```

}
int Stat_d::getB() {
    return B;
}

```

За межами головної функції та за межами класу проініціалізувати статичні дані класу. В головній функції оголосити два об'єкти. В результаті виконання програми видно, що зміни значень закритого статичного даного В у об'єкті а відображаються на об'єкті b, і навпаки. Для доступу до значень В потрібні методи через специфікатор private. Доступ до даного А відбувається звичайним способом, тому що цей статичний член-дане має специфікатор доступу public. Фрагмент програмного коду головної функції наведений у лістингу 17.3.

Лістинг 17.3 – Фрагмент програмного коду головної функції

```

double Stat_d ::A = 2.75;
int Stat_d ::B = 777;
int main(int argc, char* argv[]) {
    Stat_d a ( 10, 10 ), b ( -10, -10 ) ;
    cout << "a\t" << a.m_x << '\t' << a.m_y << endl ;
    cout << "b\t" << b.m_x << '\t' << b.m_y << endl ;
    cout << "for a and b static A =\t" << Stat_d :: A << endl ;
    cout << "for a static B =\t" << a.getB ( ) << endl ;
    cout << "for b static B =\t" << b.getB ( ) << endl ;
    Stat_d :: A = 1.2e308 ;
    cout << "after change A =\t" << Stat_d    A << endl ;
    a.setB ( 100 ) ;
    cout << "after change in a b.B =\t" << b.getB ( ) << endl ;
    b.setB ( -100 ) ;
    cout << "after change in b a.B =\t" << a.getB ( ) << endl ;
    return 0;
}

```

Відлагодити програму. Пояснити роботу програми, в яких випадках в класі оголошують статичне поле і як ним користуються. Індивідуально прокоментувати програмний код.

Частина II

Завдання. Додати в проект новий клас зі статичними методами, пояснити мету використання в програмі даних статичних полів та методів, замінити в головній функції символи "???" на номер за варіантом і пояснити подальше виведення в консолі. Відлагодити, протестувати і прокоментувати програму.

Порядок виконання роботи.

Оголосити клас Stat_m та внести специфікацію та реалізацію за зразком, наведеним в лістингах 18.1-18.2.

Лістинг 18.1 – Специфікація класу

```
class Stat_m
{
public:
    Stat_m ( char* );
    Stat_m ( );
    virtual ~Stat_m ( );
    static void initAmount ( const int ) ;

    static int getAmount ( ) ;

private
    static int incObj ( );

    static int objAmount ;
    int lenS ( ) ;
    int n;
    char m_s [ 40 ] ;
};
```

Лістинг 18.2 – Реалізація класу

```
Stat_m :: Stat_m ( char* s ): n(40)
{
    strncpy ( m_s, s, 40 ) ; incObj ( ) ;
}
Stat_m :: Stat_m ( ): n(40)
{
    m_s [ 0 ] = '\0 ; incObj ( ) ;
}
Stat_m :: ~Stat_m ( )
{ --objAmount ; }

void Stat_m ::initAmount ( const int v )
{
    objAmount = v ;
}
int Stat_m ::getAmount ( )
{
    n++;
    return objAmount ;
}
int Stat_m ::incObj ( )
{ return ++objAmount ; }

int Stat_m ::lenS ( )
{
    return strlen ( m_s ) ;
}
```

```
}
```

За межами головної функції та за межами класу оголосити статичну змінну класу objAmount. В межах головної функції проініціалізувати статичну змінну класу objAmount за допомогою методу initAmount ().

Фрагмент програмного коду головної функції наведений у лістингу 18.3.

Лістинг 18.3 – Фрагмент програмного коду головної функції

```
int Stat_m :: objAmount ;

int main(int argc, char* argv[])
{

    Stat_m :: initAmount ( 0 ) ;
    Stat_m o1 ( "Kyiv" ), o2 ( "Dnipro" ) ;
    Stat_m o [ 12 ] , o3 ;
    Stat_m* p = new Stat_m [ ?? ] ; //замінити ?? на номер за
    варіантом
    cout << Stat_m :: getAmount ( ) << endl ;
    delete [ ] p ;
    cout << Stat_m :: getAmount ( ) << endl ;
    return 0;
}
```

У лістингу головної функції для закритої статичної змінної objAmount класу Stat_m оголошені три статичних методи, два з них є відкритими, а один - закритим. Член-дане objAmount використовується для підрахунку існуючих в оперативній пам'яті об'єктів класу. Для ініціалізації objAmount цілим значенням реалізований відкритий статичний метод initAmount (), для одержання значення - getAmount (). Закритий статичний метод incObj () призначений для збільшення на 1 значення статичної змінної класу objAmount. Цей метод викликається всіма конструкторами класу. Щоб програма визначала правильно кількість існуючих об'єктів, у деструктор доданий декремент objAmount.

Додати підключення необхідних бібліотек. Відлагодити програму.

У звіті навести індивідуальний хід роботи (копіаст тексту завдання не враховується), в якому пояснити, які статичні поля класу оголошені, де проініціалізовані і чому їх потрібно було зробити не звичайними, а статичними, які статичні методи і чому саме статичні, а не звичайні. Знайти в програмі помилку. Пояснити роботу програми. Індивідуально прокоментувати програмний код.

ЛАБОРАТОРНА РОБОТА № 19-20

Складання та відлагодження програми з використанням шаблонів функцій

Мета: навчитись складати та відлагоджувати програми з використанням шаблонів функцій.

Теоретична частина

У мові C++ існує можливість замість декількох ідентичних по операціях функцій створювати тільки одну універсальну функцію, що називається шаблоном або шаблонною функцією.

Шаблонна функція (template function) — це функція, що повністю контролює відповідність типів даних, які задаються їй як параметри.

Оголошення шаблонної функції повторює визначення звичайної функції, але першим рядком в оголошенні обов'язково повинний бути рядок наступного виду (замість ключового слова class може бути використано typename):

```
template < class ім'я_типу1, ..., class ім'я_типа >
```

У кутових дужках < > після ключового слова template вказується список формальних параметрів шаблону. Ім'я_типу називають параметром типу, що являє собою ідентифікатор типу й використовується для визначення типу параметра в заголовку шаблонної функції, типу значення, що повертається функцією й типу змінних, що оголошені у тілі функції. Кожний параметр типу повинен мати унікальний ідентифікатор, що може бути використаний у різних шаблонах. Кожному параметру типу повинне передувати ключове слово class або typename.

За рядком template йде рядок зі стандартним оголошенням функції, де у списку параметрів використовуються як параметри типу, так і будь-які інші припустимі базові або похідні типи даних.

Частина 1

Постановка задачі. Створити консольний додаток C++ на основі наведених нижче фрагментів програмного коду. В ході роботи пояснити доцільність використання шаблонної функції замість звичайної, навести назви і пояснити призначення параметрів шаблону і де вони використані в програмі.

Прокоментувати програмний код своїми коментарями за темою і завданням, в тому числі виклики функцій.

Порядок виконання роботи

В консольному додатку створити шаблонну функцію для виведення масивів різних типів.

Як параметр типу шаблонна функція використовує ідентифікатор `T`, який застосовується для покажчика `p` в списку параметрів шаблонної функції. Покажчик визначає адресу масиву, елементи якого необхідно вивести.

Другий параметр має явний тип `size_t`. Функція не повертає значення і має тип `void`. Для кожного масиву генерується функція з відповідним типом. Замість параметра типу `T` підставляються значення `int` для масиву `ia`, `double` для масиву `da`, `long` для масиву `la` і `char` для масиву `s`. У кожному виклику функції `view()` вказується ім'я масиву, відповідне його адресі, а також розмір масиву, який обчислюється за допомогою оператора `sizeof()`.

Шаблон `reorg()` дозволяє змінювати порядок проходження елементів в масиві будь-якого числового типу так, що на початку розташовуються негативні елементи, після яких записуються позитивні. Важливо пам'ятати, що покажчик `p` на вихідний масив не можна оголошувати з специфікатором `const`, тому що зміни здійснюються безпосередньо у вихідному масиві. Функція використовує єдиний параметр типу `T`, застосовуваний в заголовку і в тілі шаблону. Головна функція перевіряє роботу шаблону масиву довгого цілого типу `long` і масиву чисел з плаваючою точкою `float`.

```
// шаблонна функція виведення елементів масиву
template < class T > void view ( const T *p, const size_t size ) {
    for ( int i = 0; i < size; i++ )
        cout << p [ i ] << '\t' ;
    cout << endl ;
}
// шаблонна функція, реорганізує елементи в масиві: спочатку від'ємні і
0, потім додатні
template < class T > void reorg ( T *p, const int size ) {
    T tmp ;
    for ( int i = 0, j = 0 ; i < size ; i++ ) {
        if ( p [ i ] > 0 )    continue ;
        tmp = p [ i ] ;
        for ( int k = i ; k > j ; k-- ) p [ k ] = p [ k - 1 ] ;
        p [ j ] = tmp ;
        j++ ;
    }
}
```

```

    }
}
// головна функція
int main ( ) {
    int ia [ ] = { 1, -3, 0, -9, 2, 7, 0, -19 } ;
    double da [ ] = { 0.0875, 1.25, -3, 0.0, -7.986 } ;
    long la [ ] = { 1L, -2L, 3L, -100L } ;
    float fa [ ] = { -2.12, 1.5, -3.25, -17.2 } ;
    char s [ ] = "Vivat!" ;
    cout << "\nint Array\n" ;
    view ( ia, sizeof ( ia ) / sizeof ( int ) ) ;
    cout << "\ndouble Array\n" ;
    view ( da, sizeof ( da ) / sizeof ( double ) ) ;
    cout << "\nlong Array\n" ;
    view ( la, sizeof ( la ) / sizeof ( long ) ) ;
    cout << "\nchar Array\n" ;
    view ( s, strlen ( s ) ) ;

    long la [ ] = { 1L, -2L, 3L, -100L, -200L } ;
    float fa [ ] = { -2.12, 1.5, -3.25, -17.2 } ;
    int n = sizeof ( la ) / sizeof ( long ) ;
    cout << "\nInitial long array\n" ; view ( la, n ) ;
    cout << "Reorganization\n" ; reorg ( la, n ) ; view ( la, n ) ;
    n = sizeof ( fa ) / sizeof ( float ) ;
    cout << "\nInitial float array\n" ; view ( fa, n ) ;
    cout << "Reorganization\n" ; reorg ( fa, n ) ; view ( fa, n ) ;
    return 0 ;
}

```

Частина 2

Постановка задачі. Самостійно створити шаблон функції у відповідності з варіантом. Продемонструвати роботу програми з масивами різних типів (наприклад, окремо з цілочисельним і дійсним). Прокоментувати програму з погляду на використання шаблонів.

№ варіанту	Призначення функції для роботи з масивом (не сортування, а перестановка)
1	Спочатку розташовуються нульові елементи, після яких записуються позитивні, а потім негативні
2	Якщо в масиві зустрічаються негативні елементи, то їх значення подвоюється
3	Спочатку розташовується мінімальний елемент, після нього розташовуються всі інші, крім мінімального
4	Якщо в масиві зустрічаються позитивні елементи, то до кожного з них додається значення 4
5	Якщо в масиві зустрічаються негативні елементи, то їх значення змінюється на протилежне
6	Спочатку розташовується максимальний елемент, після нього розташовуються всі інші, крім тих, що дорівнюють максимальному
7	Спочатку розташовуються негативні елементи, після яких записуються

	позитивні, а потім нульові
8	Якщо в масиві зустрічаються позитивні елементи, то їх значення складається з максимальним
9	Спочатку розташовується подвоєний мінімальний елемент, після нього розташовуються всі інші, крім тих, що дорівнюють мінімальному
10	Спочатку розташовуються позитивні елементи, після яких записуються нульові, а потім негативні
11	Якщо в масиві зустрічаються негативні елементи, то їх значення складається з мінімальним
12	Якщо в масиві зустрічаються нульові елементи, то до кожного такого елемента додається середнє арифметичне мінімального і максимального значень
13	Якщо в масиві зустрічаються негативний елемент, який стоїть на парному місці, то його значення змінюється на протилежне плус мінімальне
14	Спочатку розташовується максимальний елемент, якщо він стоїть на непарному місці, або мінімальний елемент в іншому випадку, після нього розташовуються всі інші, елементи, які стоять на парних місцях
15	Спочатку розташовуються негативні елементи, які стоять на парних місцях, після яких записуються позитивні, а потім всі інші
16	В масиві міняються місцями елементи, які стоять на парних і непарних місцях
17	Спочатку розташовуються нульові елементи, після яких записуються позитивні, а потім негативні
18	Якщо в масиві зустрічаються негативні елементи, то їх значення подвоюється
19	Спочатку розташовується мінімальний елемент, після нього розташовуються всі інші, крім тих, що дорівнюють мінімальному
20	Якщо в масиві зустрічаються позитивні елементи, то до кожного з них додається значення 5
21	Якщо в масиві зустрічаються негативні елементи, то їх значення змінюється на протилежне
22	Спочатку розташовується максимальний елемент, після нього розташовуються всі інші, крім тих, що дорівнюють максимальному
23	Спочатку розташовуються негативні елементи, після яких записуються позитивні, а потім нульові
24	Якщо в масиві зустрічаються позитивні елементи, то їх значення складається з максимальним
25	Спочатку розташовується подвоєний мінімальний елемент, після нього розташовуються всі інші, крім тих, що дорівнюють мінімальному
26	Спочатку розташовуються позитивні елементи, після яких записуються нульові, а потім негативні
27	Спочатку розташовуються нульові елементи, після яких записуються позитивні, а потім негативні
28	Якщо в масиві зустрічаються негативні елементи, то їх значення подвоюється

ЛАБОРАТОРНА РОБОТА № 21-22

Складання, відлагодження та тестування програми з використанням простого успадкування класів

Мета: навчитись складати програми з використанням простого успадкування класів. Закріпити навички з відлагодження та тестування програм

Теоретичні відомості

Коли ви породжуєте клас з базового класу, приватні елементи базового класу доступні похідному класу тільки через інтерфейсні функції базового класу. Таким чином, похідний клас не може безпосередньо звернутися до приватних елементів базового класу, використовуючи операцію доступу.

Конструктор похідного класу викликає конструктор базового класу. Щоб викликати конструктор базового класу явно, вкажіть ім'я конструктора базового класу з необхідними параметрами після двокрапки в конструкторі похідного класу в списку ініціалізації.

Частина 1

Постановка задачі

Створити базовий клас Basis, елементи якого (дані і методи обробки цих даних) успадковує клас Inherit. Проініціалізувати поля в конструкторі за замовчуванням. Продемонструвати роботу, пояснити, як що успадковується, в якій послідовності створюється.

Порядок виконання роботи

Створити консольний проект C++ і два класи у відповідності:

- Basis — базовий;
- Inherit — похідний від Basis.

Створити класи можна наступним чином:

- базовий клас створити першим, як звичайно;

- для класу Inherit, який є похідним від базового класу, потрібно, крім того, у полі Derived from (успадкований від) внести ім'я базового класу Basis та у полі As (успадковано як) вибрати специфікатор доступу до класу – public (фактично буде рядок програмного коду `class Inherit : public Basis { ... } ;`);

- внести зміни згідно наведеного нижче програмного коду в заголовні файли класів, файли з реалізацією класів та головну функцію.

Оскільки клас Inherit є похідним класу Basis, клас Inherit може звертатися до загальних елементів класу Basis, як якщо б всі ці елементи були визначені усередині класу Inherit.

```
// Basis.h - Специфікація базового класу Basis
class Basis {
public:
    int a, b;
    Basis();
    Basis(int, int);
    virtual ~Basis();
    int Sum_B();
    int Razn_B();
};

// Inherit.h -Специфікація похідного класу Inherit
#include "Basis.h"
class Inherit : public Basis {
    int mult;
    float div;
    int proc;
public:
    Inherit();
    Inherit(int a_bas, int b_bas, int pr_in);
    virtual ~Inherit();
    void friend view(Inherit&)
    void Set_mult();
    void Set_div();
};
```

В заголовні файли (.h) класів Basis, Inherit, внесені оголошення методів-конструкторів з параметрами та методів відповідних класів, а у файли (.cpp) тих же класів – реалізація методів.

Виконувати збереження, компіляцію та перебудову проекту за необхідністю. Ці дії краще виконувати кожного разу після редагування пари файлів (.h) і (.cpp) для відповідного класу.

```
// Реалізація класу Basis
Basis::Basis()
{
}
Basis::Basis(int x, int y): a(x), b(y)
{
cout << "Basis a= " << a << "b= " << b << endl;
}
Basis::~~Basis()
{
}
int Basis::Sum_B()
{
    return a+b;
}
int Basis::Razn_B()
{
    return a-b;
}

// Реалізація класу Inherit
Inherit::Inherit()
{
}
Inherit::~~Inherit()
{
}
Inherit::Inherit(int a_bas, int b_bas, int pr_in): proc(pr_in),
Basis(a_bas, b_bas)
{
    cout << "Inherit a_bas= " << a_bas << "b_bas= " << b_bas < endl;
    cout << "Inherit a= " << a << "b= " << b << endl;
}
void Inherit::Set_mult()
{
    cout << "Inherit Set_mult: a= " << a << " b= " << b << endl;
    if ((a==0) || (b==0)) mult = 0;
    else mult = a*b;
    cout << "Inherit mult= " << mult << endl;
}
void Inherit::Set_div()
{
    if (b != 0) {div = a/b; cout << "Inherit div= " << div << endl;}
    else cout << "Inherit div ne obchislene " << endl;
}
```

Головна функція проекту може бути відредагована у відповідності з текстом:

```
void view(Inherit& ob)
```

```

{
cout << "ob a= " << ob.a << " b= " << ob.b << " proc= " << ob.proc <<
endl;
}
int main(int argc, char* argv[])
{
inherit ob1(5,6,20);
int summa = ob1.Sum_B();
int razn = ob1.Razn_B();
cout << "summa= " << summa << " razn= " << razn << endl;
view(ob1);
ob1.Set_div();
ob1.Set_mult();
return 0;
}

```

Відлагодити програму, провести тестування, пояснити результати, оформити звіт з індивідуальним ходом роботи та коментарями.

В звіті пояснити:

- що таке просте поодиноке успадкування і як відповідно називаються класи;
- як успадковані елементи класу Basis і який до них доступ через об'єкт класу Inherit.

Частина 2

Провести модифікації програми, яка була виконана в частині 1, таким чином:

- значення даних базового класу потрібно вводити з клавіатури (тобто, їх слід передавати через параметри конструктора похідного класу);
- значення даних похідного класу потрібно вводити з клавіатури;
- передбачити захищений специфікатор доступу для полів базового класу;
- змінити специфікатор доступу перед базовим класом з public на:
 - парні варіанти – private
 - непарні варіанти – protected

та внести в зв'язку з цим зміни до програмного коду та пояснити в звіті;

- замість введення значення для поля proc додати до класу Inherit метод, який розраховує відсоток поля a базового класу від суми полів базового класу;
- додати до класу Basis метод, який виводить значення даних класу та скористатися цим методом з похідного класу.

Відлагодити програму, провести тестування, пояснити виконані зміни, отримані результати, оформити звіт.

ЛАБОРАТОРНА РОБОТА № 23-24

Проектування конструкторів класів при побудові ієрархії класів. Відлагодження та тестування програми з перевизначенням методів базового класу

Мета: навчитись проектувати конструктори класів при побудові ієрархії класів, відлагоджувати та тестувати програми з перевизначенням методів базового класу.

Теоретичні відомості

Коли породжується клас з базового класу, конструктор похідного класу повинен викликати конструктор базового класу. Щоб викликати конструктор базового класу, в списку ініціалізації конструктора похідного класу слід вказати ім'я конструктора базового класу з необхідними параметрами.

Похідний клас може перевизначити метод базового класу, якщо такому методу необхідно задати іншу реалізацію.

У лістингу надається програмний код, у відповідності з яким створюються ієрархія трьох рівнів і два класи послідовно використовуються як базові для створення класу-нащадку.

```
class Bas1
{
public:
    Bas1():a(0) { cout << "Bas1 constructor... ";}
    virtual ~Bas1() { cout << "Bas1 destructor... ";}
    void MetBas1() const { cout << "MetBas1!... ";}
private:
    int a;
};

class Bas2: public Bas1;
{
public:
    Bas2():b(0) { cout << "Bas2 constructor... "; }
    virtual ~Bas2() { cout << "Bas2 destructor... "; }
```

```

        void MetBas2( ) const { cout << "MetBas2... "; }
        void Met_2Bas2() const {
            cout << " Met_2Bas2! "; }
private:
    int b;
};

class Nasled: public Bas2{
public:
    void MetBas2() const { MetBas1(); }
    Nasled():c(0) {      cout << "Nasledconstructor... "; }
    ~ Nasled() {      cout << "Nasleddestructor... "; }
private:
    int c;
};

const int MagicNumber = 2;

int main(int argc, char* argv[])
{
    Bas1* ArRan[MagicNumber] ;
    Bas2* ArAvi[MagicNumber] ;
    Bas1 * pBas1;
    Bas2 * pBas2;
    int choice,i;
    for (i=0; i<MagicNumber; i++)
    {
        cout << "\n(1)Bas1 (2) Nasled: ";
        cin >> choice;
        if (choice == 2)
            pBas1 = new Nasled
        else
            pBas1 = new Bas1;
        ArRan[i] = pBas1;
    }
    for (i=0; i<MagicNumber; i++)
    {

```

```

cout << "\n(1)Bas2 (2) Nasled: ";
cin >> choice;
if (choice == 2)
pBas2 = new Nasled;
else
pBas2 = new Bas2;
ArAvi[i] = pBas2;
}
cout << endl;
for (i=0; i<MagicNumber; i++)
{
cout << "\nArRan["<< i << "] : ";
ArRan[i]->MetBas1();
delete ArRan[i];
}
for (i=0; i<MagicNumber; i++)
{cout << "\nArAvi["<< i << "]: ";
ArAvi[i]->MetBas2();
ArAvi[i]->Met_2 Bas2();
delete ArAvi[i];
}
return 0;
}

```

Конструктор та деструктор виводять на екран повідомлення про свою роботу, а метод MetBas1() виводить на екран повідомлення "MetBas1!...".

