**Практическая работа №6**

**РЕШЕНИЕ ЗАДАЧ НА НАСЛЕДОВАНИЕ В ЯЗЫКЕ С++**

**Цель:** Целью данной практической работы является приобретение практических навыков использования принципа ООП – наследования для разработки программ на языке программирования C++.

**Задания:**

1. Написать иерархию классов, описывающих имущество налогоплательщиков. Она должна состоять из абстракт- ного базового класса Property и производных от него классов Appartment, Car и CountryHouse. Базовый класс должен иметь поле worth (стоимость), конструктор с одним параметром, заполняющий это поле, и чисто виртуальный метод рассчета налога, переопределенный в каждом из производных классов. Налог на квартиру вычисляется как 1/1000 ее стоимости, на машину — 1/200, на дачу — 1/500. Также, каждый производный класс должен иметь конструктор с одним параметром, передающий свой параметр конструктору базового класса. В функции main завести массив из 7 указателей на Property и заполнить его указателями на динамические объекты производных классов (первые 3 — Appartment, следующие 2 — Car и последние 2 — CountryHouse). Вывести на экран величину налога для всех 7 объектов. Не забудьте также уничтожить динамические объекты перед завершением программы.

2. Написать набор классов, представляющий выражения. В этом наборе должен быть один абстрактный базо- вый тип, а также набор производных от него типов по видам выражений (константа, переменная, сумма, разность, произведение, частное, sin, cos, exp, ln). У каждого из классов должны быть следующие виртуальные функции: напе- чатать выражение (без параметров), вычислить выражение (параметр — значение переменной, результат — значение выражения), вернуть производную выражения (без параметров), создать копию выражения (тоже без параметров).

**Задание №1**

#include "pch.h"

#include <iostream>

#include "windows.h"

using namespace std;

class Property //класс Собственность

{

protected:

double worth;

public:

Property(double worth) //конструктор

{

this->worth = worth;

}

virtual double calculateTax() = 0;//виртуальный метод рассчета налога

virtual ~Property() //деструктор

{

}

};

class Appartment : public Property //класс Квартира

{

public:

Appartment(double worth) : Property(worth) {};

double calculateTax()

{

return worth / 1000;

}

};

class Car : public Property //класс Автомобиль

{

public:

Car(double worth) : Property(worth) {};

double calculateTax()

{

return worth / 200;

}

};

class CountryHouse : public Property //класс Загородный дом

{

public:

CountryHouse(double worth) : Property(worth) {};

double calculateTax()

{

return worth / 500;

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("color 0A");

Property \*x[7];

for (int i = 0; i < 7; i++)

{

double worth;

cout << "Введите стоимость " << i + 1 << " : ";

cin >> worth;

if (i < 3)

{

x[i] = new Appartment(worth);

}

else

{

if (i < 5)

{

x[i] = new Car(worth);

}

else

{

x[i] = new CountryHouse(worth);

}

}

}

cout << "Налоги:" << endl;

for (int i = 0; i < 7; i++)

{

cout << i + 1 << ". " << x[i]->calculateTax() << endl;

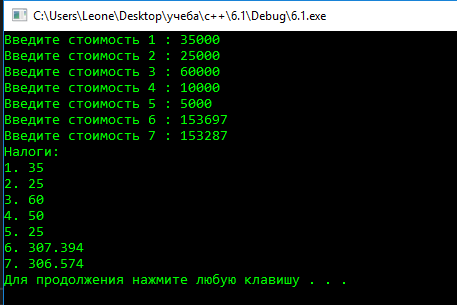
}

system("pause");

delete[] x;

return 0;

}



**Задание №2**

#include "pch.h"

#include <iostream>

#include <cmath>

#include "windows.h"

using namespace std;

class Function // Базовый класс функция

{

public:

virtual void printFunction() = 0; // Выводит функцию на экран

virtual double count(double x) = 0; // Считает значение функции от аргумента

virtual void takeDerivative() = 0; // Выводит производную

};

class Const : public Function // Класс, описывающий константу

{

double c;

public:

Const(double c) { this->c = c; }

void printFunction() { cout << "Функция: f(x) = " << c << endl; }

double count(double x) { return c; }

void takeDerivative() { cout << "Производная: 0" << endl; }

};

class Variable : public Function { // Класс, описывающий переменную

public:

void printFunction() { cout << "Функция: f(x) = x" << endl; }

double count(double x) { return x; }

void takeDerivative() { cout << "Производная: 1" << endl; }

};

class Sum : public Function { // Класс, описывающий сумму двух чисел

double y;

public:

Sum(double y) { this->y = y; }

void printFunction() { cout << "Функция: f(x) = x + y" << endl; }

double count(double x) { return x + y; }

void takeDerivative() { cout << "Производная: 2" << endl; }

};

class Difference : public Function { // Класс, описывающий разность двух чисел

double y;

public:

Difference(double y) { this->y = y; }

void printFunction() { cout << "Функция: f(x) = x - y" << endl; }

double count(double x) { return x - y; }

void takeDerivative() { cout << "Производная: 0" << endl; }

};

class Product : public Function { // Класс, описывающий произведение двух чисел

double y;

public:

