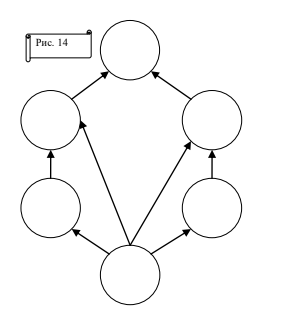
**Лабораторна робота N6 (2022).**

**Тема:** Спадкування. Віртуальні класи. Віртуальні функції. Абстрактні класи.Методичні вказівки

**Завдання 1**

Задача. Створити дві ієрархії класів з віртуальним та без віртуального успадкуванням з елементами даних класів у кожному класі. Схема успадкування на рисунку за варіантами. Створити об’єкти похідних класів з віртуальним та без віртуального успадкуванням. Вивести розміри об’єктів даних класів.

****

**Код:**

**#include <iostream>**

**using namespace std;**

**// Без віртуального успадкування**

**class A1 { int a1; };**

**class B1 : public A1 { int b1; };**

**class C1 : public A1 { int c1; };**

**class D1 : public B1, public C1 { int d1; };**

**// З віртуальним успадкуванням**

**class A2 { int a2; };**

**class B2 : virtual public A2 { int b2; };**

**class C2 : virtual public A2 { int c2; };**

**class D2 : public B2, public C2 { int d2; };**

**int main() {**

**cout << "Без віртуального успадкування:\n";**

**cout << "Size of A1: " << sizeof(A1) << endl;**

**cout << "Size of B1: " << sizeof(B1) << endl;**

**cout << "Size of C1: " << sizeof(C1) << endl;**

**cout << "Size of D1: " << sizeof(D1) << endl;**

**cout << "\nЗ віртуальним успадкуванням:\n";**

**cout << "Size of A2: " << sizeof(A2) << endl;**

**cout << "Size of B2: " << sizeof(B2) << endl;**

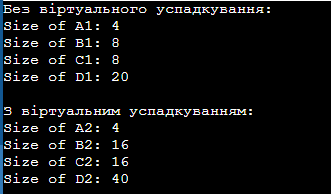
**cout << "Size of C2: " << sizeof(C2) << endl;**

**cout << "Size of D2: " << sizeof(D2) << endl;**

**return 0;**

**}**

**Скрін:**

****

**Завдання 2**

**Задача 2.4. Створити абстрактний клас функція із віртуальною функцією обчислення значення з координатою x. Створити похідні класи: пряма, еліпс, гіпербола зі своїми функціями обчислення значення за координатою x. Рівняння прямої: y=ax+b, еліпса:x2/a2+y2/b2=1, гіперболи: x2/a2-y2/b2=1.**