Клас Bas2 крім конструктора та деструктора містить методи MetBas2() та Met_2Bas2(), кожний з яких виводить на екран відповідне повідомлення. В реальних програмах ці методи можуть виконувати конкретну роботу.

Клас Nasled походить від базового класу Bas2, а той, в свою чергу, від Bas1. Клас Nasled перевизначає метод MetBas2() так, щоб він викликав метод MetBas1(), який є похідним від класу Bas1.

Зверніть увагу, що кожний раз при створенні об'єкту класу Nasled на екран виводяться повідомлення, які свідчать про те, що створюється і частина Bas2 і

частина Bas1 єдиного об'єкту класу Nasled. При видаленні об'єкту класу Nasled аналогічно видаляються його частини, які відносяться до класів Bas2 та Bas1.

Частина I

Постановка завдання. На основі запропонованого програмного коду створити ієрархію класів трьох рівнів, кожний клас — специфікацію і реалізацію в окремих файлах. Пояснити послідовність виклику конструкторів і деструкторів і створення та знищення об'єктів і ієрархії. Пояснити, що таке перевизначення методів, який або які методи перевизначаються, а які ні і чому. Прокоментувати в головній функції, який метод викликається з якого класу.

Відлагодити програму. Оформити звіт, в звіті пояснити наведені питання.

Частина II

Завдання. Спроекувати ієрархію класів, передбачити конструктори з параметрами та ініціалізацію, перевизначені методи для обчислення та виведення.

Розробити програму, яка містить в окремих файлах базовий клас та клас-нащадок. Базовий клас повинен містити двомірну матрицю в якості члена-даного, конструктор за замовченням та конструктор з параметрами, в якому ініціалізуються члени-дані кількість рядків та кількість стовпців матриці.

Базовий клас повинен мати методи:

- m_Input() для введення користувачем матриці;
- m_Calc() для виведення повідомлення;
- m_Output() для виведення матриці.

Клас-нащадок повинен працювати з матрицею базового класу та містити конструктор за замовченням та конструктор з параметрами для передачі значень в конструктор базового класу. Перевизначити метод m_Output() для виведення результату обчислення згідно індивідуального завдання.

В головній функції надати можливість користувачу ввести кількість рядків та кількість стовпців матриці та передати для створення об'єкту класу-нащадку з використанням конструктора з параметрами.

Індивідуальні завдання за варіантами.

1. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив сум елементів рядків матриці.
2. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив добутків елементів стовпчиків матриці.
3. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив з кількістю непарних елементів в стовпчиках матриці.
4. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив з кількістю нульових елементів в рядках матриці.
5. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив сум від'ємних елементів рядків матриці.
6. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив сум додатних елементів рядків матриці.
7. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив добутків від'ємних елементів рядків матриці.
8. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив добутків додатних елементів рядків матриці.
9. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив сум непарних елементів рядків матриці.
10. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив сум парних елементів рядків матриці.
11. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив з кількістю непарних елементів в рядках матриці.
12. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив з кількістю парних елементів в рядках матриці.
13. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив середньоарифметичних значень рядків матриці.
14. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив середньгеометричних значень стовпців матриці.
15. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив сум елементів в парних стовпцях матриці.

16. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив сум елементів в непарних рядках матриці.
17. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив добутків елементів в парних рядках матриці.
18. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив добутків елементів в непарних стовпцях матриці.
19. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` куб максимального елемента матриці.
20. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` квадрат мінімального елемента матриці.
21. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив сум модулів елементів в парних рядках матриці.
22. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив добутків модулів елементів в непарних стовпцях матриці.
23. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив середньоарифметичних значень стовпців матриці.
24. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив середньгеометричних значень рядків матриці.
25. Розрахувати в перевизначеному методі класу-нащадку `m_Calc()` масив сум від'ємних значень стовпців матриці.

Оформити звіт, в якому на прикладі виконаних завдань пояснити принципи проектування конструкторів класів при побудові ієрархії класів, перевизначення методів базового класу.

ЛАБОРАТОРНА РОБОТА № 25-26

Складання та відлагодження програми з використанням шаблону класу

Мета: навчитись складати та відлагоджувати програми з використанням шаблонів класів

Теоретичні відомості

Шаблонні класи створюються аналогічно шаблонним функціям — перед назвою класу записується ключове слово `template`. Розглянемо використання шаблонів класів на прикладі програми побудови списку з десяти елементів. Ці елементи можуть бути різних типів. В прикладі функція `push` визначена всередині класу, а функція `pop` — ззовні. Для всіх функцій, визначених за межами класу, необхідно обов'язково вказувати ключове слово `template`. Список параметрів типів перед ім'ям функції співпадає з тим, яке вказується перед ім'ям класу.

Частина 1

У лістингу надається програмний код, у відповідності з яким створюються класи.

Лістинг 11.1 – Програмний код шаблону.

```
template <class Type>
class stack {
private:
    int top;
    Type s[10];

public:
    stack (): top(-1)
    {}

    void push(Type var)
    {
        top++;
        s[top] = var;
    }

    Type pop();
};

template <class Type>
Type stack<Type>::pop()
{
    Type var = s[top];
```

```

    top--;
    return var;
}

```

У шаблонних класів є особливість — крім стандартних типів вони можуть працювати і з користувацькими. Для демонстрації цього в прикладі створений простий клас `warrior`, наведений в лістингу 11.2.

Лістинг 11.2 – Програмний код класу `warrior`.

```

class warrior {
public:
    int health;
    warrior () : health(0) {}
};

```

При створенні об'єкту після імені класу треба поставити кутові дужки, в яких вказати необхідний тип. Після цього об'єкти використовуються як зазвичай. Також тепер в списках можна розміщувати змінні типа `warrior`, що наведено в лістингу 11.3.

Лістинг 11.1 – Програмний код головної функції.

```

int main(int argc, char* argv[]) {
    stack<int> s1;
    stack<float> s2;

    s1.push(3);
    s1.push(2);

    s2.push(0.5);

    stack<warrior> s;

    warrior w1;
    warrior w2;
    warrior w3;

    s.push(w1);
    s.push(w3);

    s.pop();
    s.push(w2);
    return 0;
}

```

Розмістіть кожний клас в окремому файлі (для шаблону разом специфікацію і реалізацію).

Допишіть можливість введення кількості елементів списку з екрану. Додайте в програмі перевірку, щоб в список не можна було ввести більше елементів, ніж це значення.

В шаблоні класу створіть метод, який виводитиме на екран вміст списку. Продемонструйте роботу методу на прикладі виклику для об'єктів `s1`, `s2`.

Передбачте перевантаження виведення для класу `waitior`.

Відлагодіть програму.

Прокоментуйте роботу програми, оформіть звіт, поясніть використання шаблонів класів та хід роботи.

Частина II

Розробити програму з використанням шаблону класу, в якому вводяться елементи масиву, проводиться обчислення за варіантом й виводиться на екран результат.

В головній функції створити два об'єкти класу, один з яких передбачає роботу з масивом цілих чисел, а другий - дійсних.

Варіант 1.

Обчислити:

1. Суму негативних елементів масиву.
2. Добуток елементів масиву, розташованих між першим максимальним і першим мінімальним елементами.
3. Упорядкувати елементи масиву за зростанням.

Варіант 2.

Обчислити:

1. Суму позитивних елементів масиву.
2. Добуток елементів масиву, розташованих між першим максимальним за модулем і останнім мінімальним по модулю елементами.
3. Упорядкувати елементи масиву за спаданням.

Варіант 3.

Обчислити:

1. Добуток елементів масиву з парними номерами.
2. Суму елементів масиву, розташованих між першим і останнім нульовими елементами.
3. Перетворити масив таким чином, щоб спочатку розташовувалися всі позитивні елементи і 0, а потім все негативні.

Варіант 4.

Обчислити:

1. Суму елементів масиву з парними номерами.
2. Суму елементів масиву, розташованих між першим і останнім негативними елементами.
3. Стиснути масив, видаливши з нього елементи, модуль яких не перевищує 1. Елементи в кінці масиву, які звільнилися, заповнити нулями.

Варіант 5.

Обчислити:

1. Максимальний елемент масиву.
2. Суму елементів масиву, розташованих до останнього позитивного елемента.
3. Стиснути масив, видаливши з нього всі елементи, модуль яких знаходиться в інтервалі $[a, b]$. Вивільнені в кінці масиву елементи заповнити нулями.

Варіант 6.

Обчислити:

1. Найменший позитивний елемент масиву.
2. Суму елементів масиву, розташованих між першим і останнім позитивними елементами.
3. Перетворити масив таким чином, щоб спочатку розташовувалися всі елементи, рівні нулю, а потім всі інші.

Варіант 7.

Обчислити:

1. Номер максимального елемента масиву.
2. Добуток елементів масиву, розташованих між першим і останнім позитивними елементами.
3. Перетворити масив таким чином, щоб у першій його половині розташовувалися елементи, які стоять в непарних позиціях, а в другій половині - елементи, які стоять в парних позиціях.

Варіант 8.

Обчислити:

1. Номер мінімального елемента масиву.
2. Суму елементів масиву, розташованих між першим і другим негативними елементами.
3. Перетворити масив таким чином, щоб спочатку розташовувалися елементи, модуль яких не перевершує 1, а в другій половині - всі інші.

Варіант 9.

Обчислити:

1. Максимальний по модулю елемент масиву.
2. Суму елементів масиву, розташованих між першим і другим позитивними елементами.
3. Перетворити масив таким чином, щоб елементи, рівні нулю, розташовувалися після всіх інших.

Варіант 10.

Обчислити:

1. Найбільший негативний елемент масиву.
2. Суму елементів масиву, розташованих між першим елементом масиву та першим елементом масиву, який дорівнює нулю.

3. Перетворити масив таким чином, щоб у першій його половині розташовувалися елементи, які стоять в парних позиціях, а в другій половині - елементи, які стоять в непарних позиціях.

Варіант 11.

Обчислити:

1. Номер мінімального за модулем елемента масиву.
2. Суму елементів масиву, розташованих після першого негативного елемента.
3. Стиснути масив, видаливши з нього всі елементи, модуль яких знаходиться в інтервалі $[a, b]$. Вивільнені в кінці масиву елементи заповнити нулями.

Варіант 12.

Обчислити:

1. Номер максимального по модулю елемента масиву.
2. Суму елементів масиву, розташованих після першого позитивного елемента.
3. Перетворити масив таким чином, щоб спочатку розташовувалися всі елементи, ціла частина яких знаходиться в інтервалі $[a, b]$, а потім - всі інші.

Варіант 13.

Обчислити:

1. Кількість елементів масиву, що лежать в діапазоні від A до B .
2. Суму елементів масиву, розташованих після першого максимального елемента.
3. Упорядкувати елементи масиву за зменшенням модулів елементів.

Варіант 14.

Обчислити:

1. Кількість елементів масиву, рівних нулю.
2. Суму елементів масиву, розташованих після першого мінімального елемента.
3. Упорядкувати елементи масиву за зростанням модулів елементів.

Варіант 15.

Обчислити:

1. Кількість елементів масиву, більших заданого значення C .
2. Добуток елементів масиву, розташованих після останнього максимального за модулем елемента.
3. Перетворити масив таким чином, щоб спочатку розташовувалися всі негативні елементи, потім всі позитивні, після всіх інших - рівні нулю.

Варіант 16.

Обчислити:

1. Суму елементів масиву з номерами, кратними трьом.
2. Суму елементів масиву, розташованих між першим і останнім негативними елементами.
3. Стиснути масив, видаливши з нього елементи, модуль яких не перевищує середнє арифметичне елементів масиву. Вивільнені в кінці масиву елементи заповнити нулями.

Варіант 17.

Обчислити:

1. Максимальний елемент масиву.
2. Суму елементів масиву, розташованих до останнього позитивного елемента.
3. Стиснути масив, видаливши з нього всі елементи, модуль яких знаходиться в інтервалі $[a, b]$. Вивільнені в кінці масиву елементи заповнити нулями.

Варіант 18.

Обчислити:

1. Кількість елементів масиву, менших заданого значення C .
2. Суму елементів масиву, розташованих між першим і останнім позитивними елементами.
3. Перетворити масив таким чином, щоб спочатку розташовувалися всі елементи, рівні нулю, а потім всі інші.

Варіант 19.

Обчислити:

1. Номер максимального елемента масиву.
2. Добуток елементів масиву, розташованих між першим і останнім позитивними елементами.
3. Перетворити масив таким чином, щоб у першій його половині розташовувалися елементи, які стоять в непарних позиціях, а в другій половині - елементи, які стоять в парних позиціях.

Варіант 20.

Обчислити:

1. Суму елементів масиву з номерами, кратними п'яти.
2. Суму елементів масиву, розташованих між першим і другим негативними елементами.
3. Перетворити масив таким чином, щоб спочатку розташовувалися елементи, модуль яких не перевершує 1, а в другій половині - всі інші.

Варіант 21.

Обчислити:

1. Максимальний по модулю елемент масиву.
2. Суму елементів масиву, розташованих між першим і другим позитивними елементами.
3. Перетворити масив таким чином, щоб елементи, рівні нулю, розташовувалися після всіх інших.

Варіант 22.

Обчислити:

1. Мінімальний по модулю елемент масиву.
2. Суму елементів масиву, розташованих між першим елементом масиву, та першим елементом, який дорівнює нулю.

3. Перетворити масив таким чином, щоб у першій його половині розташовувалися елементи, які стоять в парних позиціях, а в другій половині - елементи, які стоять в непарних позиціях.

Варіант 23.

Обчислити:

1. Суму негативних елементів масиву.
2. Добуток елементів масиву, розташованих між максимальним і мінімальним елементами.
3. Упорядкувати елементи масиву за зростанням.

Варіант 24.

Обчислити:

1. Суму позитивних елементів масиву.
2. Добуток елементів масиву, розташованих між максимальним за модулем і мінімальним по модулю елементами.
3. Упорядкувати елементи масиву за спаданням.

Варіант 25.

Обчислити:

1. Добуток елементів масиву з парними номерами.
2. Суму елементів масиву, розташованих між першим і останнім нульовими елементами.
3. Перетворити масив таким чином, щоб спочатку розташовувалися всі позитивні елементи і 0, а потім все негативні.

Варіант 26.

Обчислити:

1. Суму елементів масиву з парними номерами.
2. Суму елементів масиву, розташованих між першим і останнім негативними елементами.

3. Стиснути масив, видаливши з нього елементи, модуль яких не перевищує 1. Елементи в кінці масиву, які звільнилися, заповнити нулями.

Варіант 27.

Обчислити:

1. Максимальний елемент масиву.
2. Суму елементів масиву, розташованих до останнього позитивного елемента.
3. Стиснути масив, видаливши з нього всі елементи, модуль яких знаходиться в інтервалі $[a, b]$. Вивільнені в кінці масиву елементи заповнити нулями.

ЛАБОРАТОРНА РОБОТА № 27-28

Складання та відлагодження програми з множинним успадкуванням класів

Мета: навчитись складати та відлагоджувати програми з множинним успадкуванням класів.

Теоретична частина

Спадкування припускає, що у кожного похідного класу є базовий клас, від якого він успадковує дані і методи обробки цих даних. Похідний клас, в свою чергу, може бути базовим класом для інших класів програми. Такий набір класів називається ієрархією класів. Ієрархічна організація успадкування може мати багато рівнів. Похідний клас успадковує підоб'єкт та використовує відкриті і захищені дані базового класу, який розташований вище по ієрархії.

Постановка задачі

Спроекувати ієрархію класів згідно діаграми класів, наведеної на рисунку 13.1, ієрархію складають класи "Вагон", "Вантажний вагон", "Цистерна", "Платформа", "Пасажирський вагон". У всіх похідних класах існують дані, які специфікують об'єкти свого класу. Наприклад, у класу Платформа визначається член-дане "кількість бортів". Такого даного немає в жодному з наявних класів. Проаналізувати використання косвенного базового класу та дружнього класу. Пояснити, як успадковуються елементи базових класів.

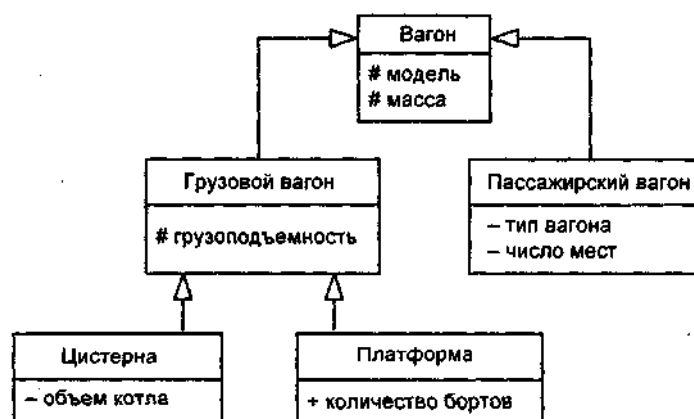


Рис. 13.1 – Діаграма класів

Хід роботи

На діаграмі базовими класами є два класи: "Вагон" та "Вантажний вагон". Клас "Вантажний вагон" одночасно і базовий (від нього успадковують класи "Цистерна" та "Платформа") і похідний (він успадковує властивості класу "Вагон"). Так, похідними від базового класу "Вагон" є класи "Вантажний вагон" і "Пасажирський вагон", які успадковують дані "модель" і "маса". Похідними від базового класу "Вантажний вагон" є класи "Цистерна" та "Платформа", наслідуючи від нього дані "модель", "маса" і "вантажопідйомність". У програмі обмежуються трьома рівнями ієрархії.

В консольному додатку створити ієрархію класів у відповідності зі схемою:

- CCar (Вагон) – базовий для всіх класів;
- CCargo(Вантажний вагон) – похідний від CCar (і базовий для CTank, CPlatform);
- CTank(Цистерна) – похідний від CCargo;
- CPlatform(Платформа) – похідний від CCargo;
- CCarriage(Пасажирський вагон) – похідний від CCar.

В специфікації всіх класів (у файли (.h)) включено також friend class CMenu; , який означає, що класи дозволяють об'єктам класу CMenu отримувати доступ до своїх закритих та захищених членів.

Текст програми наведений в лістингу 13.1.

