МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

Кафедра программного обеспечения вычислительной техники и

информационных сетей

Объектно-орентированное программирование

ЛАБОРАТОРНАЯ РАБОТА №5

Программная реализация отношения простого наследования

Руководитель Ершов Е.В

Исполнитель

Студент 1ПИб-01-21оп

Хорева Л.А.

Оценка

Подпись

2020 год

**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 7, используя одиночное наследование.
2. Проверьте работоспособность АТД и производного класса на тестовом наборе данных.

**Решение:**

**Код программы:**

#include <string.h>

#include <iostream>

using namespace std;

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

class HomeAppliances {

public:

HomeAppliances<T1, T2>(const char\* programa, T1 v, T2 p);

HomeAppliances<T1, T2>();

void print();

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

bool operator==(const HomeAppliances& a);

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

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

~HomeAppliances<T1, T2>();

protected:

char\* program; T1 cost; T2 power;

};

template <class T>

class ExampleHomeAppliances : public HomeAppliances< > {

public:

ExampleHomeAppliances<T>(const char\* programa, T v, T p);

ExampleHomeAppliances<T>();

ExampleHomeAppliances<T>(const ExampleHomeAppliances& a);

void print();

~ExampleHomeAppliances<T>();

ExampleHomeAppliances<T> operator+(const ExampleHomeAppliances& a);

protected:

char\* program; T cost; T power;

};

template <class T>

ExampleHomeAppliances<T>::ExampleHomeAppliances<T>(const char\* programa, T v, T p) {

program = new char[strlen(programa) + 1];

strcpy(program, programa);

cost = v;

power = p;

};

template <class T>

ExampleHomeAppliances<T>::ExampleHomeAppliances<T>() {

program = new char[strlen("Aceline") + 1];

strcpy(program, "Aceline");

cost = 1499;

power = 1000;

};

template <class T>

ExampleHomeAppliances<T>::ExampleHomeAppliances<T>(const ExampleHomeAppliances& a) {

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

strcpy(program, a.program);

cost = a.cost;

power = a.power;

};

template <class T>

ExampleHomeAppliances<T> ExampleHomeAppliances<T>::operator+(const ExampleHomeAppliances& a) {

ExampleHomeAppliances<T> v;

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

strcpy(v.program, program);

strcat(v.program, ",");

strcat(v.program, a.program);

v.cost = cost + a.cost;

v.power = power + a.power;

return v;

}

template <class T>

void ExampleHomeAppliances<T>::print() {

cout << "Название:" << program << endl;

cout << "Стоимость:" << cost << " руб." << endl;

cout << "Мощность: " << power << "Вт" << endl;

};

template <class T>

ExampleHomeAppliances<T>::~ExampleHomeAppliances<T>() {

delete[] program;

};

template <class T1, class T2>

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

program = new char[strlen("Polaris PM C") + 1];

strcpy(program, "Polaris PM C");

cost = 2999;

power = 770;

};

template <class T1, class T2>

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

cout << "Название:" << program << endl;

cout << "Стоимость:" << cost << " руб." << endl;

cout << "Мощность: " << power << "Вт" << endl;

};

template <class T1, class T2>

HomeAppliances<T1, T2>::HomeAppliances<T1, T2>(const char\* programa, T1 v, T2 p) {

program = new char[strlen(programa) + 1];

strcpy(program, programa);

cost = v;

power = p;

};

template <class T1, class T2>

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

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

strcpy(program, a.program);

cost = a.cost;

power = a.power;

};

template <class T1, class T2>

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

delete[] program;

};

template <class T1, class T2>

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

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

cost == a.cost &&

power == a.power);

};

template <class T1, class T2>

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

HomeAppliances<T1, T2> v;

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

strcpy(v.program, program);

strcat(v.program, ",");

strcat(v.program, a.program);

v.cost = cost + a.cost;

v.power = power + a.power;

return v;

}

template <class T1, class T2>

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

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

strcpy(program, a.program);

cost = a.cost;

power = a.power;

return(\*this);

};

int main()

{

setlocale(LC\_ALL, "Russian");

char\* name = new char[10];

double cost, power;

cout << "Введите модель бытовой техники";

cin >> name;

cout << endl << " Введите стоимость модели бытовой техники";

cin >> cost;

cout << endl << "Введите мощность ";

cin >> power;

cout << "Использование производного класса";

ExampleHomeAppliances<int> w;

w.print();

ExampleHomeAppliances<int> q(name,cost,power);

q.print();

ExampleHomeAppliances<int> s(q + w);

s.print();

cout << "float and int" << endl;

HomeAppliances <float, int> q1(name, cost, power);

q1.print();

cout << "float and double" << endl;

HomeAppliances <float, double> q2(name, cost, power);

q2.print();

cout << "int and float" << endl;

