## 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.

## Задание

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**:

#include <iostream>

#include <Windows.h>

using namespace std;

class ElectricDevice {//Абстрактный базовый класс

public:

string name; int volt, ampr;

ElectricDevice() {

name = "";

volt = NULL;

ampr = NULL;

}

virtual void har(int a) = 0;//Чисто виртуальная функция с параметром

virtual void work() {//виртуальная функция без параметра

cout << "Потребляет электричество" << endl;

};

};

class Toast : public ElectricDevice {//Класс тостеров, наследник абстрактного класса

public:

int timework; string degdonenes;

void work() override {//Переописываем виртуальною функцию

cout << "Готовит тосты" << endl;

}

void har(int a) override {//Переописываем виртуальною функцию

if (a > 1) {

if (a == 3) {

cout << "Название: " << name << endl

<< "Мощность: " << volt << endl

<< "Сила тока: " << ampr << endl

<< "Время готовки: " << timework << endl

<< "Степень прожарки: " << degdonenes << endl;

}

else {

cout << "Время готовки:" << timework << endl

<< "Степень прожарки:" << degdonenes << endl;

}

}

else{

cout << "Название: " << name << endl

<< "Мощность: " << volt << endl

<< "Сила тока: " << ampr << endl;

}

}

friend std::istream& operator>> (std::istream& in, Toast& a) {//перегрузка оператора cin

cout << "Название: "; in >> a.name;

cout << "Мощность: "; in >> a.volt;

cout << "Сила тока: "; in >> a.ampr;

cout << "Время приготовления: "; in >> a.timework;

cout << "Степень прожарки: "; in >> a.degdonenes;

return in;

}

Toast() {

timework = NULL;

degdonenes = "";

}

};

class Fridge : public ElectricDevice {//Класс холодильников, наследник абстрактного класса

public:

int height, width;

void work() override {//Переописываем виртуальною функцию

cout << "Продлевает срок годности продуктов" << endl;

}

void har(int a) override {//Переописываем виртуальною функцию

if (a > 1) {

if (a == 3) {

cout << "Название: " << name << endl

<< "Мощность: " << volt << endl

<< "Сила тока: " << ampr << endl

<< "Высота: " << height << endl

<< "Ширина: " << width << endl;

}

else {

cout << "Высота: " << height << endl

<< "Ширина: " << width << endl;

}

}

else {

cout << "Название: " << name << endl

<< "Мощность: " << volt << endl

<< "Сила тока: " << ampr << endl;

}

}

friend std::istream& operator>> (std::istream& in, Fridge& a) {//перегрузка оператора cin

cout << "Название: "; in >> a.name;

cout << "Мощность: "; in >> a.volt;

cout << "Сила тока: "; in >> a.ampr;

cout << "Высота: "; in >> a.height;

cout << "Ширина: "; in >> a.width;

return in;

}

Fridge() {

height = NULL;

width = NULL;

}

};

template<typename T>//Объявление шаблона

class Stack {//Класс отвечающий за реализацию стек

private:

class Node {//Вложенный класс отвечающий за компоненты стека

public:

T \*date;

Node\* Next;

Node(T\* date, Node\* next) {//Конструктор класса

this->date = date;

this->Next = next;

}

};

Node\* Head;

int size;

public:

Stack() {//Конструктор класса

Head = nullptr;

size = 0;

};

~Stack() {//Диструктор класса

clear();

};

bool emp() {//Проверка на пустоту

return Head == nullptr;

}

void clear() {//Удаление объектов стека

while (size) {

pop();

};

}

void push(T \*a) {//Добавление нового компонента в стек

Head = new Node(a, Head);

size++;

}

void print() {//Вывод содержимого стека в консоль

if (!emp()) {

Node\* current = Head;

int a;

cout << "Режим отображения:" <<endl

<<"1-Только параметры базового класса" <<endl

<<"2-Только параметры класса потомка" <<endl

<<"3-Все параметры"; cin >> a;

while (!emp()) {

Head->date->har(a);