Лістинг 13.1 – Текст програми.

```
#pragma once
#include <string>
#include <iostream>
using namespace std ;

// Car.h – Специфікація базового класу CCar (Вагон)
class CCar
{
protected:
    string m_model ; // модель
    double m_mass ; // маса
public:
    CCar ( ) ; CCar ( const string&, const double ) ;
    CCar ( const CCar& ) ; ~CCar ( ) ;
    void mod ( int ) ;
```

```

        friend class CMenu;                // CMenu -друг класу CCar
    };
// Реалізація конструкторів CCar
CCar :: CCar ( ) : m_model ( "" ), m_mass ( 0.0 ) { }
CCar :: CCar ( const string& md, const double m )
: m_model ( md ), m_mass ( m ) { }
CCar :: CCar ( const CCar& c )
: m_model ( c.m_model ), m_mass ( c.m_mass ) { }
// Car.cpp - Реалізація CCar
CCar :: ~CCar ( ) { }
void CCar :: mod ( int n )
{
    m_mass += n ;
}

// Cargo.h -Специфікація похідного класу CCargo (Вантажний вагон)
class CCargo : public CCar
{
protected:
    int m_carCapacity ;        // вантажопід'ємність
public:
    CCargo ( ) ; CCargo ( const string&, const double, const int ) ;
    CCargo ( const CCargo& ) ; ~CCargo ( ) ;
    void mod ( int ) ;
    friend class CMenu ;        // CMenu -друг класу CCargo
} ;
// Реалізація конструкторів CCargo
CCargo :: CCargo ( ) : CCar ( ), m_carCapacity ( 0 ) { }
CCargo :: CCargo ( const string& md, const double m, const int c )
: CCar ( md, m ), m_carCapacity ( c ) { }
CCargo :: CCargo ( const CCargo& c )
: CCar ( c ), m_carCapacity ( c.m_carCapacity ) { }
// Cargo.cpp - Реалізація CCargo
CCargo :: ~CCargo ( ) { }
void CCargo :: mod ( int n )
{
    CCar :: mod ( n ) ;        // виклик функції базового класу
    m_carCapacity += n ;
}

// Tank.h - Специфікація похідного класу CTank (Цистерна)
class CTank : public CCargo
{
    double m_volume ;        // об'єм котла
public:
    CTank ( ) ;
    CTank ( const string&, const double, const int, const double ) ;
    CTank ( const CTank& ) ; ~CTank ( ) ;
    void setVolume ( const double ) ; double getVolume ( ) const ;
    void mod ( int ) ;
    friend class CMenu ;        // CMenu -друг класу CTank
} ;
// Реалізація конструкторів CTank
CTank :: CTank ( ) : CCargo ( ), m_volume ( 0.0 ) { }
CTank :: CTank ( const string& md, const double m, const int c, const double v )
: CCargo ( md, m, c ), m_volume ( v ) { }
CTank :: CTank ( const CTank& c )
: CCargo ( c ), m_volume ( c.m_volume ) { }
// Tank.cpp - Реалізація CTank
CTank :: ~CTank ( ) { }
void CTank :: setVolume ( const double v ) { m_volume = v ; }
double CTank :: getVolume ( ) const { return m_volume ; }
void CTank :: mod ( int n )
{
    m_volume += n ;
}
// Platform.h - Специфікація похідного класу CPlatform (Платформа)
class CPlatform : public CCargo
{
public:

```

```

    int m_boards ;           // кількість бортів
    CPlatform ( ) ;
    CPlatform ( const string&, const double, const int, const int ) ;
    CPlatform ( const CPlatform& ) ; ~CPlatform ( ) ;
    void mod ( int ) ;
    friend class CMenu ;           // CMenu -друг класу CPlatform
} ;
// Реалізація конструкторів CPlatform
CPlatform :: CPlatform ( ) : CCargo ( ), m_boards ( 0 ) { }
CPlatform :: CPlatform ( const string& md, const double m, const int c, const int
b )
: CCargo ( md, m, c ), m_boards ( b ) { }
CPlatform :: CPlatform ( const CPlatform& c )
: CCargo ( c ), m_boards ( c.m_boards ) { }
// Platform.cpp - реалізація CPlatform
CPlatform :: ~CPlatform ( ) { }
void CPlatform :: mod ( int n )
{    m_boards -= n ;    }

// Специфікація похідного класу CCarriage (Пасажирський вагон)
class CCarriage : public CCar
{
    string m_type ;    // тип вагону
    int m_places ;           // кількість місць
public:
    CCarriage ( ) ;
    CCarriage ( const string&, const double, const string&,           const int
) ;
    CCarriage ( const CCarriage& ) ;    ~CCarriage ( ) ;
    void setType ( const string& ) ;    void setPlaces ( const int ) ;
    string getType ( ) const ;           int getPlaces ( ) const ;
    void mod ( int ) ;
    friend class CMenu ;           // CMenu -друг класу CCarriage
} ;
// Реалізація конструкторів CCarriage
CCarriage :: CCarriage ( ) : CCar ( ), m_type ( "" ), m_places ( 0 ) { }
CCarriage :: CCarriage ( const string& md, const double m,           const
string& t, const int p )
: CCar ( md, m ), m_type ( t ), m_places ( p ) { }
CCarriage :: CCarriage ( const CCarriage& c )
: CCar ( c ), m_type ( c.m_type ), m_places ( c.m_places ) { }
// Carriage.cpp - Реалізація CCarriage
CCarriage :: ~CCarriage ( ) { }
void CCarriage :: setType ( const string& t ) { m_type = t ; }
void CCarriage :: setPlaces ( const int p ) { m_places = p ; }
string CCarriage :: getType ( ) const { return m_type ; }
int CCarriage :: getPlaces ( ) const { return m_places ; }
void CCarriage :: mod ( int n )    {
    CCar :: mod ( n ) ;           // виклик функції базового класу
++m_places ;
}

// Menu.h - Специфікація CMenu
class CMenu
{
public:
    CCar* m_p ;           // вказівник на базовий тип
    CMenu ( ) { }    ~CMenu ( ) { }
    void showMenuAndSelect ( ) ;
private:
    void tank ( ) ; void platform ( ) ; void carriage ( ) ;
} ;
// Menu.cpp -Реалізація класу CMenu
// активізація меню
void CMenu :: showMenuAndSelect ( )
{

```

```

char choice ;
do
{
    cout << "\n1. Tank\t2. Platform\t3. Carriage\t0. Exit\n" ;
    cout << "Please, your choice -> " ; cin >> choice ;
    switch ( choice )
    {
        case '1': tank ( ) ; break ;
        case '2': platform ( ) ; break ;
        case '3': carriage ( ) ; break ;
    }
} while ( choice != '0' ) ;
}
// перевірка класу CTank
void CMenu :: tank ( )
{
    cout << "\nTank\n" ;
    CTank* p = new CTank ( "15-1547", 24.5, 68, 85 ) ;
    cout << "Model\t" << p -> m_model << endl ;
    cout << "Mass\t" << p -> m_mass << endl ;
    cout << "Carriage capacity\t" << p -> m_carCapacity << endl ;
    cout << "Volume of the boiler\t" << p -> m_volume << endl ;
    cout << "\nTANK UPDATING\n" ;
    m_p = p ;
    m_p -> mod ( 2 ) ; // виклик через вказівник на базовий тип
    cout << "Volume of the boiler\t" << p -> m_volume << endl ;
    p -> mod ( 2 ) ; // виклик через вказівник на похідний тип
    cout << "Volume of the boiler\t" << p -> m_volume << endl ;
    CTank o ( *p ) ; m_p = & o ;
    m_p -> mod ( 4 ) ; // виклик через вказівник на базовий тип
    cout << "Volume of the boiler\t" << o.m_volume << endl ;
    delete p ;
}
// перевірка класу CPlatform
void CMenu :: platform ( )
{
    cout << "\nPlatform\n" ;
    CPlatform* p = new CPlatform ( "13-4012", 21.4, 71, 4 ) ;
    cout << "Model\t" << p -> m_model << endl ;
    cout << "Mass\t" << p -> m_mass << endl ;
    cout << "Carriage capacity\t" << p -> m_carCapacity << endl ;
    cout << "Quantity of boards\t" << p -> m_boards << endl ;
    cout << "\nMODIFY PLATFORM\n" ;
    m_p = p ; m_p -> mod ( 2 ) ;
    cout << "Quantity of boards\t" << p -> m_boards << endl ;
    delete p ;
}
// перевірка класу CCarriage
void CMenu :: carriage ( )
{
    cout << "\nCarriage\n" ;
    CCarriage* p = new CCarriage ( "61-807", 55, "Compartment", 36 ) ;
    cout << "Model\t" << p -> m_model << endl ;
    cout << "Mass\t" << p -> m_mass << endl ;
    cout << "Type\t" << p -> m_type << endl ;
    cout << "Number of places\t" << p -> m_places << endl ;
    cout << "\nMODIFY CARRIAGE\n" ;
    m_p = p ; m_p -> mod ( 2 ) ;
    cout << "Mass\t" << p -> m_mass << endl ;
    cout << "Number of places\t" << p -> m_places << endl ;
    delete p ;
}

int _tmain(int argc, _TCHAR* argv[])
{

```

```
CMenu* p = new CMenu;  
p -> showMenuAndSelect ( ) ;  
delete p ;  
return 0;  
}
```

Виконати програму, проаналізувати результати. Оформити звіт, навести індивідуальний хід роботи, прокоментувати програму в індивідуальному порядку.

В звіті приділити увагу ієрархії класів, який клас є косвеним, що і як успадковано, де при виклику використана операція розширення області видимості.

ЛАБОРАТОРНА РОБОТА № 29-30

Складання програми з ромбовидною ієрархією класів. Відлагодження програми з конфліктами при успадкуванні

Мета: навчитись складати програми з ромбовидною ієрархією класів та з різними специфікаторами доступу при успадкуванні, відлагоджувати програми з конфліктами при успадкуванні

Теоретична частина

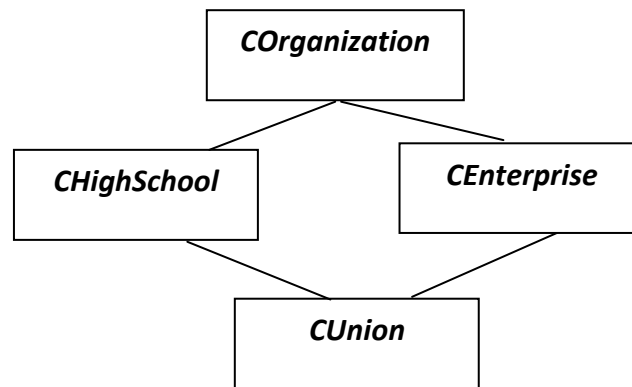
Проектування та створення програми з множинним успадкуванням класів та ромбовидною ієрархією.

Множинне успадкування означає, що клас має декілька базових класів. Якщо в базових класах є однойменні елементи, при цьому може відбутися конфлікт ідентифікаторів, який усувається за допомогою операції доступу до області видимості.

У випадку ромбовидної ієрархії конфлікт при успадкуванні полягає в тому, що нащадок отримує дві копії підоб'єкта косвеного базового класу, тому при зверненні до його елемента буде помилка, пов'язана з тим, що невідомо, до якої копії звертатися. Цей конфлікт усувається використанням віртуального успадкування, ключове слово `virtual` пишеться в оголошенні перед іменем базового класу, який для нащадка стане косвеним в ромбовидній ієрархії.

Частина I

Постановка задачі. Створити ієрархію класів у відповідності зі схемою на наведеними фрагментами програмного коду. Замінити в головній функції назву організації на власне прізвище. Пояснити в звіті своїми словами використання віртуального успадкування. Прокоментувати в програмному коді, де воно оголошується, прокоментувати, як успадковані окремі елементи косвеного базового класу та як до них звертатися з нащадка, які є конфлікти при успадкуванні і як вони вирішуються, в головній функції де з яких класів викликаються методи `CalcPayment`.



```

// Organization.h - Специфікація COrganization (Організація)
#include <string>
using namespace std ;
#pragma once
class COrganization
{
protected:
    double m_fund ;          // обсяг резервного фонду
public:
    string m_name ;          // назва організації
    COrganization ( ) ;
    COrganization ( const string&, const double ) ;
    ~COrganization ( ) ;
    void setFund ( double ) ; double getFund ( ) const ;
    // обчислює суму щомісячних відрахувань
    double calcPayment ( const int = 0 ) const ;
} ;

// Organization.cpp - Реалізація COrganization
#include "Organization.h"
COrganization :: COrganization ( ) : m_name ( "" ), m_fund ( 0.0 ) { }
COrganization :: COrganization ( const string& n, const double f )
: m_name ( n ), m_fund ( f ) { }
COrganization :: ~COrganization ( ) { }
void COrganization :: setFund ( double f ) { m_fund = f ; }
double COrganization :: getFund ( ) const { return m_fund ; }
double COrganization :: calcPayment ( const int p ) const
{ return ( m_fund * p / 1200 ) ; }

// HighSchool.h - Специфікація CHighSchool (Вуз)
#pragma once
#include "Organization.h"
class CHighSchool : virtual public COrganization
{
protected:
    int m_rate ;              // % відрахунків з резервного фонду
public:
    CHighSchool ( ) ;
    CHighSchool ( const string&, const double, const int ) ;
    ~CHighSchool ( ) ;
    void setRate ( int ) ; int getRate ( ) const ;
    double calcPayment ( const int = 0 ) const ;
} ;

```

```

// HighSchool.cpp - Реалізація CHighSchool
#include "HighSchool.h"
CHighSchool :: CHighSchool ( ) : COrganization ( ), m_rate ( 0 ) { }
CHighSchool ::
CHighSchool ( const string& n, const double f, const int r )
: COrganization ( n, f ), m_rate ( r ) { }
CHighSchool :: ~CHighSchool ( ) { }
void CHighSchool :: setRate ( int r ) { m_rate = r ; }
int CHighSchool :: getRate ( ) const { return m_rate ; }
double CHighSchool :: calcPayment ( const int p ) const
{
    if ( p > m_rate ) return ( m_fund * m_rate / 1200 ) ;
    else return ( COrganization :: calcPayment ( p ) ) ;
}
// Enterprise.h - Специфікація CEnterprise (Підприємство)
#pragma once
#include "Organization.h"
class CEnterprise : virtual public COrganization
{
protected:
    double m_alloc ;          // обсяг благодійного фонду
public:
    CEnterprise ( ) ;
    CEnterprise ( const string&, const double, const double ) ;
    ~CEnterprise ( ) ;
    void setAlloc ( const double ) ; double getAlloc ( ) const ;
    double calcPayment ( const int = 0 ) const ;
} ;
// Enterprise.cpp - Реалізація CEnterprise
#include "Enterprise.h"
CEnterprise :: CEnterprise ( ) : COrganization ( ), m_alloc ( 0.0 ) { }
CEnterprise ::
CEnterprise ( const string& n, const double f, const double a )
: COrganization ( n, f ), m_alloc ( a ) { }
CEnterprise :: ~CEnterprise ( ) { }
void CEnterprise :: setAlloc ( const double a )
{ m_alloc = a ; }
double CEnterprise :: getAlloc ( ) const { return m_alloc ; }
double CEnterprise :: calcPayment ( const int p ) const
{ return ( m_alloc * p / 1200 ) ; }
// Union.h - Специфікація CUnion (Науково-виробниче об'єднання)
#pragma once
#include <string>
using namespace std ;
#include "HighSchool.h"
#include "Enterprise.h"
class CUnion : public CHighSchool, public CEnterprise
{
public:
    string m_kind ;          // вид діяльності
    CUnion ( ) ;
    CUnion ( const string&, const double, const string&, const int,
const double ) ;
    ~CUnion ( ) ;
    double calcPayment ( const int = 0 ) const ;

```

```

} ;
// Union.cpp - Реалізація CUnion
#include "Union.h"
CUnion :: CUnion ( )
: m_kind ( "" ), CHighSchool ( ), CEnterprise ( ) { }
CUnion :: CUnion ( const string& n, const double f, const string& k,
↳const int r, const double a )
: m_kind ( k ), CHighSchool ( "", 0.0, r ), CEnterprise ( "", 0.0, a )
{ m_name = n ; setFund ( f ) ; }
CUnion :: ~CUnion ( ) { }
double CUnion :: calcPayment ( const int ) const { return 0.0 ; }

// main.cpp головна функція
#include <iostream>
using namespace std ;
#include "Union.h"
int main ( )
{
    CHighSchool* pH =
        new CHighSchool ( "Назва організації", 1.0e6, 2 ) ;
    cout << "\nName\t" << pH -> m_name << endl ;
    cout << "Fund\t" << pH -> getFund ( ) << endl ;
    cout << "Rate\t" << pH -> getRate ( ) << endl ;
    CEnterprise* pE =
        new CEnterprise ( "CLEVER OWL Ltd.", 1.0e8, 1.0e7 ) ;
    cout << "\nName\t" << pE -> m_name << endl ;
    cout << "Fund\t" << pE -> getFund ( ) << endl ;
    cout << "Allocation\t" << pE -> getAlloc ( ) << endl << endl ;
    CUnion* pU = new CUnion ( "VICTORY", 1.25e8, "Miracles",
↳pH -> getRate ( ), pE -> getAlloc ( ) ) ;
    cout << "\nName\t" << pU -> m_name << endl ;
    cout << "Kind\t" << pU -> m_kind << endl ;
    cout << "Fund\t" << pU -> getFund ( ) << endl ;
    cout << "Rate\t" << pU -> getRate ( ) << endl ;
    cout << "Allocation\t" << pU -> getAlloc ( ) << endl ;
    cout << "\nPayment\n" ;
    COrganization* p ;
    p = pH ;
    cout << pH-> m_name << "\t= " << p-> calcPayment ( 5 ) << endl ;
    p = pE ;
    cout << pE-> m_name << "\t= " << p-> calcPayment ( 17 ) << endl ;
    p = pU ;
    cout << pU-> m_name << "\t\t= " << p-> calcPayment ( ) << endl ;
}

```

ЛАБОРАТОРНА РОБОТА № 31-32

Складання та відлагодження програми з різними специфікаторами доступу при успадкуванні

Мета: навчитись складати та відлагоджувати програми з різними специфікаторами доступу при успадкуванні

Теоретична частина

При відкритому успадкуванні елементи базового класу успадковуються "як є", тобто, захищені як захищені і відкриті як відкриті. При цьому закриті елементи базового класу являються недоступними нащадку.

Захищене успадкування знижує доступ до успадкованих елементів базового класу до рівня захищеного.

Закрите успадкування знижує доступ до успадкованих елементів базового класу до рівня закритого.

Множинне успадкування означає, що клас має декілька базових класів. Якщо в базових класах є однойменні елементи, при цьому може відбутися конфлікт ідентифікаторів, який усувається за допомогою операції доступу до області видимості. У випадку ромбовидної ієрархії конфлікт при успадкуванні полягає в тому, що нащадок отримує дві копії підоб'єкта косвеного базового класу, тому при зверненні до його елемента буде помилка, пов'язана з тим, що невідомо, до якої копії звертатися. Цей конфлікт усувається використанням віртуального успадкування, ключове слово *virtual* пишеться в оголошенні перед іменем базового класу, який для нащадка стане косвеним в ромбовидній ієрархії.

Частини I-II

Постановка задачі. Створити ієрархію класів у відповідності зі схемою на наведених фрагментах програмного коду. Замінити успадкування від базового класу верхнього рівня на:

- непарні варіанти – *class CHighSchool : virtual protected COrganization;*
- парні варіанти – *class CEnterprise : virtual protected COrganization.*

Відлагодити програму, пояснити, що в ній змінилося.

```

// Organization.h - Специфікація COrganization (Організація)
#include <string>
using namespace std ;
#pragma once
class COrganization
{
protected:
    double m_fund ;          // обсяг резервного фонду
public:
    string m_name ;          // назва організації
    COrganization ( ) ;
    COrganization ( const string&, const double ) ;
    ~COrganization ( ) ;
    void setFund ( double ) ; double getFund ( ) const ;
    // обчислює суму щомісячних відрахувань
    double calcPayment ( const int = 0 ) const ;
} ;

// Organization.cpp - Реалізація COrganization
#include "Organization.h"
COrganization :: COrganization ( ) : m_name ( "" ), m_fund ( 0.0 ) { }
COrganization :: COrganization ( const string& n, const double f )
: m_name ( n ), m_fund ( f ) { }
COrganization :: ~COrganization ( ) { }
void COrganization :: setFund ( double f ) { m_fund = f ; }
double COrganization :: getFund ( ) const { return m_fund ; }
double COrganization :: calcPayment ( const int p ) const
{
    return ( m_fund * p / 1200 ) ; }

// HighSchool.h - Специфікація CHighSchool (Вуз)
#pragma once
#include "Organization.h"
class CHighSchool : virtual public COrganization
{
protected:
    int m_rate ;              // % відрахунків з резервного фонду
public:
    CHighSchool ( ) ;
    CHighSchool ( const string&, const double, const int ) ;
    ~CHighSchool ( ) ;
    void setRate ( int ) ; int getRate ( ) const ;
    double calcPayment ( const int = 0 ) const ;
} ;

// HighSchool.cpp - Реалізація CHighSchool
#include "HighSchool.h"
CHighSchool :: CHighSchool ( ) : COrganization ( ), m_rate ( 0 ) { }
CHighSchool ::
CHighSchool ( const string& n, const double f, const int r )
: COrganization ( n, f ), m_rate ( r ) { }
CHighSchool :: ~CHighSchool ( ) { }
void CHighSchool :: setRate ( int r ) { m_rate = r ; }
int CHighSchool :: getRate ( ) const { return m_rate ; }
double CHighSchool :: calcPayment ( const int p ) const
{
    if ( p > m_rate ) return ( m_fund * m_rate / 1200 ) ;
    else return ( COrganization :: calcPayment ( p ) ) ;
}

```

```

// Enterprise.h - Специфікація CEnterprise (Підприємство)
#pragma once
#include "Organization.h"
class CEnterprise : virtual public COrganization
{
protected:
    double m_alloc ;           // обсяг благодійного фонду
public:
    CEnterprise ( ) ;
    CEnterprise ( const string&, const double, const double ) ;
    ~CEnterprise ( ) ;
    void setAlloc ( const double ) ; double getAlloc ( ) const ;
    double calcPayment ( const int = 0 ) const ;
} ;
// Enterprise.cpp - Реалізація CEnterprise
#include "Enterprise.h"
CEnterprise :: CEnterprise ( ) : COrganization ( ), m_alloc ( 0.0 ) { }
CEnterprise ::
CEnterprise ( const string& n, const double f, const double a )
: COrganization ( n, f ), m_alloc ( a ) { }
CEnterprise :: ~CEnterprise ( ) { }
void CEnterprise :: setAlloc ( const double a )
{    m_alloc = a ;    }
double CEnterprise :: getAlloc ( ) const { return m_alloc ; }
double CEnterprise :: calcPayment ( const int p ) const
{    return ( m_alloc * p / 1200 ) ; }
// Union.h - Специфікація CUnion (Науково-виробниче об'єднання)
#pragma once
#include <string>
using namespace std ;
#include "HighSchool.h"
#include "Enterprise.h"
class CUnion : public CHighSchool, public CEnterprise
{
public:
    string m_kind ;           // вид діяльності
    CUnion ( ) ;
    CUnion ( const string&, const double, const string&, const int,
    ↵const double ) ;
    ~CUnion ( ) ;
    double calcPayment ( const int = 0 ) const ;
} ;
// Union.cpp - Реалізація CUnion
#include "Union.h"
CUnion :: CUnion ( )
: m_kind ( "" ), CHighSchool ( ), CEnterprise ( ) { }
CUnion :: CUnion ( const string& n, const double f, const string& k,
    ↵const int r, const double a )
: m_kind ( k ), CHighSchool ( "", 0.0, r ), CEnterprise ( "", 0.0, a )
{    m_name = n ; setFund ( f ) ;    }
CUnion :: ~CUnion ( ) { }
double CUnion :: calcPayment ( const int ) const { return 0.0 ; }

// main.cpp головна функція

```

```

#include <iostream>
using namespace std ;
#include "Union.h"
int main ( )
{
    CHighSchool* pH =
        new CHighSchool ( "Назва організації", 1.0e6, 2 ) ;
    cout << "\nName\t" << pH -> m_name << endl ;
    cout << "Fund\t" << pH -> getFund ( ) << endl ;
    cout << "Rate\t" << pH -> getRate ( ) << endl ;
    CEnterprise* pE =
        new CEnterprise ( "CLEVER OWL Ltd.", 1.0e8, 1.0e7 ) ;
    cout << "\nName\t" << pE -> m_name << endl ;
    cout << "Fund\t" << pE -> getFund ( ) << endl ;
    cout << "Allocation\t" << pE -> getAlloc ( ) << endl << endl ;
    CUnion* pU = new CUnion ( "VICTORY", 1.25e8, "Miracles",
        pH -> getRate ( ), pE -> getAlloc ( ) ) ;
    cout << "\nName\t" << pU -> m_name << endl ;
    cout << "Kind\t" << pU -> m_kind << endl ;
    cout << "Fund\t" << pU -> getFund ( ) << endl ;
    cout << "Rate\t" << pU -> getRate ( ) << endl ;
    cout << "Allocation\t" << pU -> getAlloc ( ) << endl ;
    cout << "\nPayment\n" ;
    COrganization* p ;
    p = pH ;
    cout << pH-> m_name << "\t= " << p-> calcPayment ( 5 ) << endl ;
    p = pE ;
    cout << pE-> m_name << "\t= " << p-> calcPayment ( 17 ) << endl ;
    p = pU ;
    cout << pU-> m_name << "\t\t= " << p-> calcPayment ( ) << endl ;
}

```

ЛАБОРАТОРНА РОБОТА № 33-34

Складання програми з використанням композиції класів. Відлагодження програми з агрегацією класів

Мета: навчитись складати та відлагоджувати програми з використанням композиції або агрегації класів

Агрегація — можливість включення в один клас в якості членів-даних об'єктів іншого класу (відношення "містить" або "частина-ціле").