**Код:**

**#include <iostream>**

**#include <cmath>**

**#include <iomanip>**

**using namespace std;**

**class Function {**

**public:**

**virtual double calculate(double x) const = 0;**

**virtual ~Function() {}**

**};**

**// Пряма: y = ax + b**

**class Line : public Function {**

**double a, b;**

**public:**

**Line(double a, double b) : a(a), b(b) {}**

**double calculate(double x) const override {**

**return a \* x + b;**

**}**

**};**

**// Еліпс: x^2/a^2 + y^2/b^2 = 1 -> y = ±b \* sqrt(1 - x^2/a^2)**

**class Ellipse : public Function {**

**double a, b;**

**public:**

**Ellipse(double a, double b) : a(a), b(b) {}**

**double calculate(double x) const override {**

**if (abs(x) > a) {**

**throw domain\_error("x поза межами еліпса");**

**}**

**return b \* sqrt(1 - (x \* x) / (a \* a));**

**}**

**};**

**// Гіпербола: x^2/a^2 - y^2/b^2 = 1 -> y = ±b \* sqrt(x^2/a^2 - 1)**

**class Hyperbola : public Function {**

**double a, b;**

**public:**

**Hyperbola(double a, double b) : a(a), b(b) {}**

**double calculate(double x) const override {**

**if (abs(x) < a) {**

**throw domain\_error("x в межах області розриву гіперболи");**

**}**

**return b \* sqrt((x \* x) / (a \* a) - 1);**

**}**

**};**

**int main() {**

**Line line(2, 3);**

**Ellipse ellipse(5, 4);**

**Hyperbola hyperbola(3, 2);**

**double x = 2;**

**cout << fixed << setprecision(3);**

**cout << "Line: y = " << line.calculate(x) << endl;**

**try {**

**cout << "Ellipse: y = " << ellipse.calculate(x) << endl;**

**} catch (const exception &e) {**

**cout << "Ellipse: " << e.what() << endl;**

**}**

**try {**

**cout << "Hyperbola: y = " << hyperbola.calculate(x) << endl;**

**} catch (const exception &e) {**

**cout << "Hyperbola: " << e.what() << endl;**

**}**

**return 0;**

**}**

**Скрін:**

****

**Завдання 3**

**Задача 3.4. Ієрархія типів складається з сутностей: працівник, батько-сімейства, і**

**працівник-батько сімейства.  
Код:  
#include <iostream>**

**#include <string>**

**using namespace std;**

**class Працівник {**

**protected:**

**string ім\_я;**

**string посада;**

**double зарплата;**

**public:**

**Працівник() : ім\_я(""), посада(""), зарплата(0) {}**

**Працівник(string ім\_я, string посада, double зарплата) :**

**ім\_я(ім\_я), посада(посада), зарплата(зарплата) {}**

**virtual ~Працівник() {}**

**virtual void print() const {**

**cout << "Ім'я: " << ім\_я << ", Посада: " << посада << ", Зарплата: " << зарплата << endl;**

**}**

**friend istream& operator>>(istream& in, Працівник& p) {**

**cout << "Ім'я: "; in >> p.ім\_я;**

**cout << "Посада: "; in >> p.посада;**

**cout << "Зарплата: "; in >> p.зарплата;**

**return in;**

**}**

**friend ostream& operator<<(ostream& out, const Працівник& p) {**

**out << "Ім'я: " << p.ім\_я << ", Посада: " << p.посада << ", Зарплата: " << p.зарплата;**

**return out;**

**}**

**};**

**class БатькоСімейства {**

**protected:**

**int кількість\_дітей;**

**string прізвище;**

**public:**

**БатькоСімейства() : кількість\_дітей(0), прізвище("") {}**

**БатькоСімейства(int діти, string прізвище) :**

**кількість\_дітей(діти), прізвище(прізвище) {}**

**virtual ~БатькоСімейства() {}**

**virtual void print() const {**

**cout << "Прізвище: " << прізвище << ", Кількість дітей: " << кількість\_дітей << endl;**

**}**

**friend istream& operator>>(istream& in, БатькоСімейства& b) {**

**cout << "Прізвище: "; in >> b.прізвище;**

**cout << "Кількість дітей: "; in >> b.кількість\_дітей;**

**return in;**

**}**

**friend ostream& operator<<(ostream& out, const БатькоСімейства& b) {**

**out << "Прізвище: " << b.прізвище << ", Кількість дітей: " << b.кількість\_дітей;**

**return out;**

**}**

**};**

**class ПрацівникБатькоСімейства : public Працівник, public БатькоСімейства {**

**public:**

**ПрацівникБатькоСімейства() {}**

**ПрацівникБатькоСімейства(string ім\_я, string посада, double зарплата,**

**int діти, string прізвище)**

**: Працівник(ім\_я, посада, зарплата),**

**БатькоСімейства(діти, прізвище) {}**

**void print() const override {**

**Працівник::print();**

**БатькоСімейства::print();**

**}**

**friend istream& operator>>(istream& in, ПрацівникБатькоСімейства& pbs) {**

**in >> static\_cast<Працівник&>(pbs);**

**in >> static\_cast<БатькоСімейства&>(pbs);**

**return in;**

**}**

**friend ostream& operator<<(ostream& out, const ПрацівникБатькоСімейства& pbs) {**

**out << static\_cast<const Працівник&>(pbs) << endl;**

**out << static\_cast<const БатькоСімейства&>(pbs);**

**return out;**

**}**

**virtual ~ПрацівникБатькоСімейства() {}**

**};**

**int main() {**

**ПрацівникБатькоСімейства p;**

**cin >> p;**

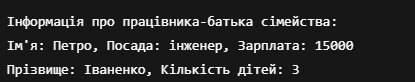
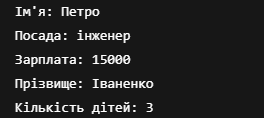
**cout << "\nІнформація про працівника-батька сімейства:\n";**

**cout << p;**

**return 0;**

**}**

**Скрін:**

****