**ПРОГРАММИРОВАНИЕ ШАБЛОННЫХ КЛАССОВ**

**И ФУНКЦИЙ**

**Цель работы**: изучить способы определения шаблонов, принцип и механизм создания шаблонных классов и шаблонов функций, получить практические навыки работы с шаблонными классами.

**ЗАДАНИЕ**

1. Модифицируйте абстрактный тип данных, реализованный по заданию лабораторной работы 6, согласно варианту задания (см. раздел III УМП\_ООП\_1.doc), используя шаблоны определения класса и шаблоны определения функции.

2. Проверьте работоспособность АТД на тестовом наборе данных для разных типов.

**РЕШЕНИЕ**

**Код**:

#include <iostream>

#include <Windows.h>

using namespace std;

class Car {//Класс отвечающий за объекты стека

private:

string name, model, type;

int cost;

public:

friend std::ostream& operator<< (std::ostream& out, const Car& mah) {//Перегруженный оператор вывода

out << "Название:" << mah.name << endl

<< "Модель:" << mah.model << endl

<< "Тип:" << mah.type << endl

<< "Стоимость:" << mah.cost << endl;

return out;

};

friend std::istream& operator>> (std::istream& in, Car& mah) {//Перегруженный оператор ввода

cout << "Название:"; in >> mah.name;

cout << "Модель:"; in >> mah.model;

cout << "Тип:"; in >> mah.type;

cout << "Стоимость"; in >> mah.cost;

return in;

};

friend bool operator ==(const Car& a1, const Car& a2) {//Перегруженный оператора стравнения "равно"

return (a1.name == a2.name);

}

Car() {//Конструктор класса без параметров

name = "";

model = "";

type = "";

cost = 0;

}

Car(string n, string m, string t, int c) {//Конструктор класса с параметрами

name = n;

model = m;

type = t;

cost = c;

}

};

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;

while (!emp()) {

cout << Head->date << " ";

Head = Head->Next;

}

Head = current;

cout << size << endl;

}

else { cout << "Салон 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 == val) {

cout << i << ":" << Head->date << endl;

Head = Head->Next;

}

else {

Head = Head->Next;

}

}

Head = cur;

};

Car& operator[](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)

{

return current->date;

}

current = current->Next;

c++;

}

}

;

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Stack<Car> salon;

int n = 1;

while (n != 0) {

cout << "Выберите действие:" << endl

<< "0 - выход" << endl

<< "1 - добавить машину в салон" << endl

<< "2 - убрать первую машину из салона" << endl

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

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

<< "5 - просмотреть все машины в салоне" << endl

<< "6 - убрать все машины из салона" << endl;

cin >> n;

int a;

Car cur;

switch (n)

{

case (1):

cin >> cur;

salon.push(cur);

break;

case (2):

salon.pop();

cout << "Машина убрана" << endl;

break;

case(3):

cin >> cur;//Не обязательно заполнять все данные точно, за исключением названия

salon.hideval(cur);

break;

case(4):

cout << "Номер:";

cin >> a;

cout << salon[a] << endl;;

break;

case(5):

salon.print();

break;

case(6):

salon.~Stack();

cout << "Машины убраны" << endl;

break;