Композиція — вид відношення "частина-ціле", при якому частина не може існувати без цілого. Об'єкти-частини, оголошені за значенням, створюються в тому порядку, в якому вони оголошені у класі цілого, який їх включає, і до того, як буде створений об'єкт цілого. Для ініціалізації об'єкта частини у списку ініціалізації конструкторів класу цілого можна явно викликати конструктори класів частин.

Для включення об'єкта частини `m_part` класу `CPart` в клас `CWhole` є альтернативи:

- явно оголосити дане - `CPart m_part` ;
- оголосити покажчик або посилання на тип частини - `CPart * m_p` або `CPart &m_r` .

Частина I

Постановка задачі

Створити клас `CTrain`, який описує рейс. До класу `CTrain` включити клас `CStation`, який описує станцію відправлення. В загальному випадку станція, крім назви, може містити, наприклад, координати на карті, а рейс — перелік станцій слідування.

Порядок виконання роботи

1 Створити клас `CTrain`, який описує рейс і до складу якого включено об'єкт класу `CStation`, який описує станцію відправлення.

3. Окремо створити клас `CStation`, який описує станцію відправлення.

Специфікації та реалізації класів приведені у лістингу.

```

// Train.h специфікація класу CTrain
class CTrain                                // рейс
{
public:
    int m_num ;                            // номер поїзда
    CStation m_sd ; // станція відправлення (об'єкт-частина)
    CTrain ( ) ; CTrain ( int, char* ) ; CTrain ( CTrain& ) ;
    ~CTrain ( ) ;
}

// Train.cpp реалізація класу CTrain
#include "Train.h"
CTrain :: CTrain ( ) : m_num ( 0 ) { }
CTrain :: CTrain ( int num, char* p ) : m_num ( num ), m_sd ( p ) { }
CTrain :: CTrain ( CTrain& t )    m_num ( t.m_num ), m_sd ( t.m_sd ) { }
CTrain :: ~CTrain ( ) { }

// Station.h специфікація класу CStation
#pragma once
class CStation                              // опис станції
{
public:
    char* m_pName ; // назва станції
    CStation ( ) ; CStation ( char* ) ; CStation ( CStation& ) ;
    ~CStation ( ) ;
}

// Station.cpp реалізація класу CStation
CStation :: CStation ( ) : m_pName ( "" ) { }
CStation :: CStation ( char* pName ) : m_pName ( pName ) { }
CStation :: ~CStation ( ) { }

```

До головної функції проекту внести програмний код:

```

// main.cpp
void view ( CTrain* ) ;    // перегляд
int main ( )
{
    CTrain t ;
    CTrain* p = new CTrain ( 2, "Прізвище студента" ) ;

```

```

CTrain copy ( *p ) ;
delete p ;
}

```

1. Підставити власне прізвище в головній функції при створенні об'єкта. Воно має бути і в скріншоті виконання. Дописати самостійно реалізацію функції view, яка б виводила на екран номер поїзда та назву станції.
2. Дописати самостійно в головній функції виклик функції view окремо для кожного створеного об'єкту класу. Пояснити, які саме об'єкти класу CTrain створюються в програмі.
3. Дописати конструктор копіювання для класу CStation.
4. Пояснити, який саме конструктор CStation спрацьовує. Виправити помилку, пов'язану з тим, що два вказівника мають одну і ту ж адресу.
5. Відлагодити програму, отримати результати та прокоментувати їх.
6. Проаналізувати, яким способом оголошено об'єкт-частку, який входить до складу цілого об'єкту і як це повинно бути реалізовано з погляду на предметну область. Вказати, що це, агрегація чи композиція і чому.
7. Оформити звіт. Зверніть увагу, що звіти мають бути оформлені індивідуально, відповіді на запитання описані своїми словами. Копіювання знижує оцінку.

Частина II

Постановка задачі

Доповнити програму, виконану за завданням частини I, наведеним нижче. Програмно передбачити реалізацію відношення агрегації класів (або структур-класів), використовуючи вказівник, посилання. Пояснити

1 Оголосити окремо структуру «Час», що описує час характеристиками: години, хвилини, часовий пояс. Додати до класу CTrain закрите поле «Час відправлення». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

2 Оголосити окремо структуру «День тижня», що описує день тижня у вигляді порядкового номеру та назви. Додати до класу CTrain закрите поле «День тижня відправлення». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

3 Оголосити окремо структуру «Платформа, колія», що описує місце відправлення у вигляді характеристик: номер платформи, номер колії. Додати до класу CTrain закрите поле «Платформа, колія відправлення». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

4 Оголосити окремо структуру «Координати», що описує координати міста на мапі. Додати до класу CStation закрите поле «Координати станції». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

5 Спроекувати CTrain таким чином, щоб він містив інформацію як про станцію відправлення, так і про станцію призначення. Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

6 Спроекувати CTrain таким чином, щоб він містив інформацію про всі станції зупинок для даного рейсу (не більше десяти). Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

7 Оголосити окремо клас «Час», що описує час у вигляді годин та хвилин. Додати до класу CTrain закрите поле «Час відправлення». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

8 Оголосити окремо клас «День тижня», що описує день тижня у вигляді порядкового номеру та назви. Додати до класу CTrain закрите поле «Дні тижня відправлення». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

9 Оголосити окремо клас «Платформа, колія», що описує місце відправлення у вигляді номеру платформи та колії. Додати до класу CTrain закрите поле «Платформа, колія відправлення». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

10 Оголосити окремо клас «Координати», що описує координати міста на мапі. Додати до класу CStation закрите поле «Координати станції». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

11 Оголосити окремо структуру «Час», що описує час у вигляді годин та хвилин. Додати до класу CTrain інформацію про час відправлення та час прибуття. Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

12 Оголосити окремо клас «Час», що описує час у вигляді годин та хвилин. Спроекувати CTrain таким чином, щоб він містив інформацію про час (годин, хвилин) всіх зупинок для даного рейсу (не більше десяти). Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

13 Оголосити окремо структуру «Час», що описує час у вигляді годин та хвилин. Додати до класу CTrain закрите поле «Час прибуття». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

14 Оголосити окремо структуру «День тижня», що описує день тижня у вигляді назви скороченої та повної. Додати до класу CTrain закрите поле «Дні тижня відправлення (не більше семи)». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

15 Оголосити окремо структуру «Платформа, колія», що описує місце відправлення у вигляді номеру платформи та назви колії. Додати до класу CTrain закрите поле «Платформа, колія відправлення». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

16 Оголосити окремо структуру «Координати», що описує координати міста на мапі. Додати до класу CStation закрите поле «Координати станції». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

17 Спроекувати CTrain таким чином, щоб він містив інформацію як про станцію відправлення, так і про станцію призначення. Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

18 Спроекувати CTrain таким чином, щоб він містив інформацію про всі станції зупинок для даного рейсу (не більше десяти). Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

19 Оголосити окремо клас «Час», що описує час у вигляді годин та хвилин. Додати до класу CTrain закрите поле «Час відправлення». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

20 Оголосити окремо клас «День тижня», що описує день тижня у вигляді порядкового номеру та назви. Додати до класу CTrain закрите поле «Дні тижня відправлення» (не більше семи). Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

21 Оголосити окремо клас «Платформа, колія», що описує місце відправлення у вигляді номеру платформи та колії. Додати до класу CTrain закрите поле «Платформа, колія відправлення». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

22 Оголосити окремо клас «Координати», що описує координати міста на мапі. Додати до класу CStation закрите поле «Координати станції». Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

23 Оголосити окремо структуру «Час», що описує час у вигляді годин та хвилин. Додати до класу CTrain інформацію про час відправлення та час прибуття. Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

24 Оголосити окремо клас «Час», що описує час у вигляді годин та хвилин. Спроекувати CTrain таким чином, щоб він містив інформацію про час (годин, хвилин) всіх зупинок для даного рейсу (не більше десяти). Передбачити ініціалізацію, методи встановлення та отримання значення, необхідні перевірки.

Відлагодити, прокоментувати, оформити звіт.

ЛАБОРАТОРНА РОБОТА № 35-36

Створення та відлагодження програми з агрегацією або композицією класів

Мета: навчитись створювати та відлагоджувати програми з агрегацією або композицією класів

Частина I

Постановка задачі

Спроектувати класи, пов'язані відношенням агрегації або композиції, скласти програму за варіантами, наведеними нижче.

Варіант 1 Напишіть клас `Figura`, який описує тип шахової фігури, її колір і початкове положення фігури на шаховій дошці (вертикалі від А до Н і горизонталі від 1 до 8). Напишіть клас `Chess`, який описує шахову дошку, яка може містити шахові фігури у квадратах (одночасно дві фігури не можуть знаходитись в одному квадраті). Передбачте початкові позиції для трьох шахових фігур і можливість вивести на екран вміст дошки.

Доповніть можливістю ходити вказаною фігурою на вказаний квадрат (змінити положення) і можливістю вбити фігуру супротивника (вилучити з шахової дошки, вказати координати 0), якщо вона є у вказаному квадраті (дотримання правил, як ходять фігури, реалізовувати не треба).

Варіант 2 Напишіть клас `Complex`, який описує дійсну та уявну частини комплексного числа. Напишіть клас `Container`, який описує стек комплексних чисел. Стек може містити довільну кількість елементів (комплексних чисел). Спочатку стек порожній, за замовчуванням елемент додається в кінець. Напишіть функціонал додавання елементу в стек, виштовхування елементу і виведення вмісту стеку.

Доповніть перевітками, щоб не можна було виштовхнути елемент з порожнього стеку тощо і можливістю вставити елемент не в кінець стека, а в певну позицію між іншими елементами.

Варіант 3 Напишіть клас `QuadrEq`, який описує члени a (квадратичний), b (лінійний), c (вільний) квадратного рівняння ($a*x^2+b*x+c$). Напишіть клас `Solution`, який реалізує функціонал по роботі з квадратним рівнянням, а саме знаходження його коренів через дискримінант.

За замовчуванням члени рівняння дорівнюють 0. Користувач має можливість вводити, розраховувати і отримувати результат в зручному вигляді. Доповніть перевірки і зручним для користувача виведенням на екран.

Варіант 4 Навчальна група має назву і містить інформацію про студентів в групі. Спроектуйте класи `Group`, `Student`. Дані про студента: ПІБ, дата народження, стать, номер телефону. Додайте в клас `Group` функціонал з можливістю зарахування студента, відрахування, переведення з однієї групи в іншу.

Допишіть можливість редагування інформації про студента з необхідними перевітками.

Варіант 5 Про співробітника на фірмі ведеться інформація: ПІБ, заробітна плата за останній період. Спроектувати клас «Фірма», який містить назву фірми та перелік співробітників (наприклад, до 10). Передбачити метод розрахунку середньої заробітної плати працівників на фірмі.

Додати можливість взяти на роботу нового співробітника, можливість звільнення. Класи мають містити необхідні перевірки і методи встановлення значень полів.

Варіант 6 Про співробітника на фірмі ведеться інформація: ПІБ, посада. Спроектувати клас «Фірма», який містить назву фірми та перелік співробітників. Передбачити метод виведення списку на екран.

Додати можливість взяти на роботу нового співробітника, можливість звільнення. Класи мають містити необхідні перевірки і методи встановлення значень полів.

Варіант 7 В бібліотеці ведуть облік книг за каталогами. Інформація про книгу містить назву, автора, рік видання, видавництво. Каталог містить назву тематичного напрямку і інформацію з переліком книг в каталозі.

Реалізувати можливість додавання книги в каталог, вилучення книги з каталогу, підрахунку кількості книг в каталозі. Класи мають містити необхідні перевірки і методи встановлення значень полів.

Варіант 8 В магазині ведеться інформація щодо груп товарів та інформації про товар. Товар характеризується назвою, ціною. Група товару містить назву, перелік товарів.

Реалізувати можливість додавання нового товару в групу, вилучення товару з групи, підрахунку кількості товарів всього. Класи мають містити необхідні перевірки і методи встановлення значень полів.

Варіант 9 В кабінети коледж закупив меблі. Меблі характеризуються видом (стіл письмовий, шафа тощо) та мають інвентарний номер. Кабінет має номер і містить меблі. Реалізувати функціонал додавання, видалення, редагування інформації про кабінети та меблі.

Реалізувати можливість прийняття меблів на баланс в кабінет, переміщення одиниці меблів в інший кабінет, видачі списку меблів в кабінеті

Варіант 10 Студенти мають можливість записуватися на курси з видачею сертифікатів. Дані про студента містять ПІБ, стать, дату народження. Дані про курс містять назву, перелік студентів, записаних на курс. Реалізувати введення інформації, додавання студента на курс.

Видавати список студентів, що слухають курс.

Варіант 11 Дані про ліки містять вид (антибіотик тощо), назву, ціну. Аптечний склад характеризується номером і містить інформацію про ліки в наявності.

Реалізувати введення даних, можливість додавання ліків на склад, переведення на інший склад. Видавати перелік ліків на кожному складі.

Варіант 12 Комплексний обід складається з першого блюда, другого і напою за бажанням відвідувача. Дані про блюдо містять вид (перше, друге, напій), назву, ціну. Дані про обід містять перелік блюд. Реалізувати введення інформації про блюда в замовлення обіду.

Реалізувати можливість вибору напою за бажанням. Видавати список в замовленні.

Варіант 13 Напишіть клас Figure («Морський бій»), який описує однопалубні кораблі, і задає початкове положення фігури на полі гри (вертикалі від А до К і горизонталі від 1 до 10). Напишіть клас SeaWar, який описує поле гри, яка містить кораблі у певних координатах (одночасно дві фігури не можуть знаходитись в одному квадраті). Передбачити можливість вивести на екран координати кораблів.

Варіант 14 Напишіть клас Complex, який описує дійсну та уявну частини комплексного числа. Напишіть клас Container, який описує стек комплексних чисел. Стек може містити кількість елементів (комплексних чисел) не більше 15. Спочатку стек порожній, за замовчуванням елемент додається в кінець. Напишіть функціонал додавання елементу в стек, виштовхування елементу і друку вмісту стеку.

Доповніть перевітками, щоб не можна було виштовхнути елемент з порожнього стеку тощо і можливістю вставити елемент не в кінець стека, а в певну позицію між іншими елементами.

15 Напишіть клас QuadrEq, який описує члени a (квадратичний), b (лінійний), c (вільний) квадратного рівняння ($a \cdot x^2 + b \cdot x + c$). Напишіть клас Solution, який реалізує функціонал по роботі з квадратним рівнянням, а саме знаходження його коренів через дискримінанту. За замовчуванням члени рівняння дорівнюють 0. Користувач має можливість вводити, розраховувати і отримувати результат в зручному вигляді.

Доповніть перевітками і зручним для користувача виведенням на екран.

16 Навчальна група має назву і містить інформацію про студентів в групі. Спроектуйте класи Group, Student. Дані про студента: ПІБ, стать, номер залікової книжки. Додайте в клас Group функціонал з можливістю зарахування студента, відрахування, переведення з однієї групи в іншу.

Допишіть можливість редагування інформації про студента з необхідними перевірками, виведення даних про студентів групи.

17 Про співробітника на фірмі ведеться інформація: ПІБ, заробітна плата за місяць. Спроектувати клас «Фірма», який містить назву фірми та перелік співробітників. Передбачити метод розрахунку середньої заробітної плати працівників на фірмі.

Додати можливість взяти на роботу нового співробітника, можливість звільнення. Класи мають містити всі необхідні перевірки і методи встановлення значень полів.

18 Про співробітника ведеться інформація: ПІБ, посада. Спроектувати клас «Фірма», який містить назву фірми та перелік співробітників. Передбачити метод виведення списку на екран.

Додати можливість взяти на роботу нового співробітника, можливість звільнення. Класи мають містити необхідні перевірки і методи встановлення значень полів.

19 В бібліотеці ведуть облік книг за каталогами. Інформація про книгу містить назву, автора, рік видання, видавництво. Каталог містить назву тематичного напрямку і інформацію з переліком книг в каталозі.

Реалізувати можливість додавання книги в каталог, перенесення книги з каталогу в каталог, підрахунку кількості книг в каталозі. Класи мають містити необхідні перевірки і методи встановлення значень полів.

20 В магазині ведеться інформація щодо груп товарів та інформації про вид товару. Вид товару характеризується назвою, кількістю одиниць товарів в наявності. Група товару містить назву, перелік видів товарів.

Реалізувати можливість додавання нового виду товару в групу, перенесення виду товару з групи в групу, підрахунку кількості товарів всього. Класи мають містити необхідні перевірки і методи встановлення значень полів.

21 До відділення коледжу відноситься ряд кабінетів, в які коледж закупив меблі. Меблі характеризуються назвою та мають інвентарний номер. Кабінет має номер і містить меблі. Реалізувати введення даних, необхідні перевірки.

Реалізувати можливість прийняття меблів на баланс в кабінет, переміщення одиниці меблів в інший кабінет), видачі списку меблів в кабінеті.

22 Студенти мають можливість записуватися на курси з видачею сертифікатів. Дані про студента містять ПІБ, стать, дату народження. Дані про курс містять назву, перелік студентів, записаних на курс. Реалізувати додавання студенту на курс.

Реалізувати можливість одночасного проходження декількох курсів студентом. Видавати список студентів, що слухають вказаний курс.

23 Дані про ліки містять вид (антибіотик тощо), назву, ціну. Склад характеризується номером і містить інформацію про ліки в наявності. Реалізувати введення даних, необхідні перевірки.

Реалізувати можливість додавання ліків на склад, видалення певних ліків зі складу. Видавати перелік ліків на кожному складі.

24 Комплексний обід складається з першого блюда, другого і десерту за бажанням відвідувача. Дані про блюдо містять вид (перше, друге, десерт), назву, ціну. Дані про замовлення обіду містять перелік блюд. Реалізувати введення інформації, замовлення обіду. Реалізувати можливість вибору десерту за бажанням. Видавати інформацію про перелік блюд в замовленні.

Частина II

Постановка завдання

Спроектувати класи, пов'язані відношенням агрегації або композиції, скласти програму за варіантами, наведеними нижче.

Описати клас `CComplex` для представлення комплексних чисел у вигляді пари (a, b) (в математичному вигляді $a+bi$). Описати клас `CCalc` для реалізації операцій над комплексними числами. Клас `CCalc` повинен включати в якості членів-даних об'єкти класу `CComplex`, оголошені через вказівник, та забезпечувати універсальний механізм виконання операцій згідно варіанту з виведенням на екран формули та результату у вигляді $a+bi$. Рекомендовано використати перевантаження операції. Написати програму, що демонструє роботу класів. Пояснити в звіті, де використана агрегація, де композиція.

Порядок виконання роботи

1 Створити в консольному проєкті класи `CComplex`, `CCalc`, передбачити необхідний та достатній для універсальної роботи класів функціонал за варіантом (конструктори, сетери, перевірки та інше) та можливість введення даних користувачем.

2 Оформити звіт, пояснити в звіті використаний вид відношення "містить", а також його вигляд на діаграмі класів.

Індивідуальні завдання

Варіант 1. Розрахувати $(c,d)-(e,f)$.

Варіант 2. Розрахувати $(c,d)/(e,f)$.

Варіант 3. Розрахувати $1/(c,d)$.

Варіант 4. Розрахувати $(c,d)-(e,f)+(g,h)$.

Варіант 5. Розрахувати $(c,d)+(e,f)$.

Варіант 6. Розрахувати $(c,d)*(e,f)$.

Варіант 7. Розрахувати $(c,d)/(e,f)/(g,h)$.

Варіант 8. Розрахувати $(c,d)-(e,f)+(g,h)$.

