

Лабораторне робота 7 (2023) [1]

Успадкування

Успадкування – один із трьох фундаментальних принципів об'єктно-орієнтовного програмування, оскільки саме завдяки йому можливе створення ієрархічних класифікацій. Використовуючи успадкування, можливо створити загальний клас, який визначає характеристики, що властиві множині зв'язаних елементів. Цей клас може бути успадкований іншими, вузькоспеціалізованими класами з додаванням у кожен з них своїх, унікальних особливостей (змінення існуючих методів і долучення власних полів і методів).

Клас, який успадковується, називається *базовим* (або *предком*). Клас, який успадковує базовий клас, називається *похідним* (або *нащадком*). Похідний клас можна використовувати як базовий для іншого похідного класу. Таким чином вбудовується багаторівнева ієрархія класів.

Найважливішою властивістю успадкування є те, що воно дає можливість уникати повторень коду, адже спільний для множини подібних класів код може бути винесено у методи їх спільного предка.

Для створення похідного класу використовується ключове слово ***class***, після якого слід записати ім'я нового класу, двокрапку, ключ доступу класу (*public*, *private*, *protected*), а потім ім'я базового класу:

```
class <ім'я_похідного_класу>:[public|protected|private]<ім'я_базового_класу>
{
    <тіло класу>
};
```

У наступному прикладі клас ***Person*** є базовим, а клас ***Employee*** – похідним.

```
#include <iostream>
#include <string>

using namespace std;
class Person //базовий клас
{
public:
```

```
string name;
int age;
void display()
{
    cout << "Name: " << name << "\tAge: " << age << endl;
}
};

class Employee : public Person
{
public:
    string company;
};

int main()
{
    Person tom;
    tom.name = "Tom";
    tom.age = 23;
    tom.display();

    Employee bob;
    bob.name = "Bob";
    bob.age = 31;
    bob.company = "Microsoft";
    bob.display();

    return 0;
}
```

7.1. Керування доступом до членів базового класу

Якщо один клас успадковує інший, члени базового класу стають членами похідного. Статус доступу членів базового класу в похідному класі визначається специфікатором доступу, який використовується для успадкування базового класу. Специфікатор доступу базового класу виражається одним з ключових слів: *public*, *private* або *protected*.

Якщо специфікатор доступу не вказаний, то за замовчуванням використовується

доц.Розова Л.В. Об'єктно-орієнтоване програмування. Спец-ті 113, 122. ІКМПФіМ. НТУ «ХПІ»
специфікатор *private*.

При оголошенні члена класу відкритим (з використанням ключового слова *public*), до нього можна отримати доступ із будь-якої частини програми. Якщо член класу оголошується закритим (за допомогою специфікатора *private*), до нього можуть отримати доступ тільки члени того самого класу. До закритих членів базового класу не мають доступу похідні класи. Якщо член класу оголошується захищеним (*protected*-членом), до нього можуть отримати доступ лише члени цього або похідних класів. Таким чином, специфікатор *protected* дозволяє успадковувати члени, але залишає їх закритими в рамках ієрархії класів.

Якщо базовий клас успадковується з використанням ключового слова *public*, його *public*-члени стають *public*-членами похідного класу, а його *protected*-члени – *protected*-членами похідного класу.

Якщо базовий клас успадковується з використанням ключового слова *protected*, його *public*- і *protected*-члени стають *protected*-членами похідного класу.

Якщо базовий клас успадковується з використанням ключового слова *private*, його *public*- і *protected*-члени стають *private*-членами похідного класу.

В усіх випадках *private*-члени базового класу залишаються закритими в рамках цього класу і не успадковуються.

Керування доступом до елементів базового і похідного класу при різних ключах доступу під час успадкування наведено в таблиці 1.

```
class A
{
public:
    int x;
protected:
    int y;
private:
    int z;
};
class B : public A
{
```

```

// x – public
// y – protected
// z – не має доступу
};
class C : protected A
{
    // x – protected
    // y – protected
    // z – не має доступу
};
class D : private A
{
    // x – private
    // y – private
    // z – не має доступу
};

```

Таблиця 1 – Режими доступу до елементів при успадкуванні

Режим доступу до елемента в базовому класі	Ключ доступу при успадкуванні класу	Режим доступу до елемента в похідному класі
private	public	недоступний
protected		protected
public		public
private	protected	недоступний
protected		protected
public		protected
private	private	недоступний
protected		private
public		private