Head = Head->Next;

}

Head = current;

cout << size << endl;

}

else { cout << "Стек пуст" << endl; }

};

void pop() {//Удаление первого компонента из стека

Node\* current = Head;

Head = Head->Next;

delete current;

size--;

}

void hideval(T\* val)//Поиск компонента по значению

{

Node\* cur = Head;

for (int i = 1; i <= size; i++) {

if (Head->date->name == val->name) {

cout << "Объект находится на:" << i << " позиций" << endl;

Head->date->har(3);

Head = Head->Next;

}

else {

Head = Head->Next;

}

}

Head = cur;

};

void hidenum(const int a) {//Пререгруженный оператора индексации "Поиск компонента по номеру в стеке"

int c = 1;

Node\* current = this->Head;

while (current != nullptr)

{

if (c == a)

{

current->date->har(3);

}

current = current->Next;

c++;

}

}

;

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Stack<ElectricDevice> stack;

int n = 1;

Toast c1;

Fridge c2;

while (n != 0) {

cout << "Выберите действие:" << endl

<< "0 - выход" << endl

<< "1 - добавить объект в стек" << endl

<< "2 - убрать первый объект из стека" << endl

<< "3 - поиск объекта по названию" << endl

<< "4 - найти объект по номеру" << endl

<< "5 - просмотреть все объекты в стеке" << endl

<< "6 - убрать все объекты из стека" << endl;

cin >> n;

int a;

switch (n)

{

case (1): {

int b;

cout << "Какои объект добавить?" << endl

<< "1-тостер" << endl

<< "2-холодильник" << endl;

cin >> b;

if (b == 1) {

cin >> c1;

stack.push(&c1);

}

else {

cin >> c2;

stack.push(&c2);

}}

break;

case (2):

stack.pop();

cout << "Объекст убран" << endl;

break;

case(3):

int b;

cout << "Какой объект хотите найти?" << endl

<<"1-Тостер" << endl

<<"2-Холодильник" << endl; cin >> b;

if(b==1){

Toast c;

cin >> c;

stack.hideval(&c); }

else {

Fridge c;

cin >> c;

stack.hideval(&c); }

break;

case(4):

cout << "Номер:";

cin >> a;

stack.hidenum(a);

break;

case(5):

stack.print();

break;

case(6):

stack.~Stack();

cout << "Объекты убраны" << endl;

break;

