Лабораторная работа 5

Программная реализация отношения простого наследования

Цель работы: изучить способы создания производного класса и особенности работы с ним, правила инициализации и доступа к элементам производного класса; приобрести практические навыки наследования.

Задания:

1. Создайте производный класс для АТД, реализованного по заданию лабораторной работы 4, используя одиночное наследование.

2. Проверьте работоспособность АТД и производного класса на тестовом наборе данных.

Код программы:

#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);

};

//Производный класс

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

class PKSoftware : public Software<T1, T2> {

public:

PKSoftware<T1, T2>();

PKSoftware<T1, T2>(T1 stars, bool license, char\* comment, const char\* soft, T1 c, T2 s);

~PKSoftware<T1, T2>();

void conclusion();

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

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

protected:

T1 stars; bool license; char\* comment;

};

template <class T1, class T2>

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

stars = 0;

license = false;

comment = new char[strlen("не добавлено") + 1];

strcpy(comment, "не добавлено");

};

template <class T1, class T2>

PKSoftware<T1, T2>::PKSoftware<T1, T2>(T1 stars, bool license, char\* comment, const char\* soft, T1 c, T2 s) : Software<T1, T2>(soft, c, s) {

this->stars = stars;

this->license = license;

this->comment = new char[strlen(comment) + 1];

strcpy(this->comment, comment);

};

template <class T1, class T2>

void PKSoftware<T1, T2>::conclusion() {

cout << "Вывод " << endl;

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

cout << endl << "Рейтинг-" << stars << endl;

cout << "Лицензия-";

if (license) cout << "Есть" << endl;

else cout << "Нет" << endl;

cout << "Комментарий- " << comment << endl;

};

template <class T1, class T2>

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

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

strcpy(comment, a.comment);

stars = a.stars;

license = a.license;

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

strcpy(this->programm, a.programm);

this->cost = a.cost;

this->size = a.size;

};

template <class T1, class T2>

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

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

strcpy(comment, a.comment);

stars = a.stars;

license = a.license;

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

strcpy(this->programm, a.programm);

this->cost = a.cost;

this->size = a.size;

return(\*this);

};

template <class T1, class T2>

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

delete[] comment;

};

int main()

{

setlocale(LC\_ALL, "Russian");

char\* soft = new char[10];

char\* comment = new char[10];

double cost, size, stars;

bool license;

cout << "Введите название программы ";

cin >> soft;

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

cin >> cost;

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

cin >> size;

cout << endl << "базовый класс " << endl;

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

x.print();

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

cout << endl << "конструктор с значениями " << endl;

cout << "Введите информацию о программе:" << endl;

cout << endl << "Рейтинг программы ";

cin >> stars;

cout << endl << "Наличие лицензии(true/false) ";

cin >> license;

cout << endl << "Комментарий о программе ";

cin >> comment;

PKSoftware<float, int> ren(stars, license, comment, soft, cost, size);

ren.conclusion();

cout << endl << "конструктор без значений " << endl;

PKSoftware<float, int> nik;

nik.conclusion();

cout << endl << "копирующий конструктор " << endl;

PKSoftware<float, int> cop(ren);

cop.conclusion();

cout << endl << "оператор = " << endl;

nik = ren;

nik.conclusion();

cout << endl << "тестирование функций базового класса через производный" << endl;

cout << endl << "1 оператор ++ и вывод " << endl;

++nik;

nik.print();

cout << endl << "2 оператор ==" << endl;

if(nik==ren)cout << "равны" << endl;

else cout << "неравны" << endl;

cout << endl << "3 оператор +" << endl;