7.2. Конструктори та деструктори при успадкуванні

Оскільки конструктори не успадковуються, при створенні похідного класу, члени, які ним успадковуються, мають бути ініціалізовані конструктором базового класу. Конструктор базового класу викликається автоматично і виконується до конструктора похідного класу.

```
#include <iostream>
```

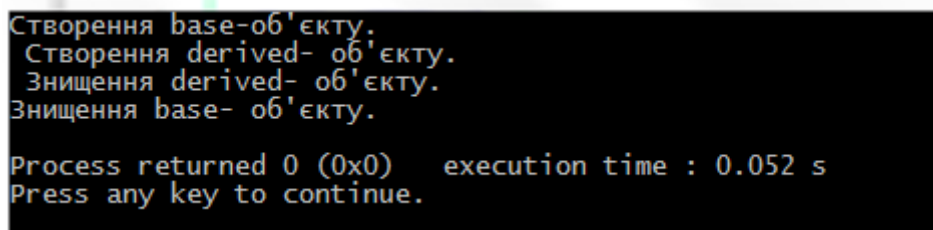
```
using namespace std;

class base {
public:
base() { cout <<"Створення base-об'єкта.\n"; }
~base() { cout <<"Знищення base-об'єкта.\n"; }
};

class derived: public base {
public:
derived() { cout <<" Створення derived-об'єкта.\n"; }
~derived() { cout <<" Знищення derived-об'єкта.\n"; }
};

int main()
{
derived ob;
return 0;
}
```

Результат виконання програми наведено на рис. 16.



```
Створення base-об'єкту.
Створення derived- об'єкту.
Знищення derived- об'єкту.
Знищення base- об'єкту.

Process returned 0 (0x0)   execution time : 0.052 s
Press any key to continue.
```

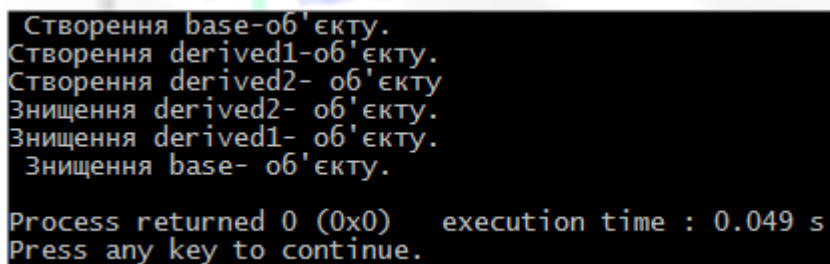
Рис. 16. Результат роботи програми

```
#include <iostream>
using namespace std;
class base {
public:
base() { cout <<" Створення base-об'єкта.\n"; }
~base() { cout <<" Знищення base-об'єкта.\n"; }
};

class derived1 : public base {
public:
derived1() { cout <<"Створення derived1-об'єкта.\n"; }
~derived1() { cout <<"Знищення derived1-об'єкта.\n"; }
};
```

```
class derived2: public derived1 {  
public:  
derived2() { cout <<"Створення derived2-об'єкта\n";}  
~derived2(){ cout <<"Знищення derived2-об'єкта.\n";}  
};  
int main()  
{  
derived2 ob;  
return 0;  
}
```

Результат виконання програми наведено на рис. 17.



```
Створення base-об'єкту.  
Створення derived1-об'єкту.  
Створення derived2- об'єкту  
Знищення derived2- об'єкту.  
Знищення derived1- об'єкту.  
Знищення base- об'єкту.  
  
Process returned 0 (0x0)   execution time : 0.049 s  
Press any key to continue.
```

Рис. 17. Результат роботи програми

Таким чином, конструктори викликаються по черзі походження класів, а деструктори – в зворотному порядку.

Параметри конструктора базового класу вказуються при визначенні конструктора похідного класу. Таким чином, виконується передача аргументів від конструктора похідного класу конструктору базового класу.

