Лабораторная работа 6

**ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ОТНОШЕНИЯ**

**МНОЖЕСТВЕННОГО НАСЛЕДОВАНИЯ**

Цель работы: изучить принципы и механизмы множественного наследования, правила доступа к базовым классам; приобрести практические навыки работы с базовыми и производными классами при множественном наследовании.

**Задания**

1. Используя предыдущую программу, создайте новый производный класс с применением множественного наследования.

2. Проверьте работоспособность АТД и производных классов на тестовом наборе данных.

**Код**

#include <iostream>

#include <stdio.h>

#include <string.h>

using namespace std;

template <class T1 = int, class T2 = float>

class Software {

public:

Software<T1, T2>(const char\* soft, T1 c, T2 s);

Software<T1, T2>();

void print();

Software<T1, T2>(const Software& a);

bool operator==(const Software& a);

Software<T1, T2> operator+(const Software& a);

Software<T1, T2>& operator++();

Software<T1, T2>& operator=(const Software& a);

~Software();

protected:

char\* programm; T1 cost; T2 size;

};

template <class T1, class T2>

Software<T1, T2>::Software<T1, T2>() {

programm = new char[strlen("Неизвестно") + 1];

strcpy(programm, "Неизвестно");

cost = 0;

size = 0;

};

template <class T1, class T2>

Software<T1, T2>::Software<T1, T2>(const char\* soft, T1 c, T2 s) {

programm = new char[strlen(soft) + 1];

strcpy(programm, soft);

cost = c;

size = s;

};

template <class T1, class T2>

void Software<T1, T2>::print() {

cout << "Программа-" << programm << endl;

cout << "Стоимость-" << cost << endl;

cout << "Размер- " << size << "Мб" << endl << endl;

};

template <class T1, class T2>

Software<T1, T2>::Software<T1, T2>(const Software& a) {

programm = new char[strlen(a.programm) + 1];

strcpy(programm, a.programm);

cost = a.cost;

size = a.size;

};

template <class T1, class T2>

bool Software<T1, T2>::operator==(const Software& a) {

return ((!strcmp(programm, a.programm)) &&

cost == a.cost &&

size == a.size);

};

template <class T1, class T2>

Software<T1, T2>::~Software<T1, T2>() {

delete[] programm;

};

template <class T1, class T2>

Software<T1, T2> Software<T1, T2>::operator+(const Software & a) {

Software<T1, T2> v;

v.programm = new char[strlen(a.programm) + strlen(programm) + 2];

strcpy(v.programm, programm);

strcat(v.programm, ",");

strcat(v.programm, a.programm);

v.cost = cost + a.cost;

v.size = size + a.size;

return v;

};

template <class T1, class T2>

Software<T1, T2>& Software<T1, T2>::operator=(const Software & a) {

programm = new char[strlen(a.programm) + 1];

strcpy(programm, a.programm);

cost = a.cost;

size = a.size;

return(\*this);

};

template <class T1, class T2>

Software<T1, T2>& Software<T1, T2>::operator++() {

cost += 1;

size += 1;

return(\*this);

};

//Второй базовый класс

template <class T1 = int, class T2 = float>

class BaseСlassSoftware {

public:

BaseСlassSoftware<T1, T2>(int type);

BaseСlassSoftware<T1, T2>();

void printClass();

protected:

int typeClass;

};

template <class T1, class T2>

BaseСlassSoftware<T1, T2>::BaseСlassSoftware<T1, T2>() {

typeClass = 1;

};

template <class T1, class T2>

BaseСlassSoftware<T1, T2>::BaseСlassSoftware<T1, T2>(int type) {

typeClass = type;

};

template <class T1, class T2>

void BaseСlassSoftware<T1, T2>::printClass() {

if (typeClass == 1)cout << "Класс программного обеспечения-Системное ПО" << endl;

if (typeClass == 2)cout << "Класс программного обеспечения-Прикладное ПО" << endl;

if (typeClass == 3)cout << "Класс программного обеспечения-Системы программирования" << endl;

};

//производный класс с множественым наследованием

template <class T1 = int, class T2 = float>

class Apply : public Software<T1, T2>, public BaseСlassSoftware<T1, T2>

{

public:

Apply<T1, T2>(const char\* soft, T1 c, T2 s, int type, char\* spec);

Apply<T1, T2>();

void printApply();

~Apply();

protected:

char\* specialization;

};

template <class T1, class T2>

Apply<T1, T2>::Apply<T1, T2>() : Software<T1, T2>(), BaseСlassSoftware<T1, T2>(){

specialization = new char[strlen("неизвестно") + 1];

strcpy(specialization, "неизвестно");