}

}

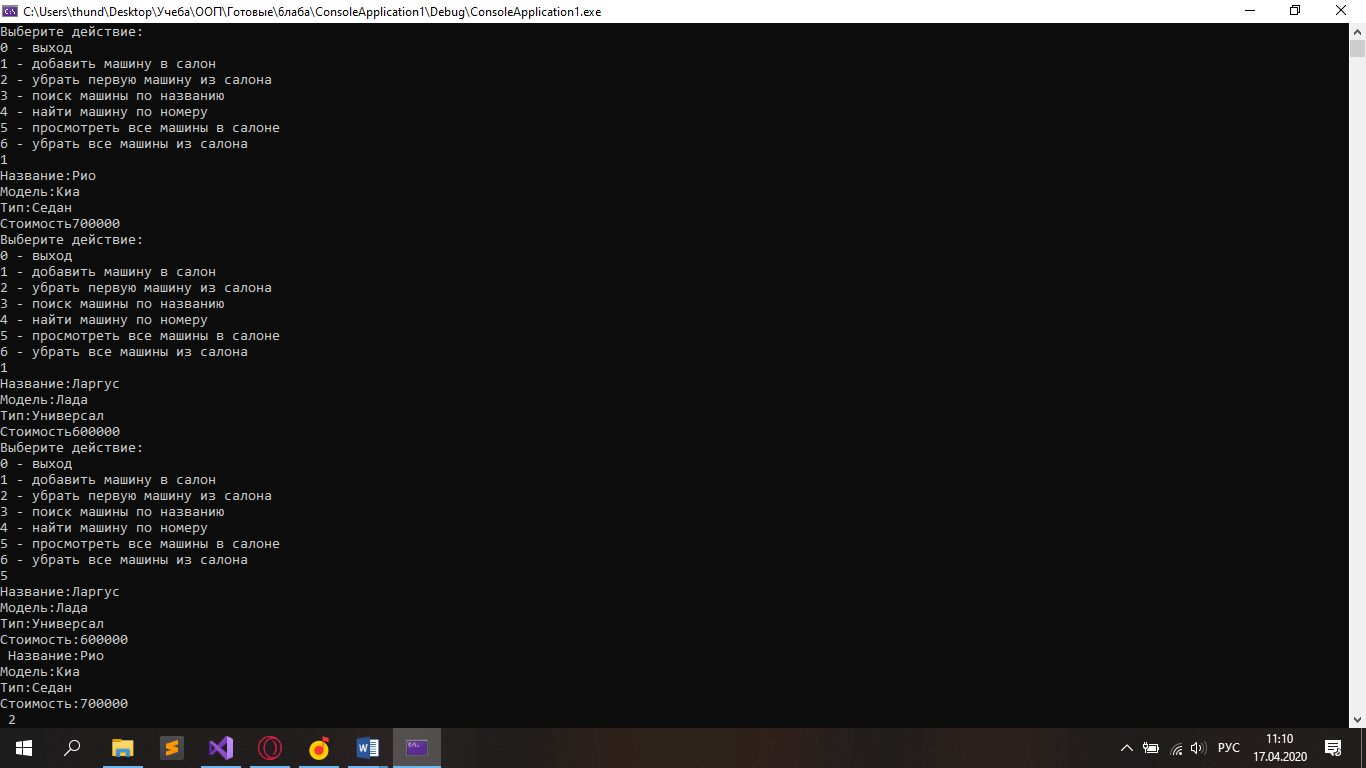
system("pause");

return 0;