У випадку, коли необхідно передати параметри конструктору базового класу, необхідно використовувати розширену форму оголошення конструктора похідного класу, в якій передбачена можливість передачі аргументів одному чи декільком конструкторам базового класу. Загальний формат такого розширеного оголошення:

Конструктор_похідного_класу (список аргументів)

: конструктор_базового_класу (список аргументів)

```
{  
тіло конструктора похідного класу  
}
```

```
#include <iostream>
using namespace std;
class base
{
protected:
    int i;
public:
    base (int x)
    {
        i = x;
        cout << "Створення base-об'єкта.\n";
    }
    ~base() {cout << "Знищення base- об'єкта.\n";}
};
class derived: public base
{
    int j;
public:
    // Клас derived використовує параметр x, а параметр y передається конструктору класу base.
    derived(int x, int y): base(y)
    {
        j = x;
        cout << "Створення derived1-об'єкта.\n";
    }
    ~derived() { cout <<"Знищення derived1- об'єкта.\n";}
    void show() { cout << i << " " << j << "\n"; }
};
int main()
{
    derived ob(3, 4)
    ob.show(); //
    return 0;
}
```

Результат виконання програми наведено на рис. 16.

```
Створення base-об'єкту.  
Створення derived1-об'єкту.  
4 3  
Знищення derived1- об'єкту.  
Знищення base- об'єкту.
```

Рис. 18. Приклад виконання програми

Якщо базовий клас містить тільки конструктори з параметрами, то похідний клас має викликати в своєму конструкторі один з конструкторів базового класу.

7.3. Принцип підстановки

Відкрите успадкування встановлює між класами *відношення «є»*: похідний клас є частиною базового класу. Це означає, що усюди, де може бути використаний об'єкт базового класу (при присвоюванні, при передачі параметрів та поверненні результату), замість нього дозволяється використовувати об'єкт похідного класу. Це положення має назву «принцип підстановки». При цьому зворотне невірне, тобто будь-який студент (похідний клас) є людиною (базовий клас), але не будь-яка людина є студентом [3].

Розглянемо основні елементи успадкування класів на наступному прикладі. Створимо базовий клас **Car** (машина), що характеризується торговою маркою, числом циліндрів, потужністю. Визначимо методи перепризначення і зміни потужності. Створимо похідний клас **Lorry** (вантажівка), додамо в нього характеристику вантажопідйомності кузова. Визначимо функції перепризначення марки і зміни вантажопідйомності. Реалізуємо функцію, яка одержує і повертає об'єкти базового класу. Застосуємо на практиці також на принцип підстановки.

UML-діаграма класів **Car** та **Lorry** наведено на рисунку 19.