Варіант 9. Розрахувати $(c,d)*(e,f)-(g,h)$.

Варіант 10. Розрахувати $((c,d)-(e,f))*(g,h)$.

Варіант 11. Розрахувати $((c,d)-(e,f))/(g,h)$.

Варіант 12. Розрахувати $(c,d)/(e,f)-(g,h)$.

Варіант 13. Розрахувати $1/(g,h)$.

Варіант 14. Розрахувати $(c,d)+(e,f)+(g,h)$.

Варіант 15. Розрахувати $((c,d)-(e,f))/(g,h)$.

Варіант 16. Розрахувати $(c,d)*(e,f)*(g,h)$.

Варіант 17. Розрахувати $(c,d)*(e,f)/(g,h)$.

Варіант 18. Розрахувати $(c,d)-(e,f)-(g,h)$.

Варіант 19. Розрахувати $(c,d)*(e,f)-(g,h)$.

Варіант 20. Розрахувати $1/(e,f)$.

Варіант 21. Розрахувати $(c,d)/(e,f)+(g,h)$.

Варіант 22. Розрахувати $(c,d)*(e,f)-(g,h)$.

Варіант 23. Розрахувати $(c,d)*(e,f)+(g,h)$.

Варіант 24. Розрахувати $(c,d)/(e,f)*(g,h)$.

Формули для довідки.

- Додавання

$$(a + bi) + (c + di) = (a + c) + (b + d)i.$$

- Віднімання

$$(a + bi) - (c + di) = (a - c) + (b - d)i.$$

- Множення

$$(a + bi) \cdot (c + di) = ac + bci + adi + bdi^2 = (ac - bd) + (bc + ad)i.$$

- Ділення

$$\frac{a + bi}{c + di} = \frac{(a + bi)(c - di)}{(c + di)(c - di)} = \frac{ac + bd}{c^2 + d^2} + \left(\frac{bc - ad}{c^2 + d^2} \right) i.$$

- Зокрема,

$$\frac{1}{a+bi} = \frac{a}{a^2+b^2} - \left(\frac{b}{a^2+b^2}\right)i.$$

ЛАБОРАТОРНА РОБОТА № 37-38

Створення та відлагодження одновіконного візуального додатку

Мета: навчитись створювати та відлагоджувати одновіконні візуальні додатки, закріпити навички з використання об'єктів класів

Постановка задачі.

Рекомендовано створити одновіконний візуальний додаток VCL Forms Application в середовищі Embarcadero C++ Builder (допускається за бажанням студента в Qt), в якому реалізовано завдання за варіантом.

Частина I

Частина 1. Ознайомитись з середовищем, спроектувати візуальну форму за варіантом. Оформити звіт, в якому описати головні вікна середовища, процес створення проекту, файли модулів, навести вигляд візуальної форми, підписати використані компоненти. Описати для використаних компонентів задані під час проектування властивості, обробники подій та інше. Навести основні фрагменти програмного коду.

Порядок виконання роботи.

1 В Embarcadero C++ Builder створити новий проект за допомогою меню File -> New, вибрати VCL Forms Application (або Windows VCL Application - C++ Builder).

2 Для виконання завдання використати різні види компонентів за сенсом: написи, редактори, списки, перемикачі, кнопки. За допомогою Палітри компонентів розмістити на візуальній формі необхідні компоненти, наприклад:

– TLabel — надпис;

– TEdit — однорядковий текстовий редактор для введення та редагування тексту під час виконання додатку;

– TMemo — багаторядковий текстовий редактор;

- TButton — кнопка для завдання дій керування додатком, таких як "Розрахувати", "Очистити" та ін.;
- TListBox — список, для вибору елементів зі списку, який можна задавати під час проектування або змінювати програмно;
- TCheckBox — прапорець, використовується як незалежний перемикач (за бажанням);
- TComboBox — випадаючий список — комбінація поля та списку, що дозволяє як вибирати елементи зі списку, який можна змінювати програмно, так і вводити текст у відповідному полі (за бажанням);
- TRadioGroup — група залежних перемикачів (за бажанням).

Щоб розмістити компонент на формі, потрібно клацнути по компоненту на Палітрі компонентів, а потім — в тому місці форми, де треба розмістити компонент.

З використанням Інспектора об'єктів задати значення властивостям візуальних об'єктів. Наприклад, для напису це властивість Caption="текст".

Частина II

Розробити програмний код за варіантом. Навести скріншот виконання програми з даними для перевірки виконання завдання.

Порядок виконання роботи.

Спроекувати та створити одновіконний візуальний додаток та власний клас згідно індивідуального завдання, для цього:

- проаналізувати постановку задачі за варіантом;
- сформулювати функції, які повинна виконувати програма, яка розробляється;
- виділити вхідну і вихідну інформацію, виявити типи даних та компоненти, які доцільно їм відповідають;
- спроекувати та розробити програму.

Введені значення рекомендовано хоча б в мінімальному обсязі перевіряти на допустимість.

Розробити програму згідно завдання варіанту, наведеного нижче.

Врахувати, що програма повинна допускати повторне введення значень та отримання результатів. Забезпечити, щоб на візуальній формі біло видно, які дані були введені, та який отриманий результат (наприклад, вносити в багаторядковий текстовий редактор).

Варіант 1. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок (вважаються прямокутними). Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	250
2	367
3	595

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити сумарний податок по ділянках кожного типу та кількість ділянок кожного типу.

Варіант 2. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	200
2	300
3	450

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити загальну площу в квадратних метрах, загальну площу в гектарах, загальну суму оподаткування.

Варіант 3. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
А	600
Б	500
В	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу площу.

Варіант 4. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
А	600
Б	500
В	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу суму оподаткування.

Варіант 5. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, табельний номер, кількість відпрацьованих годин, кількість утриманців.

Робочий час понад 40 годин у тиждень вважається понаднормовим і оплачується в розмірі у півтора рази більше. Погодинний тариф вважається однаковим для всіх робітників і складає 25 грн./год. Програма повинна розраховувати податок, стягнутий у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

Варіант 6. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Програма повинна розраховувати податок на соціальне забезпечення стягнутий у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

а також місцевий податок в розмірі 3%, але не більше 20 грн. За результатами розрахунків повинне бути виведене повідомлення, хто з працівників отримав максимальну зарплату.

Варіант 7. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин, погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 18%, але не більше 200 грн. за кожний тиждень.

Варіант 8. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: прізвище, кількість відпрацьованих годин, стаж роботи. Погодинний тариф вважається однаковим і складає 25 грн./год. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Оподаткування заробітної плати складає 15%, але не більше 100 грн. за кожний тиждень, а також повинен утримуватись місцевий податок, який складає 15%.

Варіант 9. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, що залежать від типу земельних ділянок. Установлені наступні податкові ставки залежно від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	250
2	367
3	595

Вхідні дані включають номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Необхідно обчислити сумарний податок по ділянках кожного типу, які включають: тип земельної ділянки, кількість ділянок, загальну площу у квадратних метрах, загальну площу в гектарах, загальну суму оподаткування.

Варіант 10. Постановка задачі

Розробити програму, що обчислює дохід по внеску. Програма повинна забезпечувати розрахунок простих і складних відсотків. Прості відсотки нараховуються наприкінці строку внеску, складні — щомісяця й додаються до початкової суми внеску й у наступному місяці відсотки нараховуються на нову суму. Прості відсотки нараховуються на внески з терміном зберігання до 1 року, складні — зі строком зберігання більше 1 року. Відсоткова ставка внеску залежить від суми внеску й задається в таблиці:

Сума внеску	Відсоткова ставка
До 1000 грн.	7%
1 – 10 тис. грн.	10%
10 тис. грн. і більше	15%

Користувач повинен мати можливість ввести суму, термін, обрати просту або складну схему нарахування відсотків і розрахувати значення суми, яку він отримає наприкінці терміну зберігання.

Варіант 11. Постановка задачі

Розробити програму, що дозволяє обчислювати вартість поїздки на автомобілях і повинна надавати користувачеві можливість вводити відстань до пункту призначення, вартість бензину, вибір варіанта поїздки: тільки в прямому напрямку або в прямому й у зворотному напрямках. Норма споживання бензину залежить від марки автомобіля й задається у вигляді таблиці.

Марка автомобіля	Норма споживання бензину (у літрах на 10 км)
Таврія	0.8
Нива	1.1
Славута	1.05

В результаті роботи програми повинна бути розрахована вартість поїздки.

Варіант 12. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: ім'я й прізвище, табельний номер, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Робочий час понад 40 годин на тиждень вважається понаднормовим і оплачується в півтора рази вище. Програма повинна розраховувати відрахування на соціальне забезпечення в розмірі 5%, але не більше 20 грн. за кожний тиждень, а також утримувати державний податок у розмірі 2% і місцевий податок, стягнений у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

Варіант 13. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин, погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж роботи	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 15%, але не більше 200 грн. за кожний тиждень.

Варіант 14. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок (вважаються прямокутними). Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	250
2	367
3	595

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити сумарний податок по ділянках кожного типу та кількість ділянок кожного типу.

Варіант 15. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	200
2	300
3	450

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити загальну площу в квадратних метрах, загальну площу в гектарах, загальну суму оподаткування.

Варіант 16. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
А	600
Б	500
В	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу площу.

Варіант 17. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
А	600
Б	500
В	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу суму оподаткування.

Варіант 18. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, табельний номер, кількість відпрацьованих годин, кількість утриманців.

Робочий час понад 40 годин у тиждень вважається понаднормовим і оплачується в розмірі у півтора рази більше. Погодинний тариф вважається однаковим для всіх робітників і складає 25 грн./год. Програма повинна розраховувати податок, стягнутий у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%

1 утриманець	10%
2 і більше	7%

Варіант 19. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Програма повинна розраховувати податок на соціальне забезпечення стягнутий у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

а також місцевий податок в розмірі 3%, але не більше 20 грн. За результатами розрахунків повинне бути виведене повідомлення, хто з працівників отримав максимальну зарплату.

Варіант 20. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин, погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 18%, але не більше 200 грн. за кожний тиждень.

Варіант 21. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: прізвище, кількість відпрацьованих годин, стаж роботи. Погодинний тариф вважається однаковим і складає 25 грн./год. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Оподаткування заробітної плати складає 15%, але не більше 100 грн. за кожний тиждень, а також повинен утримуватись місцевий податок, який складає 15%.

Варіант 22. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, що залежать від типу земельних ділянок. Установлені наступні податкові ставки залежно від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	250
2	367
3	595

Вхідні дані включають номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Необхідно обчислити сумарний податок по ділянках кожного типу, які включають: тип земельної ділянки, кількість ділянок, загальну площу у квадратних метрах, загальну площу в гектарах, загальну суму оподаткування.

Варіант 23. Постановка задачі

Розробити програму, що дозволяє вибрати зі списку дані про товари, які входять у покупку, обчислити вартість покупки. Знижка в 1% надається, якщо сума покупки більша 300 гривень, 2% — якщо сума покупки більше 500 гривень, 3% — якщо

сума покупки більше 1000 гривень. Інформація про вартість покупки й розмір знижки повинні виводитися на екран.

Варіант 24. Постановка задачі

Розробити програму, що обчислює дохід по внеску. Програма повинна забезпечувати розрахунок простих і складних відсотків. Прості відсотки нараховуються наприкінці строку внеску, складні — щомісяця й додаються до початкової суми внеску й у наступному місяці відсотки нараховуються на нову суму. Прості відсотки нараховуються на внески з терміном зберігання до 1 року, складні — зі строком зберігання більше 1 року. Відсоткова ставка внеску залежить від суми внеску й задається в таблиці:

Сума внеску	Відсоткова ставка
До 1000 грн.	7%
1 – 10 тис. грн.	10%
10 тис. грн. і більше	15%

Користувач повинен мати можливість ввести суму, термін, обрати просту або складну схему нарахування відсотків і розрахувати значення суми, яку він отримає наприкінці терміну зберігання.

Варіант 25. Постановка задачі

Розробити програму, що дозволяє обчислювати вартість поїздки на автомобілях і повинна надавати користувачеві можливість вводити відстань до пункту призначення, вартість бензину, вибір варіанта поїздки: тільки в прямому напрямку або в прямому й у зворотному напрямках. Норма споживання бензину залежить від марки автомобіля й задається у вигляді таблиці.

Марка автомобіля	Норма споживання бензину (у літрах на 10 км)
Таврія	0.8
Нива	1.1
Славута	1.05

В результаті роботи програми повинна бути розрахована вартість поїздки.

Варіант 26. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: ім'я й прізвище, табельний номер, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Робочий час понад 40 годин на тиждень вважається понаднормовим і оплачується в півтора рази вище. Програма повинна розраховувати відрахування на соціальне забезпечення в розмірі 5%, але не більше 20 грн. за кожний тиждень, а також утримувати державний податок у розмірі 2% і місцевий податок, стягнений у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

Варіант 27. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин, погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж роботи	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 15%, але не більше 200 грн. за кожний тиждень.

Варіант 28. Постановка задачі

Розробити програму, що дозволяє вибрати зі списку дані про товари, які входять у покупку, обчислити вартість покупки. Знижка в 1% надається, якщо сума покупки більша 300 гривень, 2% — якщо сума покупки більше 500 гривень, 3% — якщо

сума покупки більше 1000 гривень. Інформація про вартість покупки й розмір знижки повинні виводитися на екран.

ЛАБОРАТОРНА РОБОТА № 39-40

Створення та відлагодження одновіконного візуального додатку з використанням власних класів

Мета: навчитись створювати та відлагоджувати одновіконні візуальні додатки з використанням власних класів, закріпити навички з проектування класів та використання об'єктів класів

Постановка задачі.

Рекомендовано створити одновіконний візуальний додаток VCL Forms Application в середовищі Embarcadero C++ Builder, в якому реалізовано завдання за варіантом.

Частина I

Частина 1. Спроекувати візуальну форму за завданням попередньої лабораторної роботи. Оформити звіт, в якому навести вигляд візуальної форми, описати для використаних компонентів задані під час проектування властивості, обробники подій та інше. Спроекувати власний клас або класи.

Навести скріншот виконання програми з даними для перевірки виконання завдання.

Порядок виконання роботи.

Спроекувати та створити одновіконний візуальний додаток та власний клас згідно індивідуального завдання за варіантом (не в класі візуального вікна), для цього:

- проаналізувати постановку задачі за варіантом;
- сформулювати функції, які повинна виконувати програма, яка розробляється;
- виділити вхідну і вихідну інформацію, виявити типи даних та компоненти, які доцільно їм відповідають;
- спроекувати та розробити програму.

Введені значення рекомендовано хоча б в мінімальному обсязі перевіряти на допустимість.

Розробити програму згідно завдання варіанту, наведеного нижче. Врахувати, що програма повинна допускати повторне введення значень та отримання результатів. Забезпечити, щоб на візуальній формі біло видно, які дані були введені, та який отриманий результат (наприклад, вносити в багаторядковий текстовий редактор).

Варіант 1. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок (вважаються прямокутними). Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	250
2	367
3	595

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити сумарний податок по ділянках кожного типу та кількість ділянок кожного типу.

Варіант 2. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	200
2	300
3	450

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити загальну площу в квадратних метрах, загальну площу в гектарах, загальну суму оподаткування.

Варіант 3. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
А	600
Б	500
В	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу площу.

Варіант 4. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
А	600
Б	500
В	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу суму оподаткування.

Варіант 5. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, табельний номер, кількість відпрацьованих годин, кількість утриманців.

Робочий час понад 40 годин у тиждень вважається понаднормовим і оплачується в розмірі у півтора рази більше. Погодинний тариф вважається однаковим для всіх робітників і складає 25 грн./год. Програма повинна розраховувати податок, стягнутий у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

Варіант 6. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Програма повинна розраховувати податок на соціальне забезпечення стягнутий у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

а також місцевий податок в розмірі 3%, але не більше 20 грн. За результатами розрахунків повинне бути виведене повідомлення, хто з працівників отримав максимальну зарплату.

Варіант 7. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин,

погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 18%, але не більше 200 грн. за кожний тиждень.

Варіант 8. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: прізвище, кількість відпрацьованих годин, стаж роботи. Погодинний тариф вважається однаковим і складає 25 грн./год. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Оподаткування заробітної плати складає 15%, але не більше 100 грн. за кожний тиждень, а також повинен утримуватись місцевий податок, який складає 15%.

Варіант 9. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, що залежать від типу земельних ділянок. Установлені наступні податкові ставки залежно від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	250
2	367
3	595

Вхідні дані включають номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Необхідно обчислити сумарний податок по ділянках кожного типу, які включають: тип земельної ділянки, кількість ділянок, загальну площу у квадратних метрах, загальну площу в гектарах, загальну суму оподаткування.

Варіант 10. Постановка задачі

Розробити програму, що обчислює дохід по внеску. Програма повинна забезпечувати розрахунок простих і складних відсотків. Прості відсотки нараховуються наприкінці строку внеску, складні — щомісяця й додаються до початкової суми внеску й у наступному місяці відсотки нараховуються на нову суму. Прості відсотки нараховуються на внески з терміном зберігання до 1 року, складні — зі строком зберігання більше 1 року. Відсоткова ставка внеску залежить від суми внеску й задається в таблиці:

Сума внеску	Відсоткова ставка
До 1000 грн.	7%
1 – 10 тис. грн.	10%
10 тис. грн. і більше	15%

Користувач повинен мати можливість ввести суму, термін, обрати просту або складну схему нарахування відсотків і розрахувати значення суми, яку він отримає наприкінці терміну зберігання.

Варіант 11. Постановка задачі

Розробити програму, що дозволяє обчислювати вартість поїздки на автомобілях і повинна надавати користувачеві можливість вводити відстань до пункту призначення, вартість бензину, вибір варіанта поїздки: тільки в прямому напрямку або в прямому й у зворотному напрямках. Норма споживання бензину залежить від марки автомобіля й задається у вигляді таблиці.

Марка автомобіля	Норма споживання бензину (у літрах на 10 км)
Таврія	0.8

Нива	1.1
Славута	1.05

В результаті роботи програми повинна бути розрахована вартість поїздки.

Варіант 12. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: ім'я й прізвище, табельний номер, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Робочий час понад 40 годин на тиждень вважається понаднормовим і оплачується в півтора рази вище. Програма повинна розраховувати відрахування на соціальне забезпечення в розмірі 5%, але не більше 20 грн. за кожний тиждень, а також утримувати державний податок у розмірі 2% і місцевий податок, стягнений у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

Варіант 13. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин, погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж роботи	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 15%, але не більше 200 грн. за кожний тиждень.

Варіант 14. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок (вважаються прямокутними). Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	250
2	367
3	595

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити сумарний податок по ділянках кожного типу та кількість ділянок кожного типу.

Варіант 15. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	200
2	300
3	450

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити загальну площу в квадратних метрах, загальну площу в гектарах, загальну суму оподаткування.

Варіант 16. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
А	600
Б	500
В	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу площу.

Варіант 17. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
А	600
Б	500
В	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу суму оподаткування.

Варіант 18. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, табельний номер, кількість відпрацьованих годин, кількість утриманців.

Робочий час понад 40 годин у тиждень вважається понаднормовим і оплачується в розмірі у півтора рази більше. Погодинний тариф вважається

однаковим для всіх робітників і складає 25 грн./год. Програма повинна розраховувати податок, стягнутий у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

Варіант 19. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Програма повинна розраховувати податок на соціальне забезпечення стягнутий у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

а також місцевий податок в розмірі 3%, але не більше 20 грн. За результатами розрахунків повинне бути виведене повідомлення, хто з працівників отримав максимальну зарплату.

Варіант 20. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин, погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 18%, але не більше 200 грн. за кожний тиждень.

Варіант 21. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: прізвище, кількість відпрацьованих годин, стаж роботи. Погодинний тариф вважається однаковим і складає 25 грн./год. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Оподаткування заробітної плати складає 15%, але не більше 100 грн. за кожний тиждень, а також повинен утримуватись місцевий податок, який складає 15%.

Варіант 22. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, що залежать від типу земельних ділянок. Установлені наступні податкові ставки залежно від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	250
2	367
3	595

Вхідні дані включають номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Необхідно обчислити сумарний податок по ділянках кожного типу, які включають: тип земельної ділянки, кількість ділянок, загальну площу у квадратних метрах, загальну площу в гектарах, загальну суму оподаткування.

Варіант 23. Постановка задачі

Розробити програму, що дозволяє вибрати зі списку дані про товари, які входять у покупку, обчислити вартість покупки. Знижка в 1% надається, якщо сума покупки більша 300 гривень, 2% — якщо сума покупки більше 500 гривень, 3% — якщо сума покупки більше 1000 гривень. Інформація про вартість покупки й розмір знижки повинні виводитися на екран.

Варіант 24. Постановка задачі

Розробити програму, що обчислює дохід по внеску. Програма повинна забезпечувати розрахунок простих і складних відсотків. Прості відсотки нараховуються наприкінці строку внеску, складні — щомісяця й додаються до початкової суми внеску й у наступному місяці відсотки нараховуються на нову суму. Прості відсотки нараховуються на внески з терміном зберігання до 1 року, складні — зі строком зберігання більше 1 року. Відсоткова ставка внеску залежить від суми внеску й задається в таблиці:

Сума внеску	Відсоткова ставка
До 1000 грн.	7%
1 – 10 тис. грн.	10%
10 тис. грн. і більше	15%

Користувач повинен мати можливість ввести суму, термін, обрати просту або складну схему нарахування відсотків і розрахувати значення суми, яку він отримає наприкінці терміну зберігання.