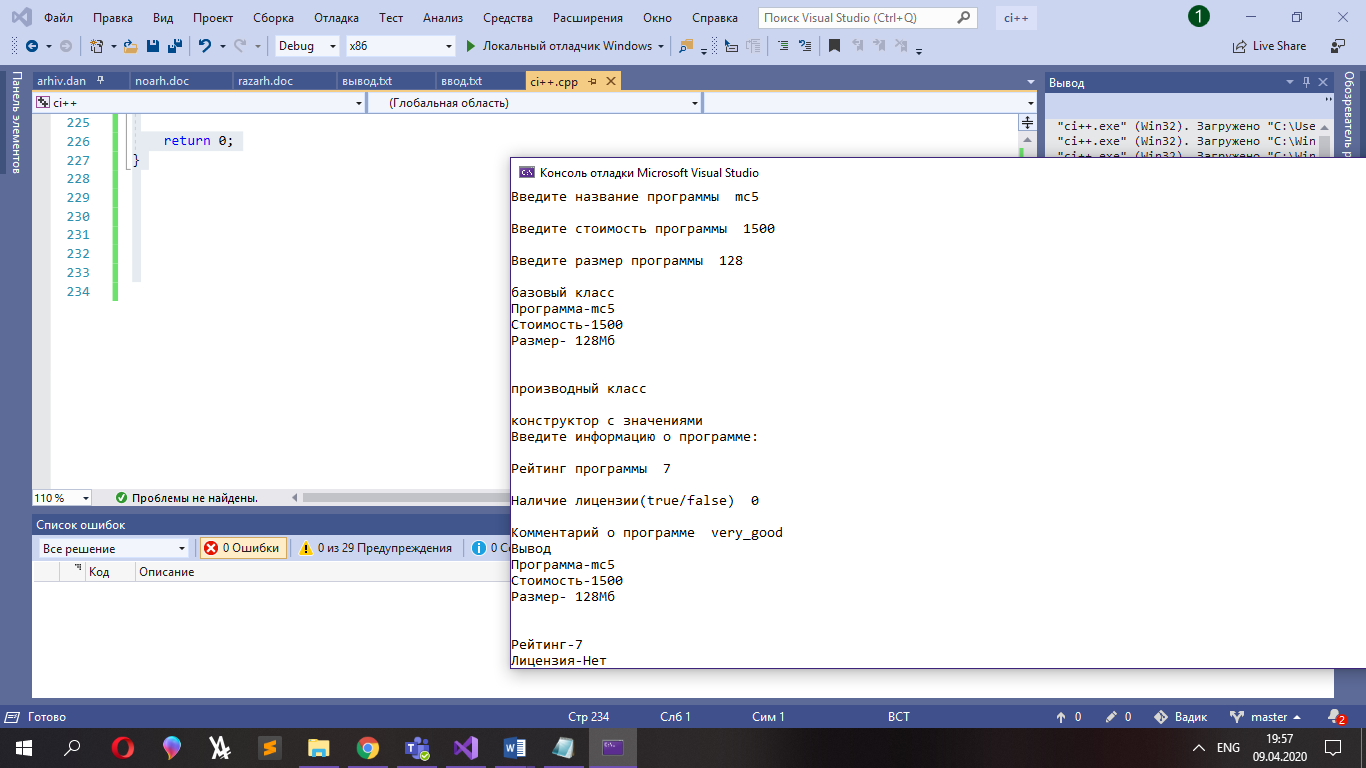
x = cop + nik;