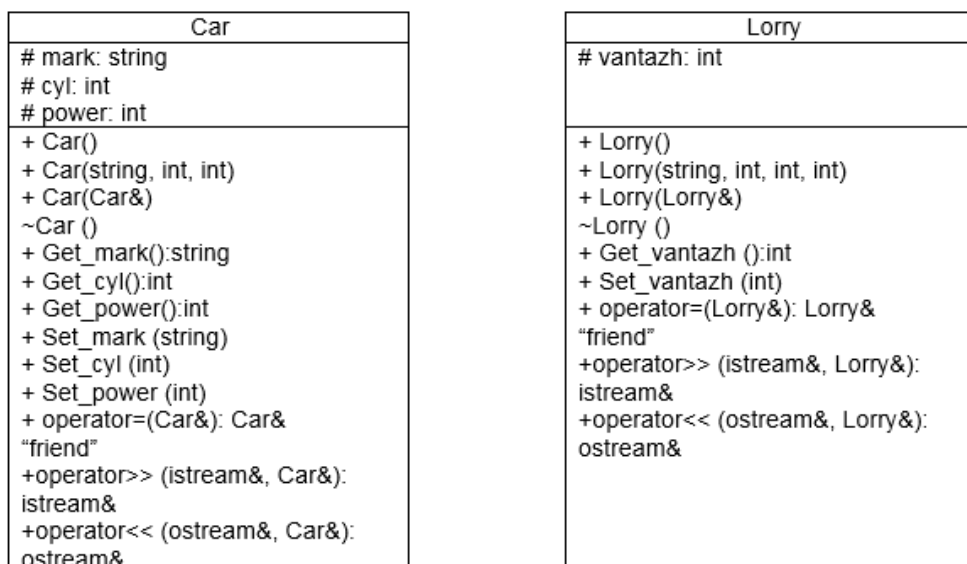


Рис. 19. UML-діаграма класів *Car* та *Lorry*

- Заголовковий файл «Car.h»

```
#ifndef CAR_H_INCLUDED
#define CAR_H_INCLUDED
#include <string>
#include <iostream>

using namespace std;

class Car
{
public:
    Car(); //конструктор без параметрів за замовчуванням.
    ~Car(); //деструктор.
    Car(string, int, int); //конструктор з параметрами.
    Car(const Car&); //конструктор копіювання.
    string Get_mark() {return mark;}
    int Get_cyl() {return cyl;}
    int Get_power() {return power;}
    void Set_mark(string);
    void Set_cyl(int);
    void Set_power(int);
    Car& operator=(const Car&); //перевантаження оператора присвоювання.
```

```
//Перевантаження операторів введення-виведення.  
friend istream& operator>>(istream&in, Car&c);  
friend ostream& operator<<(ostream&out, const Car&c);  
  
protected:  
    string mark;  
    int cyl;  
    int power;  
};  
  
#endif // CAR_H_INCLUDED
```

- Файл із реалізацією класу «Car.cpp»

```
#include "Car.h"  
  
Car::Car()  
{  
    mark="";  
    cyl=0;  
    power=0;  
}  
  
Car::~~Car() {}  
Car::Car(string M, int C, int P)  
{  
    mark=M;  
    cyl=C;  
    power=P;  
}  
  
Car::Car(const Car& car)  
{  
    mark=car.mark;  
    cyl=car.cyl;  
    power=car.power;  
}  
  
void Car::Set_cyl(int C)  
{  
    cyl=C;  
}  
  
void Car::Set_mark(string M)  
{  
    mark=M;  
}  
  
void Car::Set_power(int P)  
{  
    power=P;  
}  
  
Car& Car::operator=(const Car&c)  
{
```

```

    if (&c==this) return *this;
    mark=c.mark;
    power=c.power;
    cyl=c.cyl;
    return *this;
}

istream& operator>>(istream&in, Car&c)
{
    cout<<"\nMark:";
    in>>c.mark;
    cout<<"\nPower:";
    in>>c.power;
    cout<<"\nCyl:";
    in>>c.cyl;
    return in;
}

ostream& operator<<(ostream&out, const Car&c)
{
    out<<"\nMARK : "<<c.mark;
    out<<"\nCYL : "<<c.cyl;
    out<<"\nPOWER : "<<c.power;
    out<<"\n";
    return out;
}

```

- Заголовковий файл «Lorry.h»

```

#ifndef LORRY_H_INCLUDED
#define LORRY_H_INCLUDED

#include "car.h"
//клас Lorry успадковується від класу Car з ключом доступу public.
class Lorry :public Car
{
public:
    Lorry(); //конструктор без параметрів за замовчуванням.
    ~Lorry(); //деструктор.
    Lorry(string, int, int, int); //конструктор з параметрами.
    Lorry(const Lorry & ); //конструктор копіювання.
    int Get_vantazh () {return vantazh;}
    void Set_vantazh (int);
    Lorry& operator=(const Lorry&); //перевантаження оператора присвоювання.
    friend istream& operator>> (istream &in, Lorry&l); //Перевантаження
операторів введення та виведення.
    friend ostream& operator<<(ostream&out, const Lorry&l);
protected:
    int vantazh; //атрибут вантажності.
};
#endif // LORRY_H_INCLUDED

```

- Файл з реалізацією класу «Lorry.cpp»

```

#include "Lorry.h"
Lorry::Lorry():Car()
{
    vantazh =0;
}
Lorry::~~Lorry(){}

Lorry::Lorry(string M, int C, int P, int G):Car(M,C,P)
{
    vantazh =G;
}
Lorry::Lorry(const Lorry &L)
{
    mark=L.mark;
    cyl=L.cyl;
    power=L.power;
    vantazh =L.vantazh;
}
void Lorry::Set_vantazh (int G)
{
    vantazh =G;
}
Lorry& Lorry::operator=(const Lorry&l)
{
    if(&l==this) return *this;
    mark=l.mark;
    power=l.power;
    cyl=l.cyl;
    vantazh =l.vantazh;
    return *this;
}
istream& operator>>(istream&in,Lorry&l)
{
    cout<<"\nMark:";
    in>>l.mark;
    cout<<"\nPower:";
    in>>l.power;
    cout<<"\nCyl:";
    in>>l.cyl;
    cout<<"\nVantazh:";
    in>>l.vantazh;
    return in;
}
ostream& operator<<(ostream&out,const Lorry&l)
{
    out<<"\nMARK : "<<l.mark;
    out<<"\nCYL : "<<l.cyl;
    out<<"\nPOWER : "<<l.power;
    out<<"\nVANTAZH: "<<l.vantazh;
    out<<"\n";
    return out;
}