};

template <class T1, class T2>

Apply<T1, T2>::Apply<T1, T2>(const char\* soft, T1 c, T2 s, int type, char\* spec) : Software<T1, T2>(soft, c, s), BaseСlassSoftware<T1, T2>(type){

specialization = new char[strlen(spec) + 1];

strcpy(specialization, spec);

};

template <class T1, class T2>

void Apply<T1, T2>::printApply() {

cout << "Вывод " << endl;

Software<T1, T2>::print();

BaseСlassSoftware<T1, T2>::printClass();

cout << "Данное ПО специализируется на - " << specialization << endl;

};

template <class T1, class T2>

Apply<T1, T2>::~Apply() {

delete[] specialization;

};

int main()

{

setlocale(LC\_ALL, "Russian");

char\* soft = new char[10];

char\* specialization = new char[10];

double cost, size;

int type;

cout << "Введите название программы ";

cin >> soft;

cout << endl << "Введите стоимость программы ";

cin >> cost;

cout << endl << "Введите размер программы ";

cin >> size;

cout << endl << "Введите классификацию программы ";

cin >> type;

cout << endl << "базовые классы " << endl;

cout << " 1 " << endl;

Software<float, int>x(soft, cost, size);

x.print();

cout << " 2 " << endl;

BaseСlassSoftware<float, int>y(type);

y.printClass();

cout << endl << "производный класс " << endl;

cout << endl << "Введите применение программы: " << endl;

cin >> specialization;

cout << endl;

Apply<float, int>a(soft, cost, size, type, specialization);

a.print();

a.printClass();

a.printApply();

cout << endl << "без параметров" << endl;

Apply<float, int>b;

b.print();

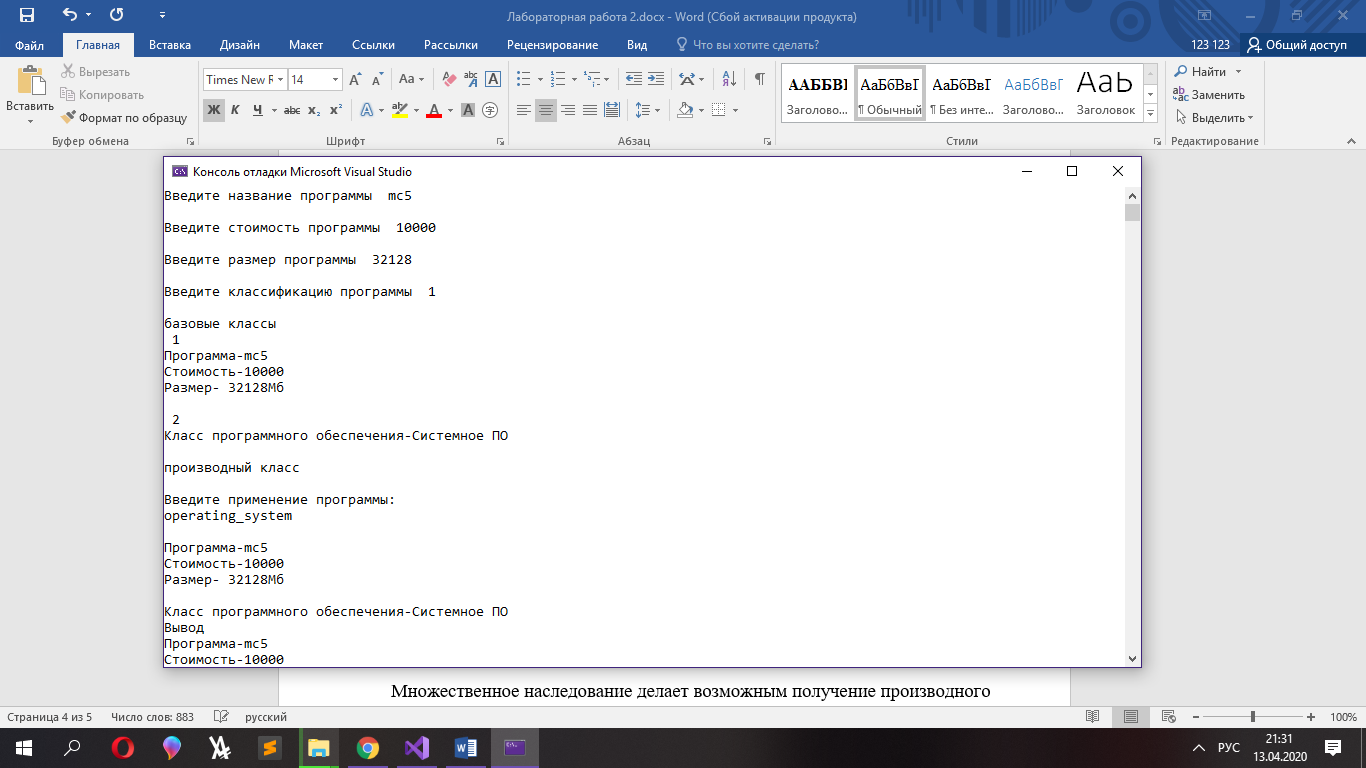
b.printClass();