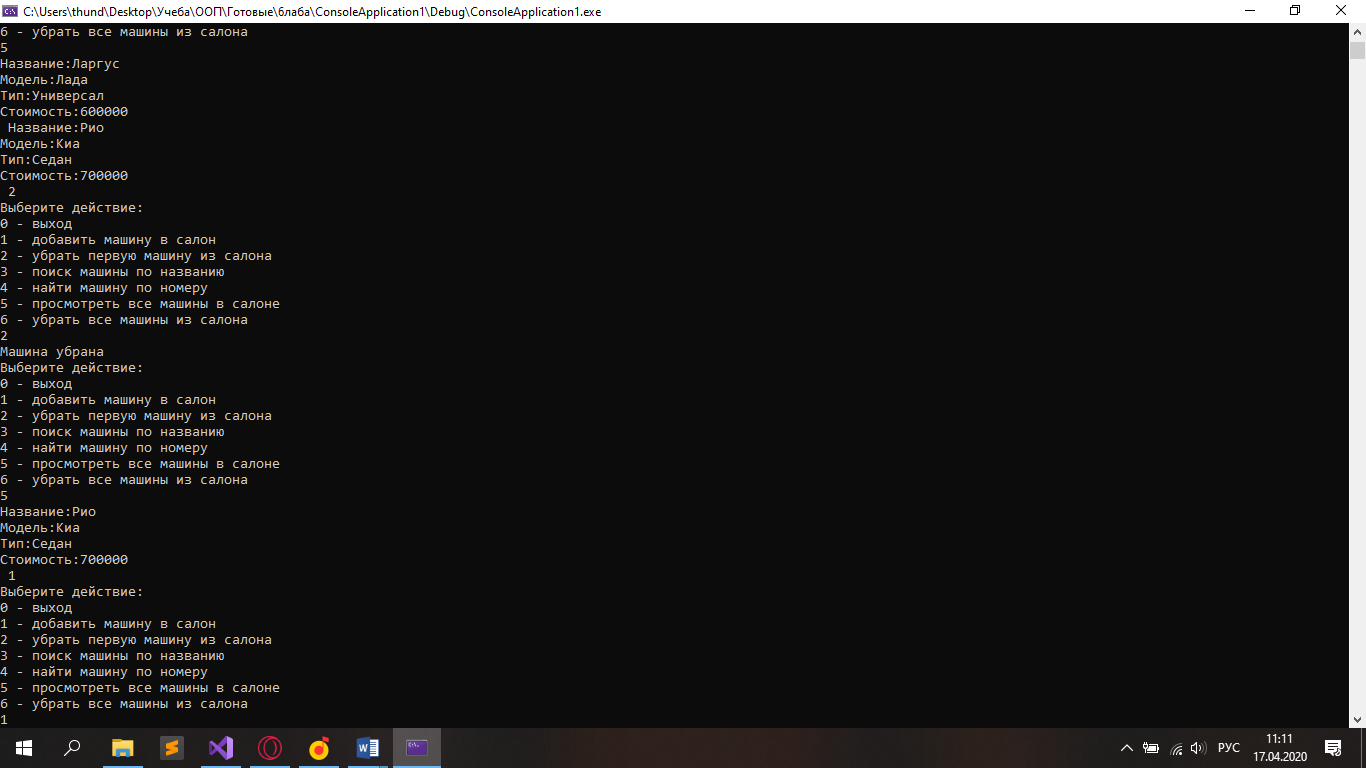
}

**Тесты**:

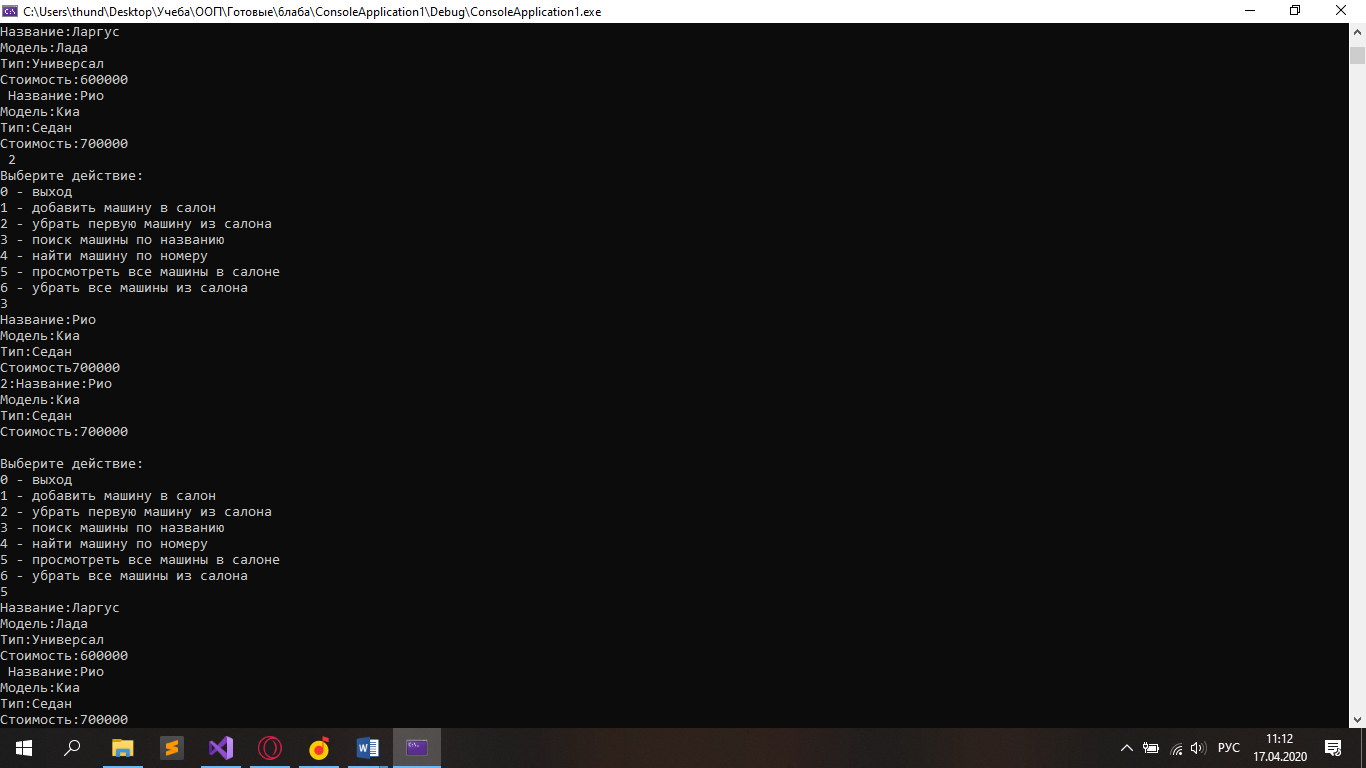
1. Добавление нового компонента в стек