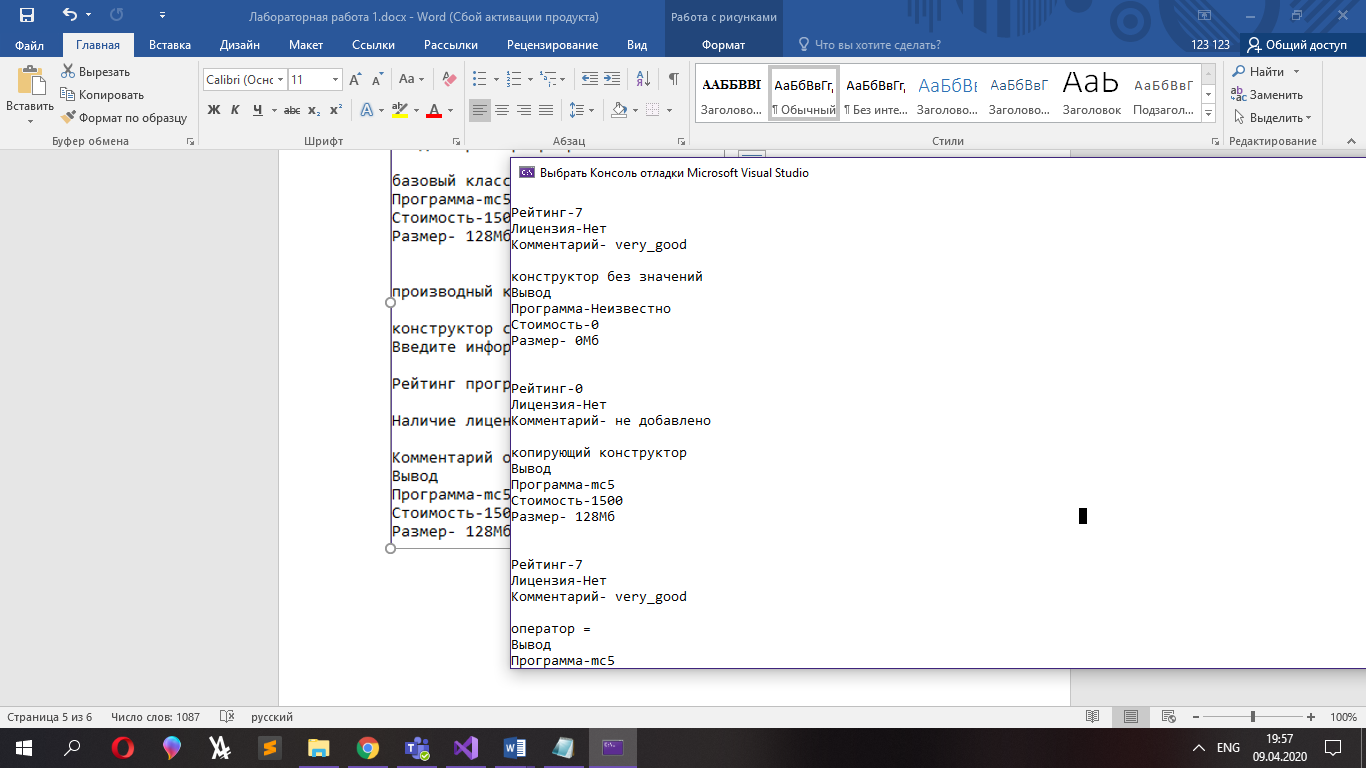
x.print();