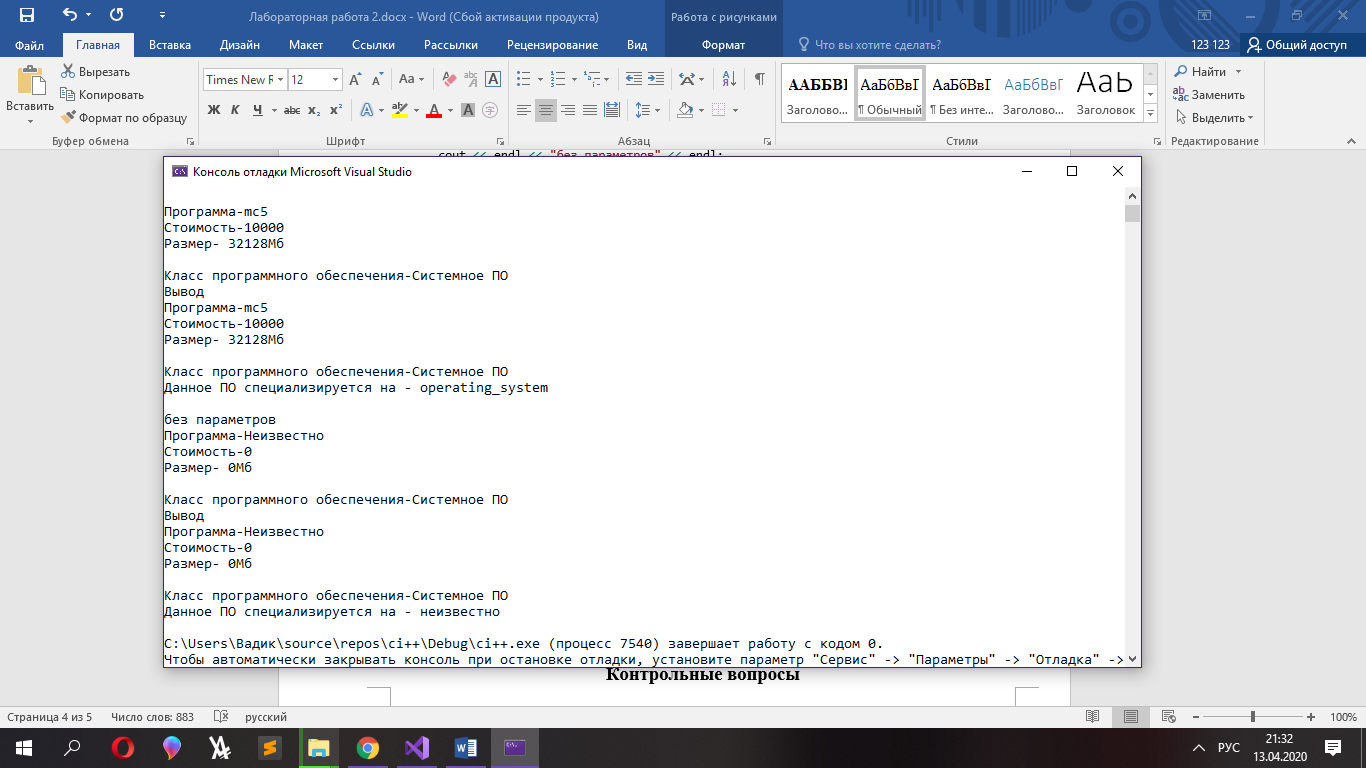
b.printApply();

return 0;

}

**Пример работы**





**Контрольные** **вопросы**

1.С какой целью и в каких случаях используется множественное наследование?

Множественное наследование делает возможным получение производного класса от нескольких базовых классов и позволяет классу перенимать их функциональность.

2.Опишите синтаксис заголовка производного класса при множественном наследовании.

Синтаксис заголовка класса расширяется, чтобы можно было использовать список базовых классов с атрибутами доступа.

class производный: public базовый\_1, private базовый\_2 {…

3.Дайте определение ориентированного ациклического графа.

Это граф, узлы которого являются классами, а ориентированные ребра направлены от производных классов к базовым

4.Где инициируются виртуальные базовые классы?

Виртуальные базовые классы инициализируются перед любыми невиртуальными базовыми классами.

5.Что будет, если из объявлений классов Cow и Buffalo убрать ключевое слово virtual?

Ключевое слово virtual в классе Cow и классе Buffalo предотвращает многократное копирование полей данных weight, price, color из предков класса Beefalo. Так же нельзя будет использовать виртуальные функции.