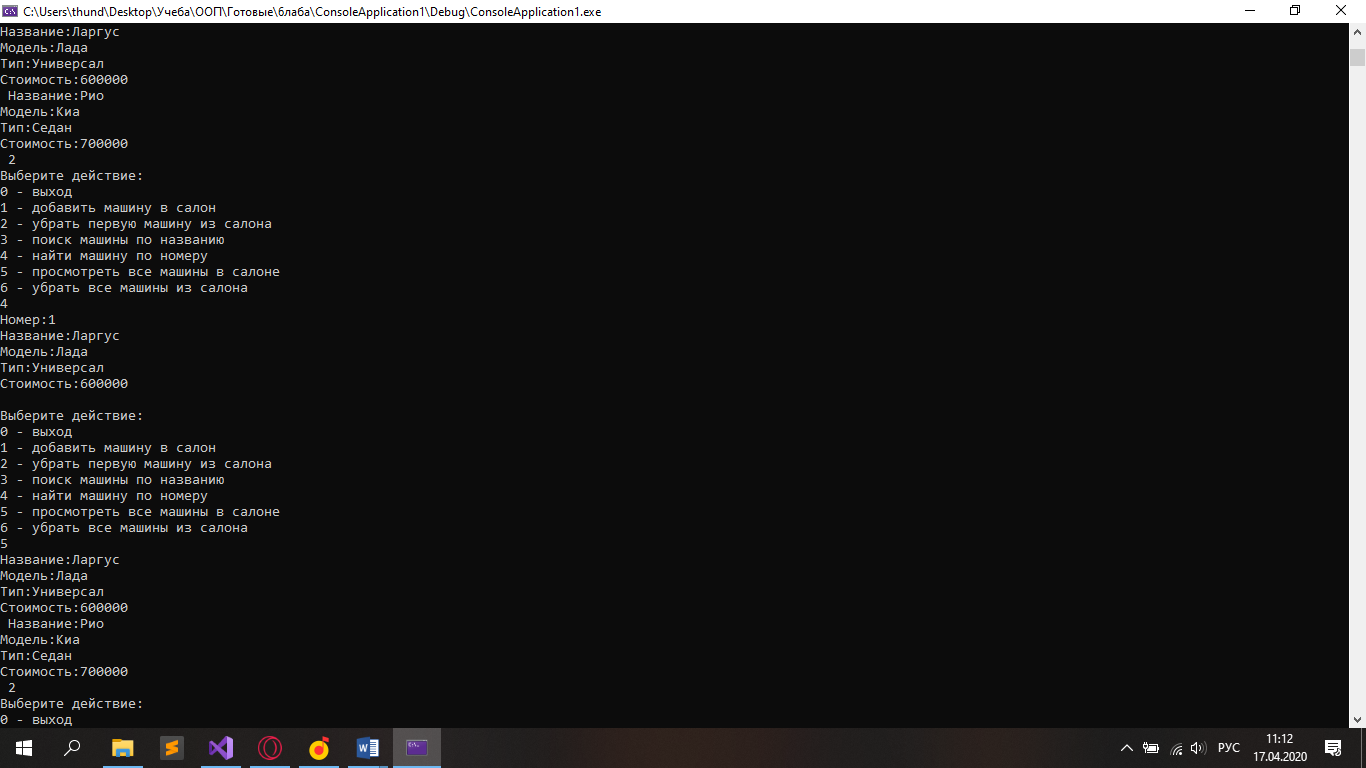
1. Удаление первого компонента в стеке



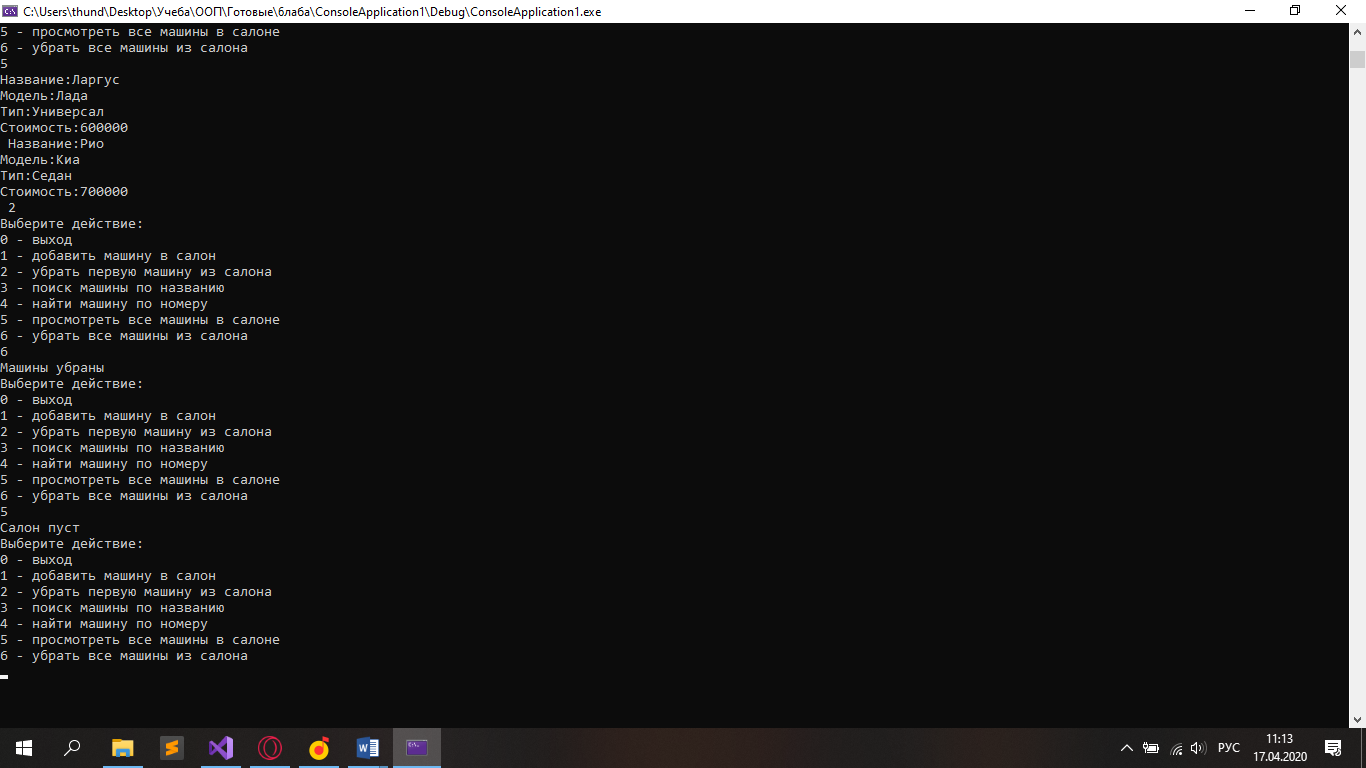
1. Поиск компонента по названию



1. Поиск компонента по позиции в стеке



1. Удаление всех компонентов



1. **Тест шаблона**