Варіант 25. Постановка задачі

Розробити програму, що дозволяє обчислювати вартість поїздки на автомобілях і повинна надавати користувачеві можливість вводити відстань до пункту призначення, вартість бензину, вибір варіанта поїздки: тільки в прямому напрямку або в прямому й у зворотному напрямках. Норма споживання бензину залежить від марки автомобіля й задається у вигляді таблиці.

Марка автомобіля	Норма споживання бензину (у літрах на 10 км)
Таврія	0.8

Нива	1.1
Славута	1.05

В результаті роботи програми повинна бути розрахована вартість поїздки.

Варіант 26. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: ім'я й прізвище, табельний номер, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Робочий час понад 40 годин на тиждень вважається понаднормовим і оплачується в півтора рази вище. Програма повинна розраховувати відрахування на соціальне забезпечення в розмірі 5%, але не більше 20 грн. за кожний тиждень, а також утримувати державний податок у розмірі 2% і місцевий податок, стягнений у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

Варіант 27. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин, погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж роботи	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 15%, але не більше 200 грн. за кожний тиждень.

Варіант 28. Постановка задачі

Розробити програму, що дозволяє вибрати зі списку дані про товари, які входять у покупку, обчислити вартість покупки. Знижка в 1% надається, якщо сума покупки більша 300 гривень, 2% — якщо сума покупки більше 500 гривень, 3% — якщо сума покупки більше 1000 гривень. Інформація про вартість покупки й розмір знижки повинні виводитися на екран.

Відлагодити, прокоментувати програму, пояснити проєктування, оформити звіт.

ЛАБОРАТОРНА РОБОТА № 41-42

Створення та відлагодження програми з використанням масиву об'єктів

Мета: навчитись проектувати та створювати додатки, закріпити навички з проектування класів та використання масиву об'єктів

Постановка задачі.

Внести зміни в додаток, розроблений за завданням попередньої лабораторної роботи, згідно доповненого за варіантом завдання. Проект візуального додатку має містити модулі власних класів (класу), в які винесена логіка роботи з предметною областю.

Крім того, передбачити створення не об'єкту класу, а масиву об'єктів, в який для кожного розрахунку поміщати введені дані та результати і виводити результат на формі за натисканням кнопки "Переглянути" (те, що в попередній лабораторній роботі виводилось в Мето після кожного розрахунку, тепер виводити в Мето за натисканням кнопки, перебираючи масив об'єктів). Скріншот виконання має мати такий вигляд, щоб демонструвати введені дані і результат.

Варіант 1. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок (вважаються прямокутними). Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1 ("дачний")	250
2 ("сільськогосподарський")	367
3 ("промисловий")	595

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити сумарний податок по ділянках кожного типу та кількість ділянок кожного типу (на основі аналізу масиву об'єктів).

Варіант 2. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1 ("дачний")	200
2 ("сільськогосподарський")	300
3 ("промисловий")	450

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити загальну площу в квадратних метрах, загальну площу в гектарах, загальну суму оподаткування (на основі аналізу масиву об'єктів).

Варіант 3. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1 ("дачний")	600
2 ("сільськогосподарський")	500
3 ("промисловий")	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу площу (на основі аналізу масиву об'єктів).

Варіант 4. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
А ("дачний")	600
Б ("сільськогосподарський")	500
В ("промисловий")	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу суму оподаткування (на основі аналізу масиву об'єктів).

Варіант 5. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, табельний номер, кількість відпрацьованих годин, кількість утриманців.

Робочий час понад 40 годин у тиждень вважається понаднормовим і оплачується в розмірі у півтора рази більше. Погодинний тариф вважається однаковим для всіх робітників і складає 25 грн./год. Програма повинна розраховувати податок, стягнутий у відповідності з наступною шкалою:

Пільгова категорія	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

Програма повинна обчислити для кожної пільгової категорії кількість робітників та загальну суму податку (на основі аналізу масиву об'єктів).

Варіант 6. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Програма повинна розраховувати податок на соціальне забезпечення стягнутий у відповідності з наступною шкалою:

Пільгова категорія	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

а також місцевий податок в розмірі 3%, але не більше 20 грн. За результатами розрахунків повинне бути виведене повідомлення по кожній пільговій категорії, хто з працівників отримав максимальну зарплату (на основі аналізу масиву об'єктів).

Варіант 7. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин, погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 18%, але не більше 200 грн. за кожний тиждень. Видати загальну суму надбавки по кожному виду стажу (на основі аналізу масиву об'єктів).

Варіант 8. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: прізвище, кількість відпрацьованих годин, стаж роботи.

Погодинний тариф вважається однаковим і складає 25 грн./год. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Оподаткування заробітної плати складає 15%, але не більше 100 грн. за кожний тиждень, а також повинен утримуватись місцевий податок, який складає 15%. Видати загальну суму надбавки по кожному виду стажу (на основі аналізу масиву об'єктів).

Варіант 9. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, що залежать від типу земельних ділянок. Установлені наступні податкові ставки залежно від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	250
2	367
3	595

Вхідні дані включають номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Необхідно обчислити сумарний податок по ділянках кожного типу, які включають: тип земельної ділянки, кількість ділянок, загальну площу у квадратних метрах, загальну площу в гектарах, загальну суму оподаткування (на основі аналізу масиву об'єктів).

Варіант 10. Постановка задачі

Розробити програму, що обчислює дохід по внеску. Програма повинна забезпечувати розрахунок простих і складних відсотків. Прості відсотки нараховуються наприкінці строку внеску, складні — щомісяця й додаються до початкової суми внеску й у наступному місяці відсотки нараховуються на нову

суму. Прості відсотки нараховуються на внески з терміном зберігання до 1 року, складні — зі строком зберігання більше 1 року. Відсоткова ставка внеску залежить від суми внеску й задається в таблиці:

Сума внеску	Відсоткова ставка
До 1000 грн.	7%
1 – 10 тис. грн.	10%
10 тис. грн. і більше	15%

Користувач повинен мати можливість ввести суму, термін, обрати просту або складну схему нарахування відсотків і розрахувати значення суми, яку він отримає наприкінці терміну зберігання. Видати загальну суму отримання по кожному виду суми внеску (на основі аналізу масиву об'єктів).

Варіант 11. Постановка задачі

Розробити програму, що дозволяє обчислювати вартість поїздки на автомобілях і повинна надавати користувачеві можливість вводити відстань до пункту призначення, вартість бензину, вибір варіанта поїздки: тільки в прямому напрямку або в прямому й у зворотному напрямках. Норма споживання бензину залежить від марки автомобіля й задається у вигляді таблиці.

Марка автомобіля	Норма споживання бензину (у літрах на 10 км)
Таврія	0.8
Нива	1.1
Славута	1.05

В результаті роботи програми повинна бути розрахована вартість поїздки. Видати загальну вартість бензину по кожній марці автомобіля (на основі аналізу масиву об'єктів).

Варіант 12. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: ім'я й прізвище, табельний номер, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Робочий час понад 40 годин на тиждень вважається понаднормовим і оплачується в півтора рази вище. Програма повинна розраховувати відрахування на соціальне забезпечення в розмірі 5%, але не більше 20 грн. за кожний тиждень, а також утримувати державний податок у розмірі 2% і місцевий податок, стягнений у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

Видати загальну суму місцевого податку по кожному варіанту кількості утриманців (на основі аналізу масиву об'єктів).

Варіант 13. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин, погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж роботи	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 15%, але не більше 200 грн. за кожний тиждень. Видати загальну суму надбавки по кожному варіанту стажу роботи (на основі аналізу масиву об'єктів).

Варіант 14. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок (вважаються прямокутними). Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
-------------	--

1	250
2	367
3	595

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити сумарний податок по ділянках кожного типу та кількість ділянок кожного типу (на основі аналізу масиву об'єктів).

Варіант 15. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	200
2	300
3	450

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити загальну площу в квадратних метрах, загальну площу в гектарах, загальну суму оподаткування (на основі аналізу масиву об'єктів).

Варіант 16. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
А	600
Б	500
В	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу площу (на основі аналізу масиву об'єктів).

Варіант 17. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, який залежить від типу земельних ділянок. Установлені наступні податкові ставки в залежності від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
А	600
Б	500
В	250

Вхідні дані містять номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Програма повинна обчислити для кожного типу ділянок загальну площу в квадратних метрах, загальну суму оподаткування та видати повідомлення, ділянки якого типу мають найбільшу суму оподаткування (на основі аналізу масиву об'єктів).

Варіант 18. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, табельний номер, кількість відпрацьованих годин, кількість утриманців.

Робочий час понад 40 годин у тиждень вважається понаднормовим і оплачується в розмірі у півтора рази більше. Погодинний тариф вважається однаковим для всіх робітників і складає 25 грн./год. Програма повинна розраховувати податок, стягнутий у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
----------------------	----------

0 утриманців	16%
1 утриманець	10%
2 і більше	7%

Видати загальну суму податку по кожному варіанту кількості утриманців (на основі аналізу масиву об'єктів).

Варіант 19. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Програма повинна розраховувати податок на соціальне забезпечення стягнутий у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

а також місцевий податок в розмірі 3%, але не більше 20 грн. За результатами розрахунків повинне бути виведене повідомлення, хто з працівників отримав максимальну зарплату (на основі аналізу масиву об'єктів).

Варіант 20. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин, погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 18%, але не більше 200 грн. за кожний тиждень.

Видати загальну суму надбавки по кожному варіанту стажу (на основі аналізу масиву об'єктів).

Варіант 21. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: прізвище, кількість відпрацьованих годин, стаж роботи. Погодинний тариф вважається однаковим і складає 25 грн./год. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж, років	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Оподаткування заробітної плати складає 15%, але не більше 100 грн. за кожний тиждень, а також повинен утримуватись місцевий податок, який складає 15%.

Видати загальну суму податків (на основі аналізу масиву об'єктів).

Варіант 22. Постановка задачі

Розробити програму, що визначає розмір оподаткування земельної власності, що залежать від типу земельних ділянок. Установлені наступні податкові ставки залежно від типу ділянки:

Тип ділянки	Податкова ставка (у гривнях за гектар)
1	250
2	367
3	595

Вхідні дані включають номер земельної ділянки, тип ділянки, довжину в метрах, ширину в метрах.

Необхідно обчислити сумарний податок по ділянках кожного типу, які включають: тип земельної ділянки, кількість ділянок, загальну площу у квадратних метрах, загальну площу в гектарах, загальну суму оподаткування (на основі аналізу масиву об'єктів).

Варіант 23. Постановка задачі

Розробити програму, що дозволяє вибрати зі списку дані про товари, які входять у покупку, обчислити вартість покупки. Знижка в 1% надається, якщо сума покупки більша 300 гривень, 2% — якщо сума покупки більше 500 гривень, 3% — якщо сума покупки більше 1000 гривень. Інформація про вартість покупки й розмір знижки повинні виводитися на екран.

Видати загальну суму знижок (на основі аналізу масиву об'єктів).

Варіант 24. Постановка задачі

Розробити програму, що обчислює дохід по внеску. Програма повинна забезпечувати розрахунок простих і складних відсотків. Прості відсотки нараховуються наприкінці строку внеску, складні — щомісяця й додаються до початкової суми внеску й у наступному місяці відсотки нараховуються на нову суму. Прості відсотки нараховуються на внески з терміном зберігання до 1 року, складні — зі строком зберігання більше 1 року. Відсоткова ставка внеску залежить від суми внеску й задається в таблиці:

Сума внеску	Відсоткова ставка
До 1000 грн.	7%
1 – 10 тис. грн.	10%
10 тис. грн. і більше	15%

Користувач повинен мати можливість ввести суму, термін, обрати просту або складну схему нарахування відсотків і розрахувати значення суми, яку він отримає наприкінці терміну зберігання.

Видати загальну суму нарахованих відсотків для простих схем (на основі аналізу масиву об'єктів).

Варіант 25. Постановка задачі

Розробити програму, що дозволяє обчислювати вартість поїздки на автомобілях і повинна надавати користувачеві можливість вводити відстань до пункту призначення, вартість бензину, вибір варіанта поїздки: тільки в прямому

напрямку або в прямому й у зворотному напрямках. Норма споживання бензину залежить від марки автомобіля й задається у вигляді таблиці.

Марка автомобіля	Норма споживання бензину (у літрах на 10 км)
Таврія	0.8
Нива	1.1
Славута	1.05

В результаті роботи програми повинна бути розрахована вартість поїздки.

Видати загальну суму вартостей поїздок для кожної марки автомобіля (на основі аналізу масиву об'єктів).

Варіант 26. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату на підставі наступних даних: ім'я й прізвище, табельний номер, кількість відпрацьованих годин, погодинний тариф, кількість утриманців.

Робочий час понад 40 годин на тиждень вважається понаднормовим і оплачується в півтора рази вище. Програма повинна розраховувати відрахування на соціальне забезпечення в розмірі 5%, але не більше 20 грн. за кожний тиждень, а також утримувати державний податок у розмірі 2% і місцевий податок, стягнений у відповідності з наступною шкалою:

Кількість утриманців	Відсоток
0 утриманців	16%
1 утриманець	10%
2 і більше	7%

Видати загальну суму місцевих податків за кожним варіантом кількості утриманців (на основі аналізу масиву об'єктів).

Варіант 27. Постановка задачі

Розробити програму, що нараховує щотижневу заробітну плату робітника на підставі наступних даних: прізвище, стаж роботи, кількість відпрацьованих годин, погодинний тариф. В залежності від стажу роботи робітнику видається надбавка у відповідності з таблицею:

Стаж роботи	Відсоток
до 5 років	15%
6-10 років	20%
11 і більше	25%

Програма повинна розраховувати податок в розмірі 15%, але не більше 200 грн. за кожний тиждень.

Видати загальну суму надбавок за кожним варіантом стажу роботи (на основі аналізу масиву об'єктів).

Варіант 28. Постановка задачі

Розробити програму, що дозволяє вибрати зі списку дані про товари, які входять у покупку, обчислити вартість покупки. Знижка в 1% надається, якщо сума покупки більша 300 гривень, 2% — якщо сума покупки більше 500 гривень, 3% — якщо сума покупки більше 1000 гривень. Інформація про вартість покупки й розмір знижки повинні виводитися на екран.

Видати загальну суму знижок за кожним варіантом відсотків (на основі аналізу масиву об'єктів).

Відлагодити, прокоментувати програму, навести індивідуальний хід роботи, оформити звіт.

ЛАБОРАТОРНА РОБОТА № 43-44

Створення та відлагодження програми з підключенням до бази даних

Мета: навчитись проектувати та створювати додатки, закріпити навички з використання компонентів підключення до бази даних

Постановка задачі

Внести зміни в додаток, розроблений за завданням попередньої лабораторної роботи.

В візуальному додатку організувати підключення до бази даних.

Створити БД (допускається довільна СКБД), яка містить таблиці згідно предметної області (рекомендовано дві таблиці). Створити проект, в якому можна переглядати дані, виконувати додавання, редагування та видалення даних (доцільних, наприклад, товарів з кошику) згідно завдання.

Хід роботи

Створити БД з необхідними таблицями, заповнити даними, які в програмі не потребують додавання згідно аналізу предметної області.

Розробити програмний додаток, що працює з базою даних і виконує такі функції:

- відображення даних з таблиць бази даних;
- виконання навігації в наборі даних;
- модифікація даних за необхідністю (додавання, редагування, видалення);
- виведення результатів запитів згідно завдання.

Рекомендовано використати компоненти (наприкладі C++ Builder):

- TADOConnection, TADOTable;
- TDataSource.

Форма має містити DBGrid, DBNavigator, інше за необхідністю.

Рекомендовано встановити значення властивостей:

Об'єкт	Властивість	Значення
ADOConnection1	ConnectionString	Use Connection String – Build – Вказати джерело даних
	LoginPrompt	False
	Active	True
ADOTable1	Connection	ADOConnection1
	TableName	Назва таблиці
	Active	True
DataSource1	DataSet	Вибрати зі списку ADOTable1
DBGrid1	DataSource	Вибрати зі списку DataSource1
DBNavigator1		

Інші використані компоненти, встановлені властивості, передбачені обробники подій навести в звіті.

Продемонструвати роботу програми, оформити звіт, прокоментувати індивідуальний хід роботи та виконане самостійно проектування (описати використані властивості, методи, події тощо).

Допускається виконати завдання без використання власних класів.

Скріншот виконання має мати такий вигляд, щоб демонструвати введені дані і результат.

Оформити звіт, навести індивідуальний хід роботи.

ЛАБОРАТОРНА РОБОТА № 45-46

Створення та відлагодження програми з використанням віртуальних методів

Мета: навчитись створювати та відлагоджувати програми з використанням віртуальних методів

Частина 1

Постановка задачі. Створити консольний додаток з чотирма класами, пов'язаними ромбовидною ієрархією у відповідності зі схемою, наведеною на рисунку, і зразком програмного коду, наведеним у лістингах. Передбачити в класах ієрархії віртуальні методи. Пояснити у звіті використання віртуального успадкування. Пояснити, в яких класах які методи є віртуальними, який вид поліморфізму. У звіті прокоментувати програму, пояснити своїми словами, для чого використовується віртуальний метод і де і з якого класу викликається, надати відповіді, навести результати виконання.

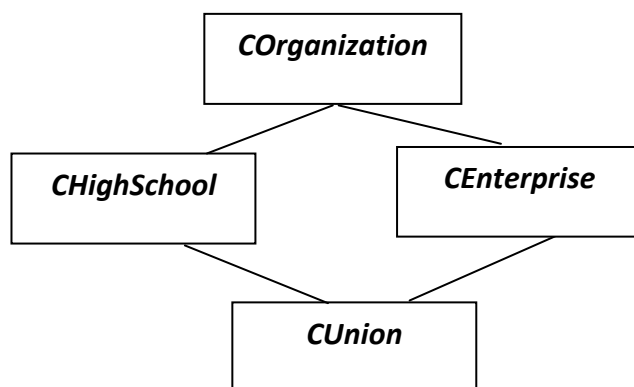


Рис. - Ієрархія класів

Порядок виконання роботи.

У лістингу похідні класи CEnterprise і CHighschool віртуально успадковують базовий клас COrganization, щоб в похідному класі CUnion не було копій підоб'єктів базового класу COrganization. Деструктор цього класу оголошений віртуальним, щоб при руйнуванні об'єктів похідних класів викликалися їх деструктори.

Метод `calcPayment()`, який обчислює суму щомісячних відрахувань в залежності від вказаного параметром `p` відсотка, оголошений як віртуальний. Цей метод перевизначений у всіх похідних класах, і для кожного класу виконує свої розрахунки.

Слід звернути особливу увагу на конструктор з параметрами похідного класу `CUnion`. У списку параметрів вказані параметри всіх класів, що стоять вище за ієрархією. Список ініціалізації починається з ініціалізації члена-даного `m_kind`. Далі вказані виклики конструкторів з параметрами класів `CHighschool` і `CEnterprise`. Два перших параметра для цих конструкторів не мають ніякого значення, так як успадкування віртуальне, і ніякої ініціалізації для об'єкта похідного класу `CUnion` автоматично проводиться не буде. Щоб дані `m_name` і `m_funds` отримали значення в результаті конструювання об'єкта, в тілі конструктора записані відповідні оператори. Для ініціалізації `m_funds` викликається метод базового класу `setFund ()`.

У головній функції через покажчики `pH`, `pE`, `pU` динамічно створюються об'єкти похідних класів. За допомогою цих покажчиків відбувається звернення до відкритих даних для виведення їх значення і до відкритих методів для отримання значень захищених даних. В кінці головної функції показано, як організувати звернення до віртуального методу через покажчик на базовий тип. Однак не можна використовувати цей покажчик для доступу до приватних даних похідного класу. Базовий клас нічого не знає про похідні класи. Через покажчик на базовий тип можна отримати доступ тільки до тієї частини похідного класу, яка успадкована від базового. У програмі можна було використовувати покажчик `p` для доступу до `m_name`, так як оголошення цього члена-даного розташоване в базовому класі.