```

- Файл «main.cpp» з прикладом використання розроблених класів

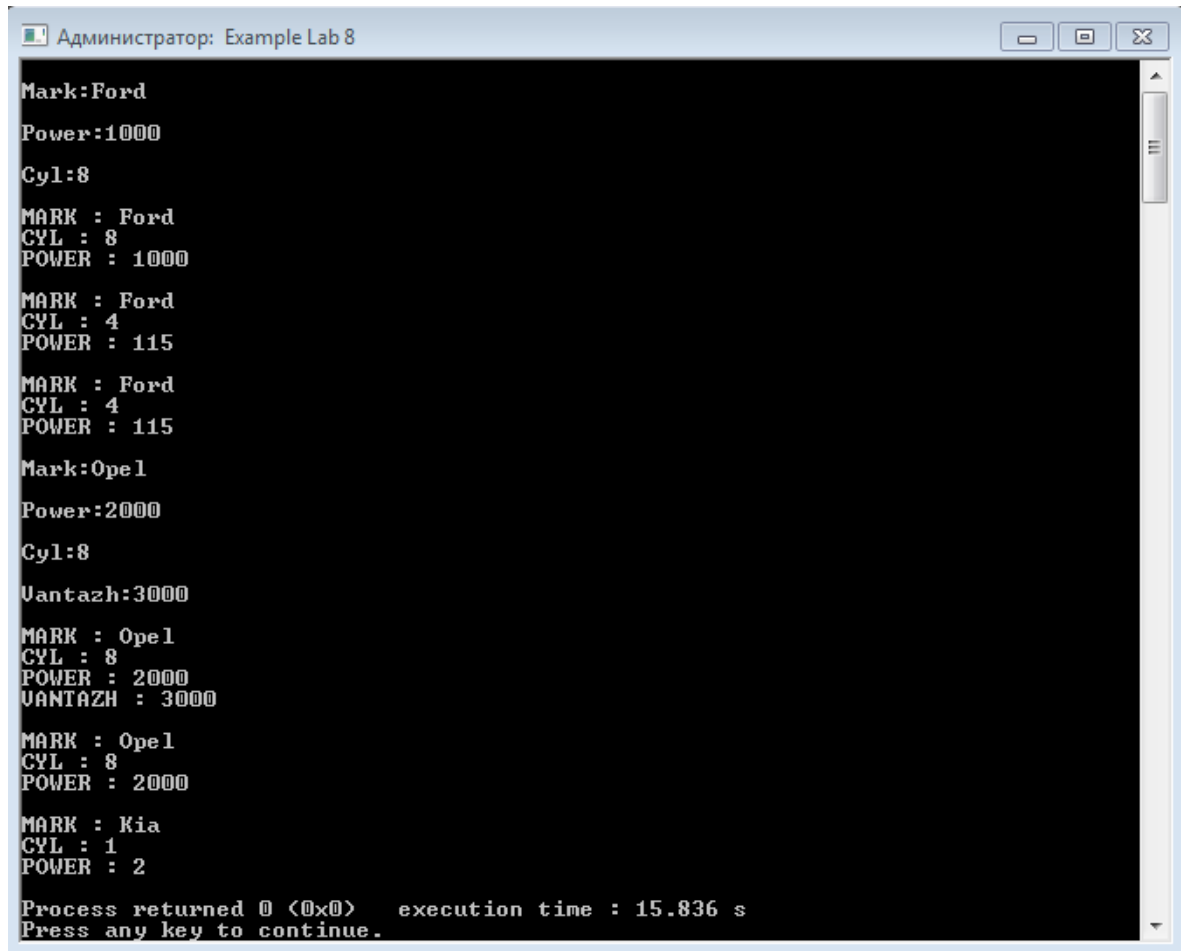
```
#include <iostream>
#include "Car.h"
#include "Lorry.h"

using namespace std;

void f1 (Car&c)
{
    c.Set_mark ("Opel");
    cout<<c;
}
Car f2 ()
{
    Lorry l ("Kia", 1, 2, 3);
    return l;
}

int main ()
{
    //Робота з класом Car.
    Car a;
    cin>>a;
    cout<<a;
    Car b ("Ford", 4, 115);
    cout<<b;
    a=b;
    cout<<a;
    //Робота з класом Lorry.
    Lorry c;
    cin>>c;
    cout<<c;
    //Принцип підстановки.
    f1 (c) ; //передаємо об'єкт класу Lorry.
    a=f2 () ; //створюємо в функції об'єкт класу Lorry.
    cout<<a;
}
```