Product(double y) { this->y = y; }

void printFunction() { cout << "Функция: f(x) = x \* y" << endl; }

double count(double x) { return x \* y; }

void takeDerivative() { cout << "Производная: y \* dx" << endl; }

};

class Quotient : public Function { // Класс, описывающий частное двух чисел

double y;

public:

Quotient(double y) { this->y = y; }

void printFunction() { cout << "Функция: f(x) = x / y" << endl; }

double count(double x) { return x / y; }

void takeDerivative() { cout << "Производная: dx / y" << endl; }

};

class Sin : public Function { // Класс, описывающий синус угла в радианах

public:

void printFunction() { cout << "Функция: f(x) = sin(x)" << endl; }

double count(double x) { return sin(x); }

void takeDerivative() { cout << "Производная: cos(x)" << endl; }

};

class Cos : public Function { // Класс, описывающий косинус угла в радианах

public:

void printFunction() { cout << "Функция: f(x) = cos(x)" << endl; }

double count(double x) { return cos(x); }

void takeDerivative() { cout << "Производная: -sin(x)" << endl; }

};

class Exp : public Function { // Класс, описывающий возведение экспоненты в степень

public:

void printFunction() { cout << "Функция: f(x) = e^x" << endl; }

double count(double x) { return exp(x); }

void takeDerivative() { cout << "Производная: e^x" << endl; }

};

class Ln : public Function { // Класс, описывающий взятие натурального логарифма

public:

void printFunction() { cout << "Функция: f(x) = ln(x)" << endl; }

double count(double x) { return log(x); }

void takeDerivative() { cout << "Производная: 1/x" << endl; }

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

system("color 0A");

double y, x;

cout << "Константа" << endl << "Введите значение: ";

cin >> y;

Const c(y);

cout << "Введите аргумент: ";

cin >> x;

c.printFunction();

cout << "f(" << x << ") = " << c.count(x) << endl;

c.takeDerivative();

cout << endl << "Переменная" << endl << "Введите аргумент: ";

cin >> x;

Variable v;

v.printFunction();

cout << "f(" << x << ") = " << v.count(x) << endl;

v.takeDerivative();

cout << endl << "Сумма" << endl << "Введите x и y: ";

cin >> x >> y;

Sum s(y);

s.printFunction();

cout << "f(" << x << ") = " << s.count(x) << endl;

s.takeDerivative();

cout << endl << "Разница" << endl << "Введите x и y: ";

cin >> x >> y;

Difference d(y);

d.printFunction();

cout << "f(" << x << ") = " << d.count(x) << endl;

d.takeDerivative();

cout << endl << "Произведение" << endl << "Введите x и y: ";

cin >> x >> y;

Product p(y);

p.printFunction();

cout << "f(" << x << ") = " << p.count(x) << endl;

p.takeDerivative();

cout << endl << "Частное" << endl << "Введите x и y: ";

cin >> x >> y;

Quotient q(y);

q.printFunction();

cout << "f(" << x << ") = " << q.count(x) << endl;

q.takeDerivative();

cout << endl << "Sin" << endl << "Введите аргумент: ";

cin >> x;

Sin si;

si.printFunction();

cout << "f(" << x << ") = " << si.count(x) << endl;

si.takeDerivative();

cout << endl << "Cos" << endl << "Введите аргумент: ";

cin >> x;

Cos co;

co.printFunction();

cout << "f(" << x << ") = " << co.count(x) << endl;

co.takeDerivative();

cout << endl << "Exp" << endl << "Введите аргумент: ";

cin >> x;

Exp e;

e.printFunction();

cout << "f(" << x << ") = " << e.count(x) << endl;

e.takeDerivative();

cout << endl << "Ln" << endl << "Введите аргумент: ";

cin >> x;

Ln l;

l.printFunction();

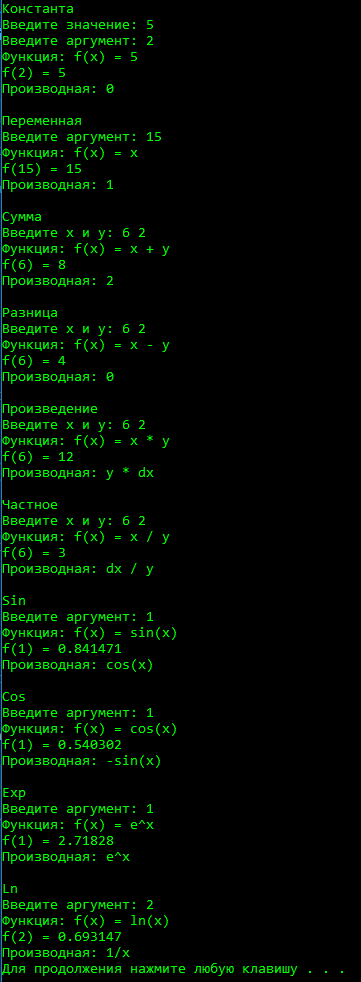
cout << "f(" << x << ") = " << l.count(x) << endl;

l.takeDerivative();

system("pause");

return 0;

}



**Вывод:** В ходе выполнения практической работы №6 было реализовано наследование классов и продемонстрированы виртуальные функции.