Лістинги файлів специфікації та реалізації класів.

```
// Organization.h - Специфікація COrganization (Організація)
#include <string>
using namespace std ;
#pragma once
class COrganization
{
protected:
```

```

        double m_fund ;           // резервний фонд
public:
    string m_name ;               // назва організації
    COrganization ( ) ;
    COrganization ( const string&, const double ) ;
    virtual ~COrganization ( ) ;
    void setFund ( double ) ; double getFund ( ) const ;
    // сума щомісячних відрахувань
    virtual double calcPayment ( const int = 0 ) const ;
} ;
// Organization.cpp - Реалізація COrganization
#include "Organization.h"
COrganization :: COrganization ( ) : m_name ( " " ), m_fund ( 0.0 ) { }
COrganization :: COrganization ( const string& n, const double f )
: m_name ( n ), m_fund ( f ) { }
COrganization :: ~COrganization ( ) { }
void COrganization :: setFund ( double f ) { m_fund = f ; }
double COrganization :: getFund ( ) const { return m_fund ; }
double COrganization :: calcPayment ( const int p ) const
{
    return ( m_fund * p / 1200 ) ; }

// HighSchool.h - Специфікація CHighSchool (ВНЗ)
#pragma once
#include "Organization.h"
class CHighSchool : virtual public COrganization
{
protected:
    int m_rate ;                 // % відрахувань з фонду
public:
    CHighSchool ( ) ;
    CHighSchool ( const string&, const double, const int ) ;
    ~CHighSchool ( ) ;
    void setRate ( int ) ; int getRate ( ) const ;
    double calcPayment ( const int = 0 ) const ;
}
// HighSchool.cpp - Реалізація CHighSchool
#include "HighSchool.h"
CHighSchool :: CHighSchool ( ) : COrganization ( ), m_rate ( 0 ) { }
CHighSchool ::
CHighSchool ( const string& n, const double f, const int r )
: COrganization ( n, f ), m_rate ( r ) { }
CHighSchool :: ~CHighSchool ( ) { }
void CHighSchool :: setRate ( int r ) { m_rate = r ; }
int CHighSchool :: getRate ( ) const { return m_rate ; }
double CHighSchool :: calcPayment ( const int p ) const
{
    if ( p > m_rate ) return ( m_fund * m_rate / 1200 ) ;
    else return ( COrganization :: calcPayment ( p ) ) ;
}

```

```

}

// Enterprise.h - Специфікація CEnterprise (Підприємство)
#pragma once
#include "Organization.h"
class CEnterprise : virtual public COrganization
{
protected:
    double m_alloc ;           // розмір благодійного фонду
public:
    CEnterprise ( ) ;
    CEnterprise ( const string&, const double, const double ) ;
    virtual ~CEnterprise ( ) ;
    void setAlloc ( const double ) ; double getAlloc ( ) const ;
    double calcPayment ( const int = 0 ) const ;
} ;

// Enterprise.cpp - Реалізація CEnterprise
#include "Enterprise.h"
CEnterprise :: CEnterprise ( ) : COrganization ( ), m_alloc ( 0.0 ) { }
CEnterprise ::
CEnterprise ( const string& n, const double f, const double a )
: COrganization ( n, f ), m_alloc ( a ) { }
CEnterprise :: ~CEnterprise ( ) { }
void CEnterprise :: setAlloc ( const double a )
{    m_alloc = a ;    }
double CEnterprise :: getAlloc ( ) const { return m_alloc ; }
double CEnterprise :: calcPayment ( const int p ) const
{    return ( m_alloc * p / 1200 ) ; }

// Union.h - Специфікація CUnion (Науково-виробниче об'єднання)
#pragma once
#include <string>
using namespace std ;
#include "HighSchool.h"
#include "Enterprise.h"
class CUnion : public CHighSchool, public CEnterprise
{
public:
    string m_kind ;           // вид діяльності
    CUnion ( ) ;
    CUnion ( const string&, const double, const string&, const int,
    ↵const double ) ;
    ~CUnion ( ) ;
    double calcPayment ( const int = 0 ) const ;
} ;

// Union.cpp - Реалізація CUnion
#include "Union.h"
CUnion :: CUnion ( )

```

```

: m_kind ( "" ), CHighSchool ( ), CEnterprise ( ) { }
CUnion :: CUnion ( const string& n, const double f, const string& k,
↳const int r, const double a )
: m_kind ( k ), CHighSchool ( "", 0.0, r ), CEnterprise ( "", 0.0, a )
{   m_name = n ; setFund ( f ) ;           }
CUnion :: ~CUnion ( ) { }
double CUnion :: calcPayment ( const int ) const { return 0.0 ;

// main.cpp головна функція
#include <iostream>
using namespace std ;
#include "Union.h"
int main ( )
{
    CHighSchool* pH =
        new CHighSchool ( "UNIVERSITY OF FANS C++", 1.0e6, 2 ) ;
    cout << "\nName\t" << pH -> m_name << endl ;
    cout << "Fund\t" << pH -> getFund ( ) << endl ;
    cout << "Rate\t" << pH -> getRate ( ) << endl ;
    CEnterprise* pE =
        new CEnterprise ( "CLEVER OWL Ltd.", 1.0e8, 1.0e7 ) ;
    cout << "\nName\t" << pE -> m_name << endl ;
    cout << "Fund\t" << pE -> getFund ( ) << endl ;
    cout << "Allocation\t" << pE -> getAlloc ( ) << endl << endl ;
    CUnion* pU = new CUnion ( "VICTORY", 1.25e8, "Miracles",
↳pH -> getRate ( ), pE -> getAlloc ( ) ) ;
    cout << "\nName\t" << pU -> m_name << endl ;
    cout << "Kind\t" << pU -> m_kind << endl ;
    cout << "Fund\t" << pU -> getFund ( ) << endl ;
    cout << "Rate\t" << pU -> getRate ( ) << endl ;
    cout << "Allocation\t" << pU -> getAlloc ( ) << endl ;
    cout << "\nPayment\n" ;
    COrganization* p ;
    p = pH ;
    cout << pH-> m_name << "\t= " << p-> calcPayment ( 5 ) << endl ;
    p = pE ;
    cout << pE-> m_name << "\t= " << p-> calcPayment ( 17 ) << endl ;
    p = pU ;
    cout << pU-> m_name << "\t\t= " << p-> calcPayment ( ) << endl ;
}

```

Частина 2

Постановка задачі. Створити вконсольному додатку ієрархію класів геометричних фігур за зразком, наведеним на рисунку. У базовому класі

верхнього рівня оголосити прототип віртуальної функції знаходження площі фігури і перевизначити її в ієрархії.

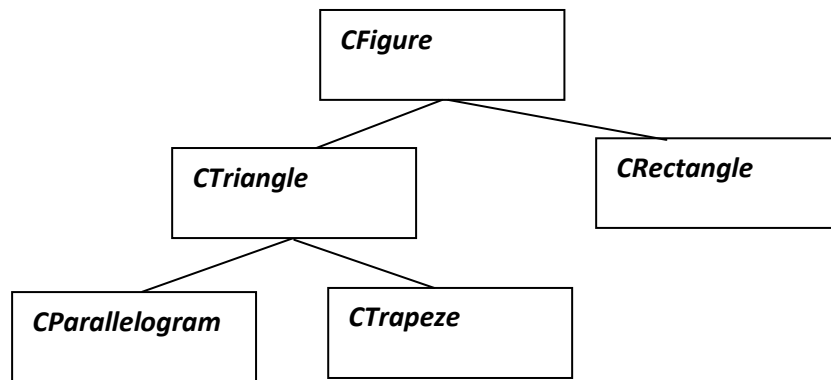


Рис. - ієрархія класів геометричних фігур

Порядок виконання роботи.

Створити в консольному додатку ієрархію класів геометричних фігур за зразком. У вершині ієрархії знаходиться клас `CFigure`, похідними від нього є класи `CTriangle` та `CRectangle`. Похідними від класу `CTriangle` є класи `CParallelogram` та `CTrapeze`. Класи пов'язані відкритим успадкуванням.

`CFigure` (фігура) – базовий для всіх класів;

`CTriangle` (трикутник) – похідний від `CFigure` (і базовий для `CParallelogram` та `CTrapeze`);

`CRectangle` (прямокутник) – похідний від `CFigure`;

`CParallelogram` (паралелограм) – похідний від `CTriangle`;

`CTrapeze` (трапеція) – похідний від `CTriangle`.

У класі `CFigure` оголошено прототип віртуальної функції `getArea()`. Всі класи-нащадки також мають функцію з таким ім'ям.

Внести зміни до файлів відповідно до лістингів цих файлів.

```
// специфікація CFigure – геометрична фігура (базовий клас)
class CFigure
{
protected:
    double m_a, m_b ;           // дві сторони фігури
public:
    CFigure ( ) ; CFigure ( const double, const double ) ;
    virtual ~CFigure ( ) ;
    // виртуальні функції
    virtual void setFigure ( double, double ) ;
}
```

```

        virtual void getFigure ( double&, double& ) const ;
        // вычисление площади
        virtual double getArea ( ) const = 0 ;
    } ;
// специфікація CTriangle - трикутник (похідний клас)
class CTriangle : public CFigure
{
protected:
    double m_c ;                // третя сторона
public:
    CTriangle ( ) ; CTriangle ( double, double, double ) ;
    ~CTriangle ( ) ;
    void setFigure ( double, double, double ) ;
    void getFigure ( double&, double&, double& ) const ;
    double getArea ( ) const ;// обчислення площі
} ;
// специфікація CRectangular - прямокутник (похідний клас)
class CRectangular : public CFigure
{
public:
    CRectangular ( ) ; CRectangular ( double, double ) ;
    ~CRectangular ( ) ;
    double getArea ( ) const ;// обчислення площі
} ;
// специфікація CParallelogram - паралелограм (похідний клас)
class CParallelogram : public CTriangle
{
public:
    CParallelogram ( ) ; CParallelogram ( double, double, double ) ;
    ~CParallelogram ( ) ;
    double getArea ( ) const ;// обчислення площі
} ;
// специфікація CTrapeze - трапеція (похідний клас)
class CTrapeze : public CTriangle
{
protected:
    double m_d, m_h ;          // четверта сторона і висота
public:
    CTrapeze ( ) ;
    CTrapeze ( double, double, double, double, double ) ;
    ~CTrapeze ( ) ;
    void setFigure ( double, double, double, double, double ) ;
    void getFigure ( double&, double&, double&, double&, double& )
const ;
    double getArea ( ) const ;// обчислення площі
} ;

```

```

// Realizations.cpp - Реалізації класів
#pragma once
#include <iostream>
// реалізація CFigure - геометрична фігура (похідний клас)
CFigure :: CFigure ( ) : m_a ( 0.0 ), m_b ( 0.0 ) { }
CFigure :: CFigure ( double a, double b ) : m_a ( a ), m_b ( b ) { }
CFigure :: ~CFigure ( ) { }
void CFigure :: setFigure ( double a, double b ) { m_a = a ; m_b = b ; }
void CFigure :: getFigure ( double& a, double& b ) const
{    a = m_a ; b = m_b ; }
//
// реалізація CTriangle - трикутник (похідний клас)
CTriangle :: CTriangle ( ) : CFigure ( ), m_c ( 0.0 ) { }
CTriangle :: CTriangle ( double a, double b, double c )
: CFigure ( a, b ), m_c ( c ) { }
CTriangle :: ~CTriangle ( ) { }
void CTriangle :: setFigure ( double a, double b, double c )
{    CFigure :: setFigure ( a, b ) ; m_c = c ; }
void CTriangle :: getFigure ( double& a, double& b, double& c ) const
{    CFigure :: getFigure ( a, b ) ; c = m_c ; }

// площа трикутника обчислюється за формулою Герона
#include <cmath>
using namespace std ;
double CTriangle :: getArea ( ) const
{
    double p = ( m_a + m_b + m_c ) / 2 ;
    return sqrt ( p * ( p - m_a ) * ( p - m_b ) * ( p - m_c ) ) ;
}
// реалізація CRectangular - прямокутник (похідний клас)
CRectangular :: CRectangular ( ) : CFigure ( ) { }
CRectangular :: CRectangular ( double a, double b )
: CFigure ( a, b ) { }
CRectangular :: ~CRectangular ( ) { }
// реалізація
double CRectangular :: getArea ( ) const { return m_a * m_b ; }
// реалізація CParallelogram - паралелограм (похідний клас)
// член-дане m_c базового класу CTriangle - це висота паралелограма
CParallelogram :: CParallelogram ( ) : CTriangle ( ) { }
CParallelogram :: CParallelogram ( double a, double b, double h )
: CTriangle ( a, b, h ) { }
CParallelogram :: ~CParallelogram ( ) { }
// реалізація
double CParallelogram :: getArea ( ) const { return m_a * m_c ; }
// реалізація CTrapeze - трапеція (похідний клас)
CTrapeze :: CTrapeze ( ) : CTriangle ( ), m_d ( 0.0 ), m_h ( 0.0 ) { }
CTrapeze :: CTrapeze ( double a, double b, double c, double d, double h
)

```

```

: CTriangle ( a, b, c ), m_d ( c ), m_h ( h ) { }
CTrapeze :: ~CTrapeze ( ) { }
void CTrapeze ::
setFigure ( double a, double b, double c, double d, double h )
{   CTriangle :: setFigure ( a, b, c ) ; m_d = d ; m_h = h ; }
void CTrapeze ::
getFigure ( double& a, double& b, double& c, double& d, double& h )
const
{   CTriangle :: getFigure ( a, b, c ) ; d = m_d ; h = m_h ; }
// реалізація
double CTrapeze :: getArea ( ) const
{   return 0.5 * ( m_a + m_b ) * m_h ; }

```

Внести зміни до головної функції проекту, яка повинна бути відредагована

у відповідності з текстом:

```

// main.cpp - главная функция
void view ( double, double ) ;
void view ( double, double, double ) ;
void view ( double, double, double, double, double ) ;
int main ( )
{
    //
    CFigure* pF [ 4 ] ;
    //
    char* name [ ] =
        { "Triangle", "Trapeze", "Parallelogram", "Rectangular" } ;
    // створення об'єктів різних типів
    CTriangle triangle ( 3.0, 4.0, 2.0 ) ;
    CTrapeze trapeze ( 5.0, 2.0, 1.9, 2.8, 1.8 ) ;
    CParallelogram parallelogram ( 15.2, 28.75, 18.3 ) ;
    CRectangular rectangular ( 100.5, 24.13 ) ;
    //
    pF [ 0 ] = &triangle ;    pF [ 1 ] = &trapeze ;
    pF [ 2 ] = &parallelogram ;    pF [ 3 ] = &rectangular ;
    // цикл виведення стану створених об'єктів та їх площадів
    double a, b, c, d, h ;
    for ( int i = 0; i < 4; i++ )
    {
        cout << '\n' << name [ i ] << " ( ";
        switch ( i )
        {
            case 0:
                triangle.getFigure ( a, b, c ) ;
                view ( a, b, c ) ; cout << " )\t\t\t" ; break ;
            case 1:
                trapeze.getFigure ( a, b, c, d, h ) ;
                view ( a, b, c, d, h ) ; cout << " )\t\t" ; break ;
            case 2:

```

```

        parallelogram.getFigure ( a, b, h ) ;
        view ( a, b, h ); cout << " )\t" ; break ;
    case 3:
        rectangular.getFigure ( a, b ) ;
        view ( a, b ) ; cout << " )\t\t" ; break ;
    }
    cout << "AREA = " << pF [ i ] -> getArea ( ) << endl ;
}
// цикл зміни стану створених об'єктів та їх площадів
for ( int j = 0; j < 4; j++ )
{
    cout << '\n' << name [ j ] << " ( ";
    switch ( j )
    {
    case 0:
        triangle.setFigure ( 3, 3, 3 ) ;
        triangle.getFigure ( a, b, c ) ;
        view ( a, b, c ) ; cout << " )\t\t\t" ; break ;
    case 1:
        trapeze.setFigure ( 5, 2, 1.8, 1.8, 1.9 ) ;
        trapeze.getFigure ( a, b, c, d, h ) ;
        view ( a, b, c, d, h ) ; cout << " )\t\t" ; break ;
    case 2:
        parallelogram.setFigure ( 14, 15, 14 ) ;
        parallelogram.getFigure ( a, b, h ) ;
        view ( a, b, h ); cout << " )\t\t" ; break ;
    case 3:
        rectangular.setFigure ( 1005, 2413 ) ;
        rectangular.getFigure ( a, b ) ;
        view ( a, b ) ; cout << " )\t\t" ; break ;
    }
    cout << "AREA = " << pF [ j ] -> getArea ( ) << endl ;
}

    return 0 ;
}
// визначення
void view ( double a, double b )
{    cout << a << ", " << b ; }
void view ( double a, double b, double c )
{    view ( a, b ) ; cout << ", " << c ; }
void view ( double a, double b, double c, double d, double h )
{    view ( a, b, c ) ; cout << ", " << d << ", " << h ; }

```

Відлагодити, прокоментувати, виконати програму.

Оформити звіт, пояснити використання віртуальних методів та створення об'єктів.

Перелічити методи, пояснити, які з них при успадкуванні залишаються віртуальними, а які ні і чому.

Пояснити використання чистого віртуального метода.

Пояснити роботу з масивом об'єктів в головній функції.

ЛАБОРАТОРНА РОБОТА №47-48

Створення та відлагодження програми з використанням абстрактних класів

Мета: навчитись створювати та відлагоджувати програми з використанням абстрактних класів.

Теоретична частина

Клас, який містить хоча б один чисто віртуальний метод, називається абстрактним. Абстрактні класи призначені для представлення спільних понять, які передбачається конкретизувати в похідних класах. Абстрактний клас може використовуватися тільки як базовий для інших класів — об'єкти абстрактного класу створювати не можна, оскільки прямий або непрямий виклик чисто віртуального методу призводить до помилки при виконанні.

При визначенні абстрактного класу необхідно мати на увазі наступне:

- абстрактний клас не можна використовувати при явному приведенні типів, для опису типу параметра і типу значення, що повертається функцією;
- допускається оголошувати покажчики та посилання на абстрактний клас, якщо при ініціалізації не потрібно створювати тимчасовий об'єкт;
- якщо клас, похідний від абстрактного, не визначає всі чисто віртуальні функції, він також є абстрактним.

Таким чином, можна створити функцію, параметром якої є вказівник на абстрактний клас. На місце цього параметра при виконанні програми може передаватися покажчик на об'єкт будь-якого похідного класу. Це дозволяє створювати поліморфні функції, що працюють з об'єктом будь-якого типу в межах однієї ієрархії.

Частина I

Постановка задачі. Створити ієрархію класів з трьох рівнів, на верхньому рівні якої клас Shape (фігура), похідні від нього Rectangle (прямокутник) та Circle

(коло). На базі класу Rectangle створити клас Square (квадрат) як похідний випадок класу Rectangle. В кожному з похідних класів перевизначити методи базового класу Draw() (намалювати), GetArea() (розрахувати площу) та ін. Відлагодити, прокоментувати текст програми. Оформити звіт, пояснити своїми словами, який клас є абстрактним і чому, для чого він призначений, які методи є віртуальними і чим це аргументовано.

Лістинг ілюструє організацію класу Shape та його похідних класів Circle та Rectangle.

// Лістинг 1. Сімейство класу Shape

```
#include <iostream>
class Shape {
public:
    Shape() {}
    virtual ~Shape() {}
    virtual double GetArea() = 0;
    virtual double GetPerim() = 0;
    virtual void Draw()= 0;
private:
};

class Circle : public Shape {
public:
    Circle();
    Circle(int radius):itsRadius(radius) {}
    ~Circle() {}
    double GetArea() { return 3.14 * itsRadius * itsRadius; }
    double GetPerim() { return 6.28 * itsRadius; }
    void Draw();
private:
    int itsRadius;
    int itsCircumference;
};

void Circle::Draw() {
    cout << "Circle drawing routine here!\n";
}

class Rectangle : public Shape {
public:
    Rectangle();
    Rectangle(int len, int width):
        itsLength(len), itsWidth(width) {}
    virtual ~Rectangle() {}
    virtual double GetArea() { return itsLength * itsWidth; }
    virtual double GetPerim() { return 2 * itsLength + 2 *
itsWidth;}
    virtual int GetLength() { return itsLength; }
    virtual int GetWidth() { return itsWidth; }
    virtual void Draw();
```

```

private:
    int itsWidth;
    int itsLength;
};
void Rectangle::Draw() {
    for (int i=0; i<itsLength; i++)
    {
        for (int j=0; j<itsWidth; j++)
            cout << "x ";
        cout << endl;
    }
}

class Square : public Rectangle {
public:
    Square();
    Square(int len);
    Square(int len, int width);
    virtual~Square() {}
    double GetPerim() { return 4 * GetLength(); }
};

Square::Square(int len):
    Rectangle(len,len) {}
Square::Square(int len, int width):
    Rectangle(len,width)
{
    if (GetLength () != GetWidth())
        cout << "Error, not a square... a Rectangle??\n";
}

int main() {
    int choice;
    bool fQuit = false;
    Shape * sp;
    while ( !fQuit )
    {
        cout << "(1)Circle (2)Rectangle (3)Square (0)Quit:
        cin >> choice;
        switch (choice) {
            case 0: fQuit = true;
            break;
            case 1: sp = new Circle(5);
            break;
            case 2: sp = new Rectangle(4,6);
            break;
            case 3: sp = new Square(5);
            break;
            default:
                cout<<"Enter a number between 0 and 3"<<endl;
                continue;
            break;
        }
        if( !fQuit )
            sp->Draw();
        delete sp;
    }
}

```

```

    sp = 0 ;
    cout << endl;
}
    return 0;
}

```

Мова C++ підтримує створення абстрактних типів даних з чистими віртуальними функціями. Чистими віртуальними називають такі функції, які ініціалізуються нульовим значенням, наприклад: `virtual void Draw() = 0;`

В наведеному прикладі класи містять функцію `Draw()`, але реалізація її порожня, викликати її не можна. В похідних класах її можна перевизначити. Будь-який клас з однією або декількома чистими віртуальними функціями є абстрактним, а створення екземплярів об'єкту такого класу заборонено.

Будь-який клас, похідний від абстрактного, успадковує чисті віртуальні функції в початковому вигляді та повинен перевизначити їх, якщо передбачається створення об'єкту цього класу. Таким чином, якщо клас `Rectangle` успадковує клас `Shape`, а `Shape` має три чисті віртуальні функції, то клас `Rectangle` повинен перевизначити всі три, інакше він також залишиться абстрактним.

Частина II

Постановка задачі. Розрахувати відповідно до варіанту для двох геометричних тіл їх об'єм та площу поверхні. Розробити програму, в якій демонструється використання абстрактного класу та чистих віртуальних функцій, а також окремих класів для кожного виду геометричної фігури. Розробити трирівневу ієрархію класів: базовий клас – «фігура», класи, що його успадковують – «багатогранники», «круглі тіла»; класи, що їх успадковують — конкретні фігури. Програма має надавати користувачеві можливості вибрати конкретне геометричне тіло, ввести початкові дані для розрахунку і отримати необхідні характеристики.