}

}

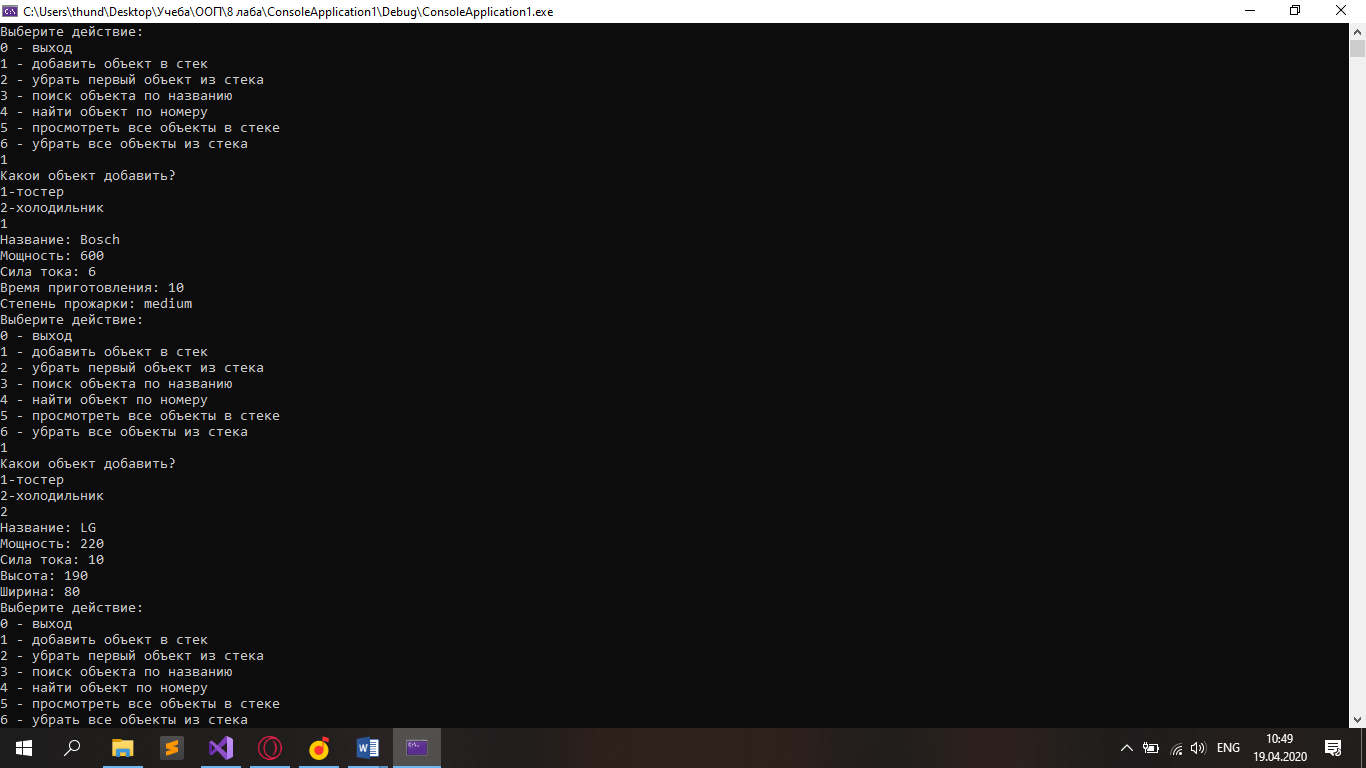