HomeAppliances < int, float> a(name, cost, power);

a.print();

HomeAppliances <int, float> b(a);

cout << "Сумма показателей бытовой техники" << endl;

HomeAppliances <int, float> c(a + b);

c.print();

if (a == b)cout << "Равны" << endl;

else cout << "Не равны" << endl;

if (a == c)cout << "Равны" << endl;

else cout << "Не равны" << endl;

HomeAppliances <int, float> d;

HomeAppliances <int, float> f(d + b);

f.print();

return 0;

}

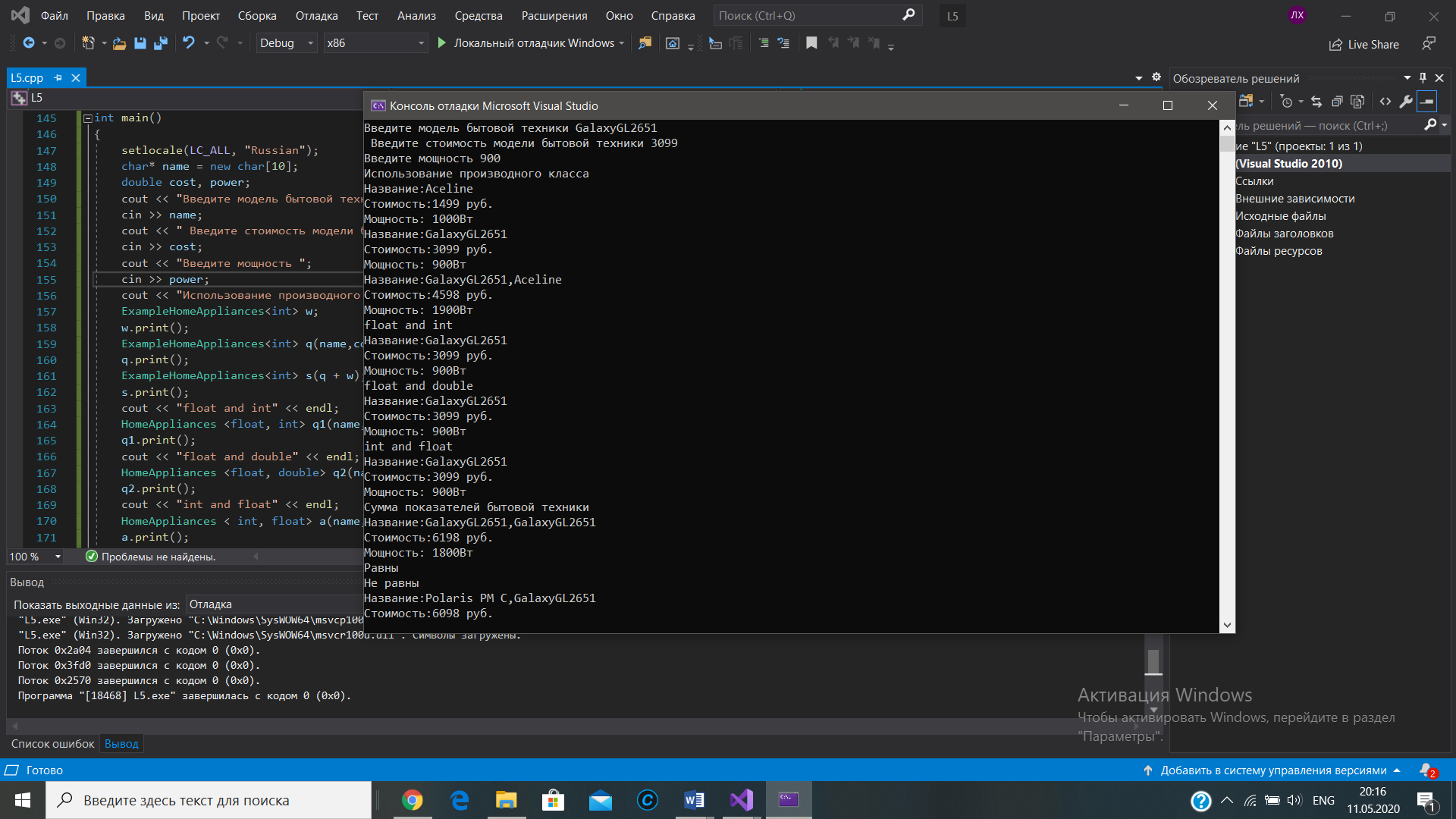
**Скрины программы:**

Рис.1.Выполнение программы

**Контрольные вопросы:**

1. **Дайте определение наследования.**

Наследование – это механизм получения нового класса на основе существующего класса.

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

Модификаторы прав доступа к членам класса (public, protected и private) 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.

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

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

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