Відлагодити, прокоментувати текст програми. Оформити звіт, пояснити, який клас є абстрактним і чому, для чого він призначений, які методи є віртуальними і чим це аргументовано.

Варіанти завдань:

№ варіанту	Геометричні тіла, що входять у варіант	
	Багатогранники	Круглі тіла
1	Трикутна призма	Конус
2	Трикутна піраміда	Циліндр
3	Усічена трикутна піраміда	Усічений конус
4	Паралелепіпед	Півкуля
5	Куб	Конус
6	Паралелепіпед	Усічений конус
7	Усічена чотирикутна піраміда	Циліндр
8	Чотирикутна призма	Куля
9	Чотирикутна піраміда	Півкуля
10	Куб	Конус
11	Паралелепіпед	Куля
12	Трикутна призма	Циліндр
13	Трикутна призма	Конус
14	Трикутна піраміда	Циліндр
15	Усічена трикутна піраміда	Усічений конус
16	Паралелепіпед	Півкуля
17	Куб	Конус
18	Паралелепіпед	Усічений конус
19	Усічена чотирикутна піраміда	Циліндр
20	Чотирикутна призма	Куля
21	Чотирикутна піраміда	Півкуля
22	Куб	Конус
23	Паралелепіпед	Куля
24	Трикутна призма	Циліндр
25	Трикутна призма	Конус
26	Трикутна піраміда	Циліндр
27	Усічена трикутна піраміда	Усічений конус
28	Паралелепіпед	Півкуля

29	Куб	Конус
30	Паралелепіпед	Усічений конус

ЛАБОРАТОРНА РОБОТА № 49-50

Створення та відлагодження візуального додатку з використанням статичних елементів класів

Мета роботи: навчитись створювати та відлагоджувати програми з використанням візуального інтерфейсу, абстрактних класів та статичних елементів класів

Абстрактні класи

Клас, який містить хоча б один чисто віртуальний метод, називається абстрактним. Абстрактні класи призначені для представлення спільних понять, які передбачається конкретизувати в похідних класах. Абстрактний клас може використовуватися тільки як базовий для інших класів - об'єкти абстрактного класу створювати не можна, оскільки прямий або непрямий виклик чисто віртуального методу призводить до помилки при виконанні.

Мова C++ підтримує створення абстрактних типів даних з чистими віртуальними функціями. Чистими віртуальними називають такі функції, які ініціалізуються нульовим значенням, наприклад: `virtual void Draw() = 0;`

Будь-який клас, похідний від абстрактного, успадковує чисті віртуальні функції в початковому вигляді та повинен перевизначити їх, якщо передбачається створення об'єкту цього класу.

Постановка задачі

Розрахувати відповідно до варіанту для двох геометричних тіл їх об'єм та площу поверхні.

Розробити візуальну програму, в якій демонструється використання власного абстрактного класу та чистих віртуальних функцій, а також окремих класів для кожного виду геометричної фігури з перевантаженням віртуальних функцій. Базовий клас – «фігура», класи, що його успадковують – «багатогранники», «круглі тіла». Класи, що їх успадковують – конкретні фігури.

Програма повинна надавати користувачеві можливості вибрати конкретне геометричне тіло, ввести початкові дані для розрахунку з проведенням контролю даних і отримати необхідні характеристики. Розрахунок проводити за сторонами, в класах передбачити закриті дані та необхідні перевірки на діапазон та допустимість.

Програма повинна бути розроблена з використанням візуальних компонентів для введення та виведення даних. Вказати, який проект створено.

Програма повинна надавати користувачеві можливості :

- вибрати конкретне геометричне тіло з будь-якої групи, ввести початкові дані (СТОРОНИ, А НЕ ПЛОЩІ) для розрахунку і отримати необхідні характеристики;

- після проведення будь-якого розрахунку для конкретних геометричних тіл користувач повинен мати можливість отримати інформацію про те, які сумарні характеристики (об'єм, площа поверхні) по кожній з груп геометричних тіл;

- після виконання будь-якого розрахунку користувач повинен мати можливість порівняння однойменних результатів для різних груп геометричних тіл.

Виконати, пояснити, оформити звіт. В звіті приділити увагу проектуванню інтерфейсу та прокоментувати створення об'єктів власних класів.

Варіанти завдань:

№ варіан ту	Геометричні тіла, що входять у варіант	
	Багатогранники	Круглі тіла
1	Трикутна призма	Конус
2	Трикутна піраміда	Циліндр
3	Усічена трикутна піраміда	Усічений конус
4	Паралелепіпед	Півкуля
5	Куб (формула: a^3)	Конус (формула: $S = S_{\text{бок.по}} + S_{\text{ос}} = \prod R L + \prod R^2$

		)
6	Паралелепіпед	Усічений конус
7	Усічена чотирикутна піраміда	Циліндр
8	Чотирикутна призма	Куля
9	Чотирикутна піраміда	Півкуля
10	Куб	Конус
11	Паралелепіпед	Куля
12	Трикутна призма	Циліндр
13	Трикутна призма	Конус
14	Трикутна піраміда	Циліндр
15	Усічена трикутна піраміда	Усічений конус
16	Паралелепіпед	Півкуля
17	Куб	Конус
18	Паралелепіпед	Усічений конус
19	Усічена чотирикутна піраміда	Циліндр
20	Чотирикутна призма	Куля
21	Чотирикутна піраміда	Півкуля
22	Куб	Конус
23	Паралелепіпед	Куля
24	Трикутна призма	Циліндр

Оформити звіт, прокоментувати використання віртуальних функцій та програмний код.

ЛАБОРАТОРНА РОБОТА № 51-52

Відношення між класами в програмі. Створення та відлагодження програми

Мета: навчитись складати та відлагоджувати програми з пов'язаними між собою класами.

Постановка задачі.

1 Клас "Книга" містить елементи, що описують книгу. Клас "Каталог" містить інформацію щодо переліку книг і функціонал пошуку книги, додавання, сортування в каталозі тощо. Розподілити обов'язки між трьома класами: CBook, CCatalog - звичайні класи, а функціонал, запропонований для класу CMenu, реалізувати на формі у вигляді кнопок тощо (за натисканням звертатися до каталогу або книги), на цій же формі спроектувати інтерфейс для введення та виведення.

Створити класи згідно завдання, для цього:

- проаналізувати постановку задачі;
- сформулювати функції, які повинна виконувати програма;
- виділити вхідну і вихідну інформацію, виявити типи даних, які доцільно їм відповідають;
- спроектувати та розробити програму;
- прокоментувати, оформити звіт.

Створити власні класи: візуальна форма (описує користувацький інтерфейс, реалізовує функціонал наведеного нижче для зразка класу CMenu), клас CCatalogue## (описує масив із 100 вказівників на об'єкти типу CBook##) та клас CBook##, де ## — **двозначний номер згідно варіанту**. Варіанти наведені в оголошенні курсу ООП.

В лістингах наведені фрагменти програмного коду для класів CMenu (зразок для функціоналу візуальної форми), CCatalogue, CBook та програмного коду головної функції як приклад, як можна було б реалізувати в консольному додатку.

Дописати запропоновані нижче функції.

Зверніть увагу, що у цьому прикладі в клас CBook доданий новий метод check(), який перевіряє збіг значення параметра методу і значення члена-даного m_title. Метод використовується об'єктом класу CCatalogue при пошуку книги в каталозі за назвою. Об'єкт класу CCatalogue не в змозі самостійно виконати перевірку на збіг значення змінної key з назвою книги m_title, тому що назва належить книзі (*m_pBook [i]), а не каталогу (m_p), якому відома лише адреса книги (m_pBook [i]). Тільки сама книга знає свою назву і може відповісти, чи збігається вона з key. Така організація взаємодії програмних об'єктів menu, *m_p і *m_pBook[i] робить програму об'єктно-орієнтованою, коли обов'язки розподіляються між об'єктами і кожен з них несе персональну відповідальність за їх виконання.

Рекомендовано використовувати перевантаження операцій, при необхідності роботу з файлами.

Запропонувати реалізацію наведеного коду в візуальному додатку C++ Builder або іншому (мовою C++) з дотриманням функціоналу наведених класів в цілому та незначною зміною інтерфейсів класів за необхідністю. Введення та виведення виконувати з використанням інтерфейсу.

```
// Зразок програмного коду головної функції, як це було б в консольному додатку
```

```
    CMenu menu ;menu.showMenuAndSelect ( ) ;
```

```
// Menu.h Специфікація класу меню CMenu, як це було б в консольному додатку
```

```
#include "Catalogue.h"
```

```
class CMenu {
```

```
public:
```

```
    CCatalogue* m_p ;           // вказівник на каталог книг
```

```
    CMenu ( ) ; ~CMenu ( ) ;    // конструктор і деструктор
```

```
    void showMenuAndSelect ( ) ; // активізація меню
```

```
private:
```

```
    void find ( ) const ;        // пошук книги за назвою
```

```
    void view ( ) const ;        // перегляд стану каталогу
```

```
    void ins ( ) ;               // додавання книги
```

```
    void del ( ) ;               // видалення книги за назвою
```

```
    void sort ( ) ;              // сортування в каталозі за автором
```

```
    void align ( int ) const ;   // вирівнювання при виведенні
```

```
};
```

```
// Menu.cpp Реалізація класу CMenu, як це було б в консольному додатку
```

```

#include <iostream>
#include <iomanip>
using namespace std ;
#include "Menu.h"
const static int N ( 512 ) ;
CMenu :: CMenu ( ) : m_p ( new CCatalogue ) {} // конструктор
CMenu :: ~CMenu ( ) { delete m_p ; } // деструктор
void CMenu :: showMenuAndSelect ( ) { // активізація меню
    char choice ;
    do {
        cout << "\n1. Find and look up" ;
        cout << "\n2. Insertion" ;
        cout << "\n3. Deletion" ;
        cout << "\n4. Sorting" ;
        cout << "\n5. View" ;
        cout << "\n0. Exit\n" ;
        cout << "Please, your choice -> " ;
        cin >> choice ; cin.ignore ( ) ;
        if ( m_p -> getNumberOfBook ( ) )
            switch ( choice )
            {
                case '1': find ( ) ; break ;
                case '2': ins ( ) ; break ;
                case '3': del ( ) ; break ;
                case '4': sort ( ) ; break ;
                case '5': view ( ) ; break ;
            }
        else
            if ( ! m_p -> getNumberOfBook ( ) && choice == '2' )
                ins ( ) ;
    } while ( choice != '0' ) ;
}
void CMenu :: find ( ) const{ // пошук по ключу
    string title ;
    cout << "\nEnter Title -> " ; getline ( cin, title ) ;
    int iKey = m_p -> find( title ) ;
    if ( iKey != -1 )
    {
        cout << "Author: " ;
        cout << m_p -> m_pBook [ iKey ] -> getAuthor( ) << endl ;
        cout << "Title" ;
        cout << m_p -> m_pBook [ iKey ] -> getTitle( ) << endl ;
        cout << "Year: " ;
        cout << m_p -> m_pBook [ iKey ] -> getYear( ) << endl ;
    }
    else cout << title << " - NOT FOUND\n" ;
}
// Розробити самостійно реалізацію функції void CMenu :: view ( ) const // перегляду
інформації всіх книг для візуальної форми

void CMenu :: view ( ) const // перегляд стану
{
...
}

```

```

void CMenu :: ins ( )                                // додавання
{
    string author, title ; int year ;
    cout << "\nEnter Title to add -> " ; getline ( cin, title ) ;
    if ( m_p -> find ( title ) < 0 )
    {
        cout << "\nEnter Author to add -> " ;
        getline ( cin, author ) ;
        cout << "\nEnter Year to add -> " ; cin>> year ;
        if ( m_p -> ins ( author, title, year ) )
            cout << "\nINSERTION IS COMPLETED!\n";
    }
    else cout << title << " - INSERTION IS IMPOSSIBLE\n" ;
}
void CMenu :: del ( )                                // видалення
{
    string title ;
    cout << "\nEnter Title -> " ; getline ( cin, title ) ;
    if ( m_p -> del ( title ) )      cout << "\nDELETION IS
COMPLETED!\n";
    else cout << title << " - DELETION IS IMPOSSIBLE\n" ;
}
void CMenu :: sort ( )                                // сортування
{    m_p -> sort ( ) ;      cout << "\nSORTING IS COMPLETED!\n" ;      }
void CMenu :: align ( int n ) const                   // вирівнювання
{    cout.width ( n ) ; cout.setf ( ios_base::left ) ;      }

// Catalogue.h Специфікація класу CCatalogue
#pragma once
static const int MAX_SIZE = 100 ;
#include "Book.h"
class CCatalogue
{
    int m_q ;                                // фактична кількість книг
public:
    CBook* m_pBook [ MAX_SIZE ] ; // масив вказівників на книги
public:
    CCatalogue ( ) ; ~CCatalogue ( ) ; // конструктор і деструктор
    bool ins ( const string&, const string&, int ) ; // додавання
    bool del ( const string& ) ; // видалення
    int find ( const string& ) const ; // пошук
    void sort ( ) ; // сортування
    int getNumberOfBook ( ) const ; // повернення кількості книг
} ;

// CCatalogue.cpp Реалізація класу CCatalogue
#include "Catalogue.h"
CCatalogue :: CCatalogue ( ) : m_q ( 0 ) { } // конструктор
CCatalogue :: ~CCatalogue ( ) // деструктор
{    for ( int i = 0; i < m_q; i++ ) delete m_pBook [ i ] ;      }
// повернення фактичної кількості книг
int CCatalogue :: getNumberOfBook ( ) const { return m_q ; }
/*    додавання нової книги
1 перевірити наявність книги в масиві;
2 якщо книга вже є в масиві, повернути false
якщо книги немає ->

```

```

        - створити новий об'єкт і занести в масив;
        - збільшити кількість книг в масиві і повернути true      */
bool CCatalogue ::
ins ( const string& author, const string& title, int year )
{
    if ( find ( title ) != -1 ) return false ;
    m_pBook [ m_q++ ] = new CBook ( author, title, year ) ;
    return true ;
}
/*   видалення книги з масиву
1 перевірити наявність книги в масиві
2 якщо книги немає в масиві, повернути false
   якщо книга знайдена ->
       - звільнити знайдений об'єкт;
       - стиснути масив;
       - зменшити кількість книг в масиві і повернути true.*/
bool CCatalogue :: del ( const string& key )
{
    int i = find ( key ) ;    if ( i < 0 )    return false ;
    delete m_pBook [ i ] ;
    while ( i < m_q - 1 ) m_pBook [ i++ ] = m_pBook [ i + 1 ] ;
    m_q-- ;    return true ;
}
/*   пошук книги за назвою
1 перевірити назву кожної книги;
2 якщо є співпадання, повернути індекс книги
   якщо немає співпадань, повернути -1.      */
int CCatalogue :: find ( const string& key ) const
{
    for ( int i = 0; i < m_q; i++ )
        if ( m_pBook [ i ] -> check ( key ) )    return i ;
    return -1 ;
}

```

// Розробити самостійно реалізацію функції. Непарні варіанти - сортувати за прізвищем автора, парні - за назвою.

// сортування масиву

```

void CCatalogue :: sort ( )
{
    ...
}
// Book.h - специфікація класу CBook
#pragma once
#include <string>
using namespace std ;
class CBook
{
public:
    CBook ( ) ; CBook ( const string&, const string&, int ) ;
    ~CBook ( ) ;
private:
    string m_author ;    // автор
    string m_title ;    // назва
    int m_year ;    // рік видання
}

```

```

public:
    // методи встановлення значень
    void setAuthor ( const string & ) ;
    void setTitle ( const string & ) ;
    void setYear ( const int ) ;
    // методи повернення значень
    string getAuthor ( ) const { return m_author ; }
    string getTitle ( ) const { return m_title ; }
    int getYear ( ) const { return m_year ; }
    // метод перевірки назви книги
    bool check ( const string& ) const ;
} ;
// Book.cpp - реалізація класу CBook
#include "Book.h"
CBook :: CBook ( ) : m_year ( 0 ), m_title ( "" ), m_author ( "" ) {}
CBook :: CBook ( const string& author, const string& title, int year )
: m_year ( year ), m_title ( title ), m_author ( author ) { }
CBook :: ~CBook ( ) { }
// встановити автора
void CBook :: setAuthor ( const string& author ) { m_author = author ; }
// встановити назву
void CBook :: setTitle ( const string& title ) { m_title = title ; }
// встановити рік видання
void CBook :: setYear ( const int year ) { m_year = year ; }
// перевірка книги за назвою
bool CBook :: check ( const string& key ) const
{
    if (m_title == key ) return true ;    return false ; }

```

Прокоментувати програму, оформити звіт, навести **індивідуальний** хід роботи.

Проаналізувати, як пов'язані між собою класи в програмі, відстежити послідовність створення об'єктів, ініціалізацію полів. **Прокоментувати в звіті**, як організовано взаємодію об'єктів, як розподілений функціонал між класами, який клас за що відповідає і як обмінюється інформацією з іншим класом.

Розробка інтерфейсу додатку. Створення та відлагодження програми

Мета: навчитись проектувати інтерфейс додатку, складати та відлагоджувати програми з пов'язаними між собою класами

Постановка задачі.

Доповнити функціонал програми попередньої лабораторної роботи наведеним нижче на основі створених власних класів: візуальної форми, класу `SCatalogue##` та класу `SBook##`, де `##` — двозначний номер згідно варіанту.

1 Клас "книга" містить додатково назву видавництва, кількість сторінок, перелік співавторів при наявності.

Передбачити в інтерфейсі можливість вибору зі списку (`ListBox` або інший) назви видавництва під час додавання нової книги. Відповідний список з переліком видавництв можна заповнювати при створенні візуального вікна.

Рік видавництва має відповідати доцільному діапазону (передбачити самостійно).

Надати можливість вводити список співавторів і зробити необов'язковим.

Передбачити в інтерфейсі перемикач (`RadioGroup` або інший) для різних варіантів пошуку: за назвою книги, за прізвищем автора, за назвою видавництва, за роком видання (не раніше вказаного). У класі спроектувати відповідний метод (методи). Інформацію про знайдені книги вивести в окремий список вікна.

2 Клас "каталог" містить додатково назву каталогу (наприклад, "Технічна література" тощо). Передбачити можливість роботи з декількома каталогами книг, переміщення книги з одного каталогу в інший, виведення кількості книг в кожному каталозі. В програмі можна передбачити певну кількість конкретних каталогів і проініціалізувати список з переліком каталогів (`ListBox` або інший) при створенні форми.

Прокоментувати програму, оформити звіт, навести індивідуальний хід роботи.

Проаналізувати, як пов'язані між собою класи в програмі, відстежити послідовність створення об'єктів, ініціалізацію полів. Прокоментувати в звіті, як організовано взаємодію об'єктів, як розподілений функціонал між класами, який клас за що відповідає і як обмінюється інформацією з іншим класом.

Створення багатовіконного візуального додатку. Аналіз взаємозв'язку класів в додатку

Мета: навчитись проектувати інтерфейс багатовіконного візуального додатку, складати та відлагоджувати програми з пов'язаними між собою класами

Теоретична частина.

Як правило, візуальні додатки містять декілька вікон (візуальних форм тощо), між якими розподілений функціонал за своєю логікою. Наприклад, одна візуальна форма може бути призначена для організації "меню" (вибору дії, поки що реалізовуємо компонентами-кнопками) та перегляду тощо, а інша — для додавання або редагування. Вони будуть мати відповідні інтерфейси.

В середовищі C++ Builder додати другу форму в проект можна з пункту головного меню "File" => "New" => "VCL Form - C++ Builder", нехай це буде Form2. Задайте їй в Інспекторі об'єктів властивість Visible=False.

Тоді по натисканню на кнопку "Додати" на першій формі другу форму можна викликати так: `Form2->ShowModal();`

З іншого боку, після виконання всіх дій на Form2 закрити її можна з використанням методу `Form2->Close();`

Зверніть увагу, що при наявності декількох модулів постає питання, що якусь змінну оголошують в одному модулі, а використовують в інших. Для цього згадайте, як створюються змінні, доступні в різних модулях програми (extern).

Постановка задачі.

Доповнити функціонал програми попередньої лабораторної роботи наведеним нижче на основі створених власних класів візуальної форми, класу `SCatalogue##` та класу `SBook##`, де ## — двозначний номер згідно варіанту:

1 Передбачити в додатку другу візуальну форму, в яку винести інтерфейс додавання або редагування книги. Функціонал зв'язку з класом "Книга" залишається таким же, тобто, введена на другій формі інформація передається в клас "Книга" для додавання або редагування, у випадку успішного виконання друга форма закривається.

2 Перша візуальна форма залишається для реалізації меню роботи і для перегляду інформації (всієї або за пошуком).

3 Для того, щоб списки "каталог", "видавництво" на двох формах заповнювалися з одного джерела, передбачити відповідні константні масиви, з яких заповнювати списки (ListBox або інші), наприклад, під час створення форми.

Прокоментувати програму, оформити звіт, навести індивідуальний хід роботи. Проаналізувати, як пов'язані між собою класи в програмі, відстежити послідовність створення об'єктів, ініціалізацію полів. Прокоментувати в звіті, як організовано взаємодію об'єктів, як розподілений функціонал між класами, який клас за що відповідає і як обмінюється інформацією з іншим класом.