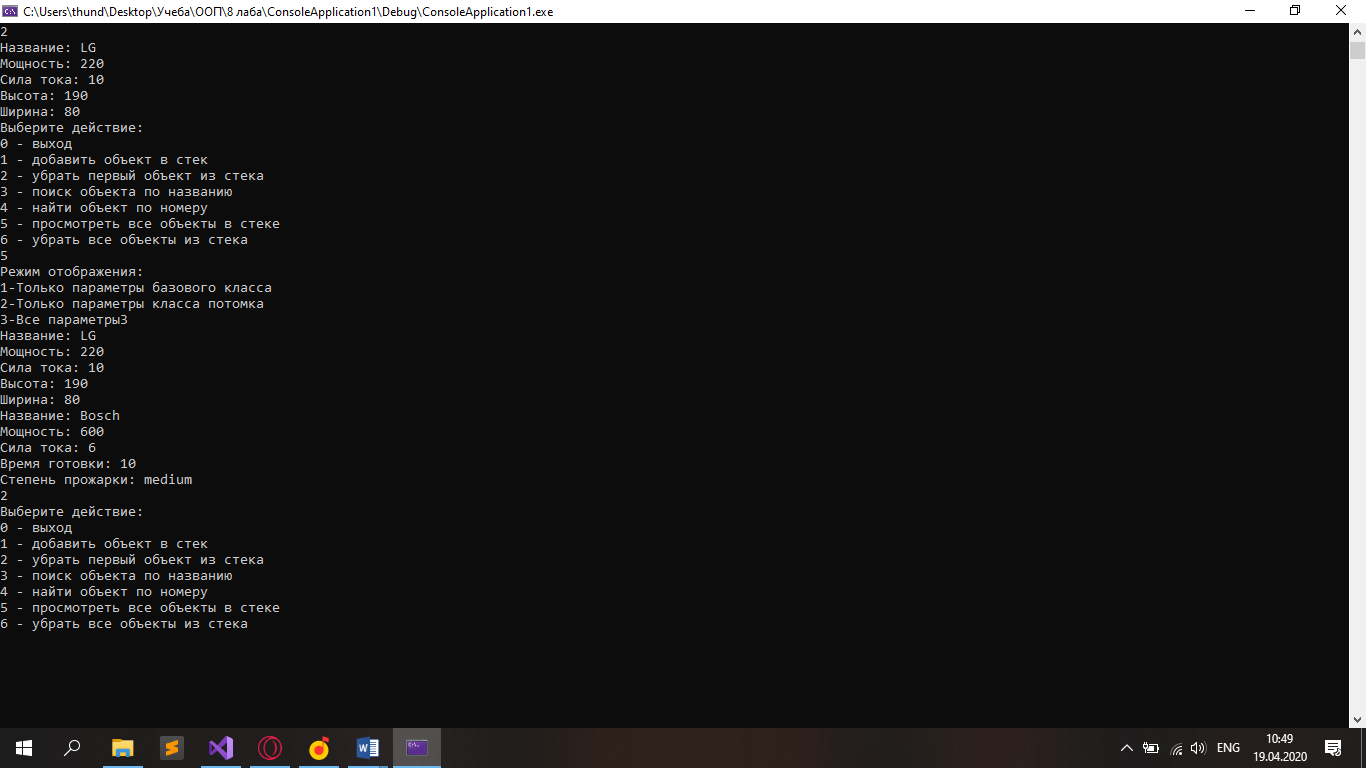
system("pause");