Приклад виконання програми наведено на рис. 20.



```
Адміністратор: Example Lab 8
Mark:Ford
Power:1000
Cyl:8
MARK : Ford
CYL : 8
POWER : 1000
MARK : Ford
CYL : 4
POWER : 115
MARK : Ford
CYL : 4
POWER : 115
Mark:Opel
Power:2000
Cyl:8
Uantazh:3000
MARK : Opel
CYL : 8
POWER : 2000
UANTAZH : 3000
MARK : Opel
CYL : 8
POWER : 2000
MARK : Kia
CYL : 1
POWER : 2
Process returned 0 (0x0)   execution time : 15.836 s
Press any key to continue.
```

Рис. 20. Приклад виконання програми

7.4. Завдання до лабораторної роботи.

Хід виконання завдання:

- 1) Створити клас для заданого поняття (базовий клас) з відповідними полями та методами.
- 2) Створити похідний клас.
- 3) Реалізувати функцію, яка одержує та повертає об'єкти базового класу, продемонструвати принцип підстановки.

Варіанти завдань

- 1) Створити клас *Man* (чоловік), із полями: ім'я, вік, стать і вага. Визначити методи перепризначення імені, зміни віку і зміни ваги. Створити похідний клас *Employee*, що має поля – посада та оклад. Визначити методи зміни полів

- 2) Створити клас *Pair* (пара чисел); визначити методи зміни полів і порівняння пар: пара $p1$ більша від пари $p2$, якщо $(p1.first > p2.first)$ або $(p1.first = p2.first)$ і $(p1.second > p2.second)$. Визначити клас-спадкоємець *Fraction* із полями: ціла частина числа і дрібна частина числа. Визначити повний набір методів порівняння.
- 3) Створити клас *Liquid* (рідина), який має поля назви і щільності. Визначити методи перепризначення і зміни щільності. Створити похідний клас *Alcohol* (спирт), який має міцність. Визначити методи перепризначення і зміни міцності.
- 4) Створити клас *Pair* (пара чисел); визначити методи зміни полів і обчислення добутку чисел. Визначити похідний клас *Rectangle* (прямокутник) з полями-сторонами. Визначити методи обчислення периметра та площі прямокутника.
- 5) Створити клас *Triad* (трійка чисел); визначити методи зміни полів і обчислення суми чисел. Визначити похідний клас *Triangle* з полями-сторонами. Визначити методи обчислення кутів і площі трикутника.
- 6) Створити клас *Liquid* (рідина), який має поля назви і щільності. Визначити методи перепризначення і зміни щільності. Створити похідний клас *Kerosene* (керосин), який має температуру кипіння та в'язкість. Визначити методи перепризначення і зміни температури кипіння та в'язкості.
- 7) Створити клас *Man* (чоловік), із полями: ім'я, вік, стать і вага. Визначити методи перепризначення імені, зміни віку та зміни ваги. Створити похідний клас *Student*, що має поле року навчання. Визначити методи перепризначення і збільшення року навчання.
- 8) Створити клас *Triangle* з полями-сторонами. Визначити методи зміни сторін, обчислення кутів, обчислення периметра. Створити похідний клас *Equilateral* (рівносторонній), що має поле площі. Визначити метод обчислення площі.
- 9) Створити клас *Liquid* (рідина), який має поля назви і щільності. Визначити методи перепризначення і зміни щільності. Створити похідний клас *Milk* (молоко), який має температуру кипіння. Визначити методи перепризначення і зміни температури кипіння.
- 10) Створити клас *Liquid* (рідина), який має поля назви і щільності. Визначити

методи перепризначення і зміни щільності. Створити похідний клас *Glycerol* (гліцерин), який має температуру кипіння та плавлення. Визначити методи перепризначення і зміни температури кипіння та плавлення.