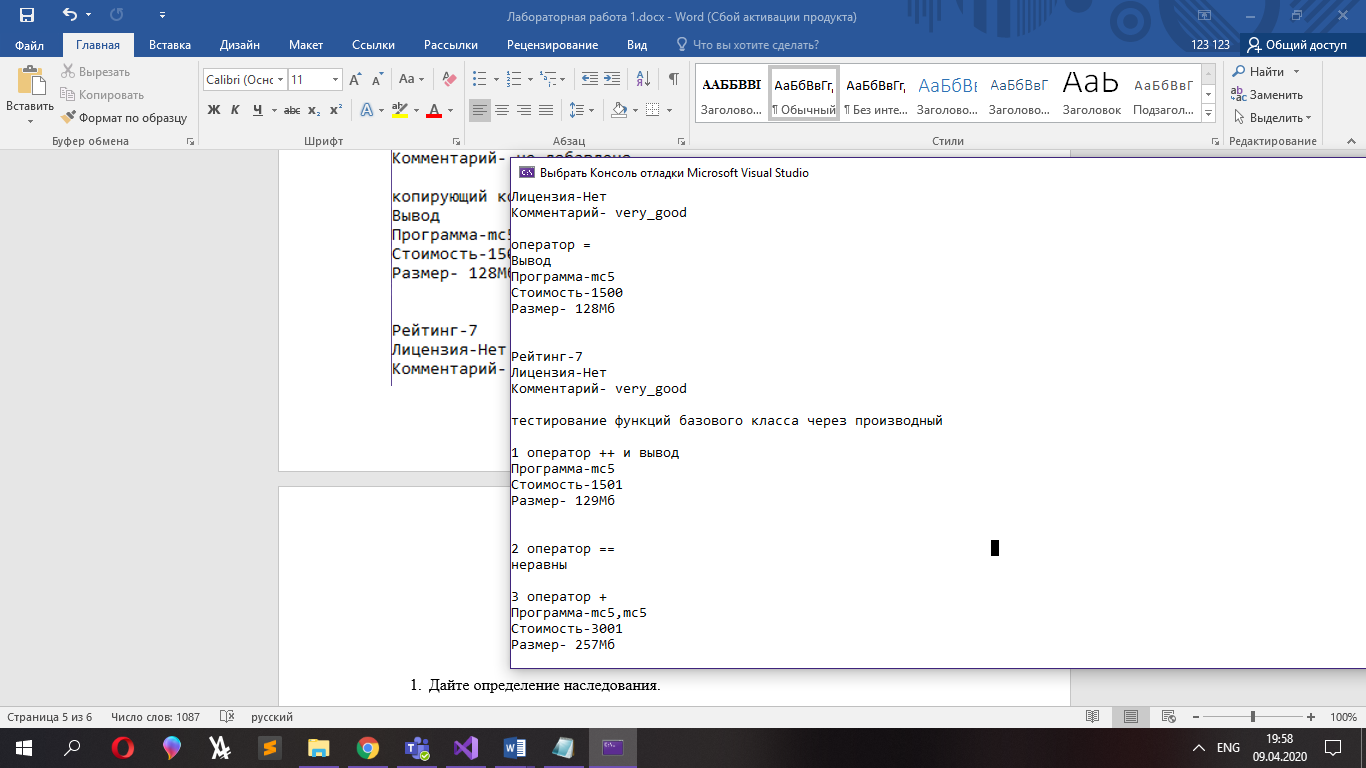
return 0;

}

Пример работы программы:







## Контрольные вопросы

1. Дайте определение наследования.

Наследование – это механизм получения нового класса на основе существующего класса. Существующий класс может быть дополнен или изменен для создания производного класса. Наследование осуществляется с помощью конструкции:

class имя\_класса:

public|protected|private)opt имя\_базового\_класса

{

объявления членов

};

1. Какие модификаторы прав доступа к производному классу вы знаете и в чем их особенности?

Модификаторы прав доступа к членам класса могут применяться в объявлении класса в любом порядке и сколько угодно раз. Открытый член доступен во всей области видимости, где виден класс. Закрытый член доступен другим функциям-членам своего класса. Защищенный член доступен не только другим функциям-членам своего класса, но и функциям-членам класса, унаследованного непосредственно от данного класса. Если все члены открыты, то ключевое слово class можно заменить ключевым словом struct.

1. Как выполняется конструктор при наследовании?

При наследовании и инициализации членов класса конструкторы выполняются в следующем порядке:

1. Базовые классы инициализируются в порядке объявления.

2. Члены инициализируются в порядке объявления.

1. Дайте определение чисто виртуальной функции.

Чисто виртуальная функция – это виртуальная функция-член, тело которой не определено. Она объявляется внутри класса:

virtual прототип\_функции = 0;

1. Какие операторы используются для разыменования указателя на член класса?

Для разыменования указателя на член класса используются два оператора: .\* и →\*.

Сначала разыменовывается указатель для получения переменной-члена, а затем происходит доступ к члену объекта.