return 0;

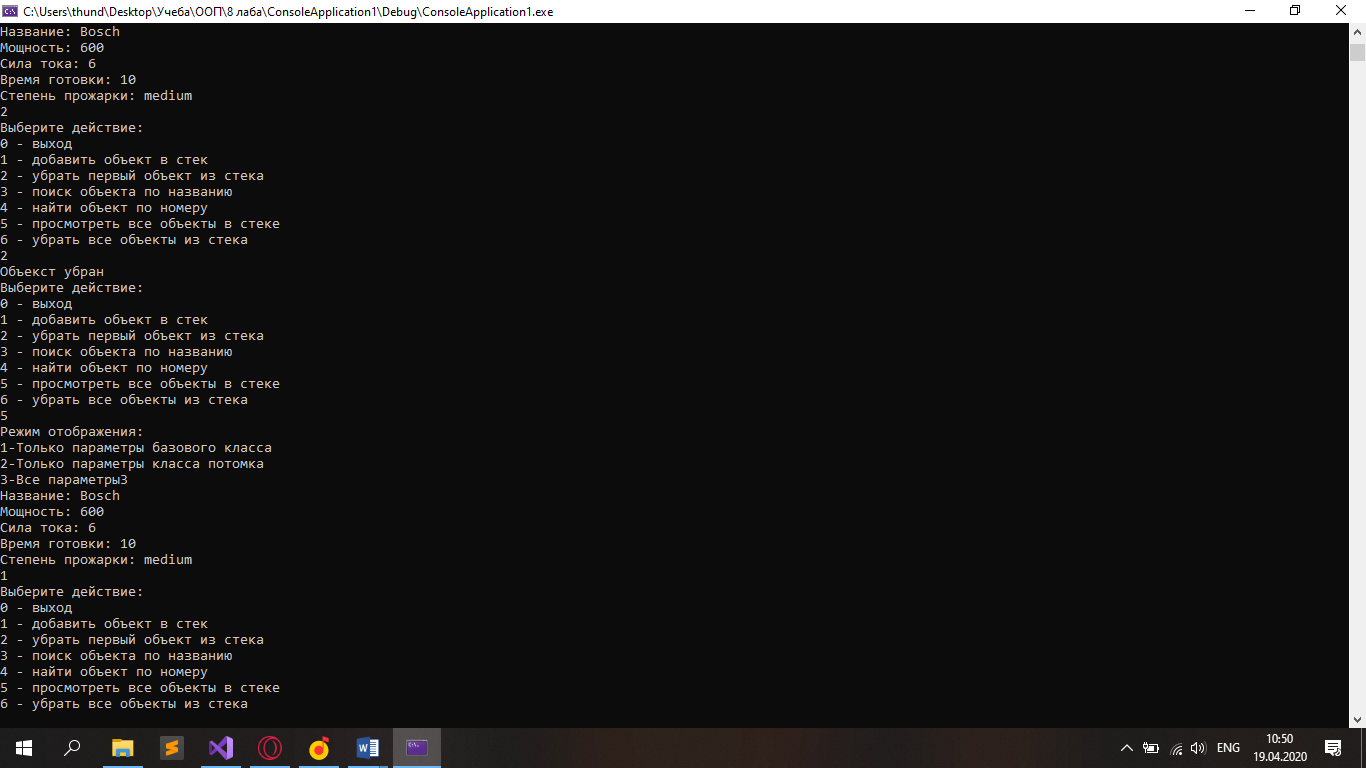
}

**Тесты**:

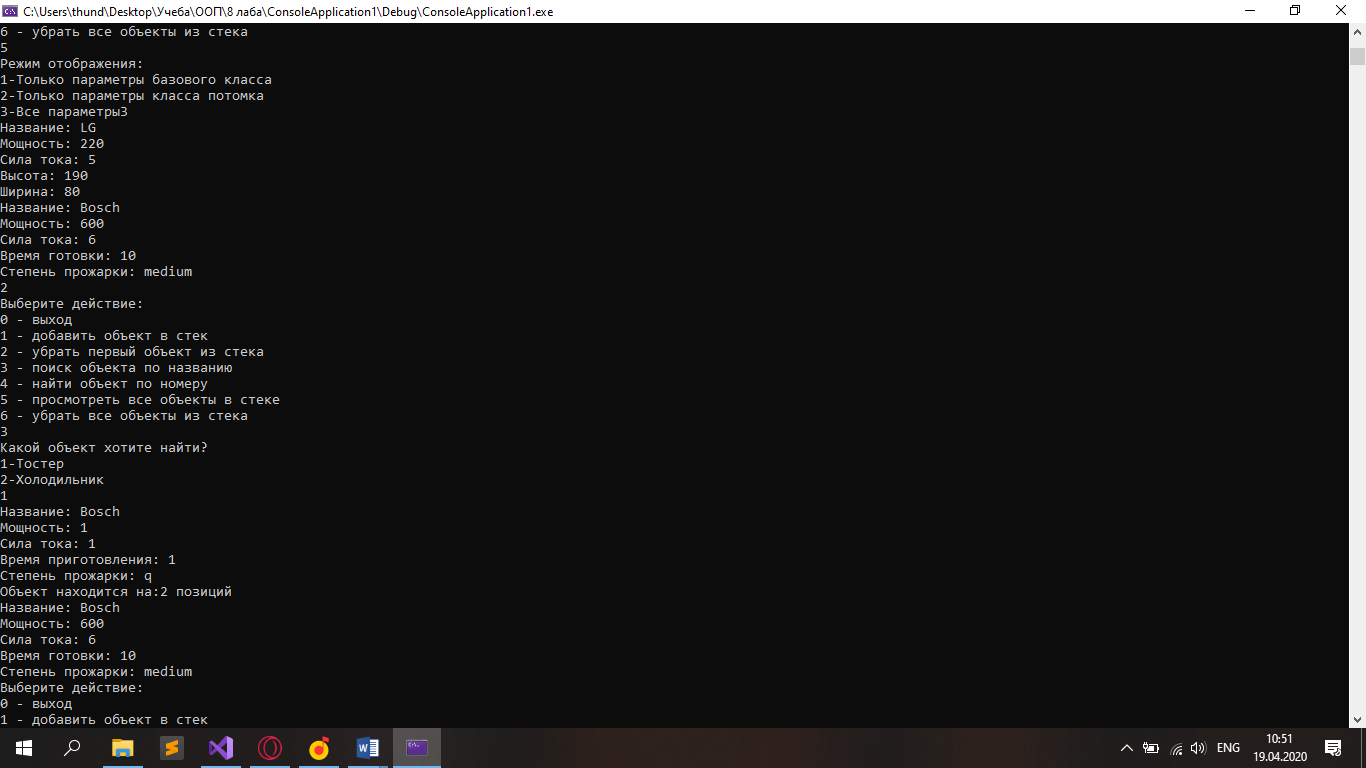
1. Добавление объекта