- 11) Створити клас *Triangle* з полями-сторонами. Визначити методи зміни сторін, обчислення кутів, обчислення периметра. Створити похідний клас *RightAngled* (прямокутний), який має поле площі. Визначити метод обчислення площі.
- 12) Створити клас *Pair* (пара чисел); визначити методи зміни полів і обчислення добутку чисел. Визначити похідний клас *RightAngled* з полями-катетами. Визначити методи обчислення гіпотенузи і площі трикутника.
- 13) Створити клас *Triad* (трійка чисел); визначити метод порівняння тріад. Визначити похідний клас *Time* з полями: година, хвилина та секунда. Визначити повний набір методів порівняння моментів часу.
- 14) Створити клас *Triad* (трійка чисел); визначити методи збільшення полів на 1. Визначити клас-спадкоємець *Time* з полями: година, хвилина, секунда. Перевизначити методи збільшення полів на 1 і визначити методи збільшення на n секунд і хвилин.
- 15) Створити базовий клас *Pair* (пара цілих чисел) з операціями перевірки на рівність і перемноження полів. Реалізувати операцію віднімання пар за формулою $(a, b) - (c, d) = (a - b, c - d)$. Створити похідний клас *Rational*; визначити нові операції додавання $(a, b) + (c, d) = (ad + bc, bd)$ і ділення $(a, b) / (c, d) = (ad, bc)$; перевизначити операцію віднімання $(a, b) - (c, d) = (ad - bc, bd)$.
- 16) Створити клас *Liquid* (рідина), який має поля назви і щільності. Визначити методи перепризначення і зміни щільності. Створити похідний клас *Petrol* (бензин), який має температуру кипіння та замерзання. Визначити методи перепризначення і зміни температури кипіння та замерзання.
- 17) Створити клас *Pair* (пара цілих чисел); визначити методи зміни полів та операцію складання пар $(a, b) + (c, d) = (a + b, c + d)$. Визначити клас-спадкоємець *Long* із полями: старша частина числа та молодша частина числа. Перевизначити операцію складання і визначити методи множення та віднімання.
- 18) Створити базовий клас *Triad* (трійка чисел) із операціями додавання числа,

множення на число, перевірки на рівність. Створити похідний клас *vector3D*, що задається трійкою координат. Повинні бути реалізовані: операція додавання векторів, скалярний добуток векторів.

- 19) Створити клас *Pair* (пара чисел); визначити метод перемноження полів та операцію складання пар $(a, b) + (c, d) = (a + b, c + d)$. Визначити похідний клас *Complex* із полями: дійсна й уявна частини числа. Визначити методи множення $(a, b) * (c, d) = (ac - bd, ad + bc)$ і віднімання $(a, b) - (c, d) = (a - b, c - d)$.
- 20) Створити клас *Pair* (пара цілих чисел); визначити метод множення на число й операцію складання пар $(a, b) + (c, d) = (a + b, c + d)$. Визначити клас-спадкоємець *Money* з полями: гривні та копійки. Перевизначити операцію складання та визначити методи вирахування і розподілу грошових сум.
- 21) Створити клас *Man* (чоловік), із полями: ім'я, вік, стать. Визначити методи перепризначення імені, зміни віку. Створити похідний клас *Teacher*, що має поля: предмет і кількість годин. Визначити методи зміни полів, а також збільшення та зменшення годин.
- 22) Створити клас *Liquid* (рідина), який має поля назви і щільності. Визначити методи перепризначення і зміни щільності. Створити похідний клас *Milk* (молоко), який має температуру кипіння. Визначити методи перепризначення і зміни температури кипіння.
- 23) Створити клас *Man* (чоловік), із полями: ім'я, вік, стать, вага. Визначити методи перепризначення імені, зміни віку, ваги. Створити похідний клас *Runner* (бегун), що має поля: тип дистанції, пульс. Визначити методи зміни полів.

7.5. Контрольні запитання

- 1) Поясніть поняття успадкування в контексті ООП? Для чого застосовується успадкування класів?
- 2) Які види успадкування Вам відомі?
- 3) Що таке ключ доступу при успадкуванні класів? На що він впливає?
- 4) Як викликаються конструктори при успадкуванні? Чи успадковуються конструктори?

- 5) Як можна передати параметри конструктору базового класу через похідний?
- 6) Як викликаються деструктори при успадкуванні? Чи успадковуються деструктори?
- 7) Поясніть принцип підстановки.

Посилання

1) Основи програмування на C++: Навчальний посібник для студентів спеціальностей 113 – Прикладна математика та 122 – Комп'ютерні науки: навч. посіб./ Водка О.О., Дашкевич А.О., Іванченко К.В., Розова Л.В., Сенько А.В. – Харків: НТУ ХП», 2021. – 114 с.