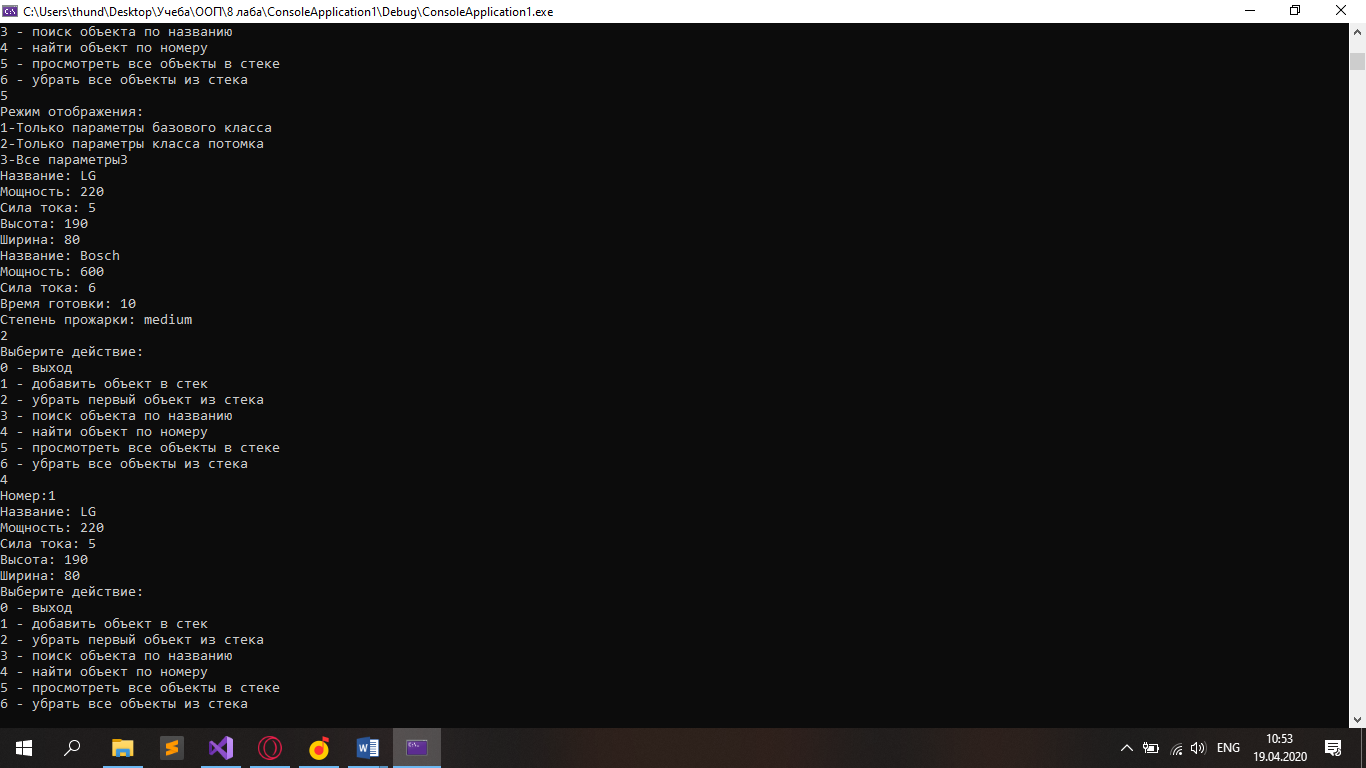
1. Удаление объекта



1. Поиск объекта по названию



1. Поиск объекта по позиции в стеке



1. Удаление всех объектов

