## ОБРАБОТКА СТРУКТУРИРОВАННЫХ

## ТИПОВ ДАННЫХ

**Цель работы**: изучить способы определения структур, правила инициализации и доступа к элементам структур; приобрести практические навыки передачи структур функциям; изучить способы создания динамических структур данных, особенности работы с объединениями.

## Задание

Разработайте алгоритм и программу, реализующую структурированный тип данных (СТД), согласно варианту. Определите функции инициализации, присваивания, вывода содержимого и обработки (сортировка, поиск, сравнение, арифметические действия и т.д.).

Вариант СТД:

1. Динамический стек;

**РЕШЕНИЕ**

**Алгоритм:**

1. Описываем структуру, которая будет отвечать за компонент стека. Она содержит поле с типом string (название объекта) и ссылку на следующий компонент.
2. Описываем структуру, которая содержит поле с ссылкой на первый элемент стека.
3. Определяем функций инициализации, присваивания, вывода содержимого и обработки.
4. С помощью цикла while и оператора switch реализуем интерфейс программы.

**Код**:

#include "stdafx.h"

#include <iostream>

#include <ctime>

#include <Windows.h>

using namespace std;

struct comp//Компонент стека

{

string val;//Значение компонента

comp \*next;//Ссылка на следующий компонент

};

struct Stac

{

comp\* head;//"Голова" стека

};

void initstac(Stac&a) {//Инициализация стека

a.head = NULL;

}

void Add\_Stack(Stac &a, string v) {//Добавление нового компонента в стек

comp\* c = new comp();

c->val = v;

c->next = a.head;

a.head = c;//Новый компонент становится "головой" стека, старая "голова" смещается на 2 место

}

void popStac(Stac &a) {//удаление одного компонента

comp\* c = a.head;

a.head = a.head->next;

delete c;

}

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

while (a.head != 0) {

cout << a.head->val << " ";

a.head = a.head->next;

}

cout << endl;

}

void findname(Stac a, string b) {//Поиск компонента по названию

int x = 1;

while (a.head != NULL){

if (a.head->val == b) {

cout << b << " находится на " << x << " позиций." << endl;

}

a.head = a.head->next;

x++;

}

}

void findnum(Stac a, int i) {//Поиск компонента по номеру в стеке

int x = 1;

while (x < i) {

a.head = a.head->next;

x++;

}

cout << "На " << i << " позиций находится:" << a.head->val << endl;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Stac c;

string name;

initstac(c);

srand(time(0));

int n = 1;

while (n != 0) {

cout << "Выберите действие:" << endl

<< "0 - выход" << endl

<< "1 - положить объект на полку" << endl

<< "2 - снять первый объект с полки" << endl

<< "3 - поиск интересующего объекта" << endl

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

<< "5 - просмотреть все объекты на полке" << endl

<< "6 - убрать все объекты с полки" << endl;

cin >> n;

int a;

switch (n)

{

case (1):

cout << "Название объекта:";

cin >> name;

Add\_Stack(c, name);

break;

case (2):

popStac(c);

cout << "объект убран" << endl;

break;

case(3):

cout << "Название объекта:";

cin >> name;

findname(c, name);

break;

case(4):

cout << "Номер:";

cin >> a;

findnum(c, a);

break;

case(5):

print(c);

break;

case(6):

while (c.head != NULL) {

popStac(c);

}

cout << "Объекты удалены" << endl;

break;

}

}

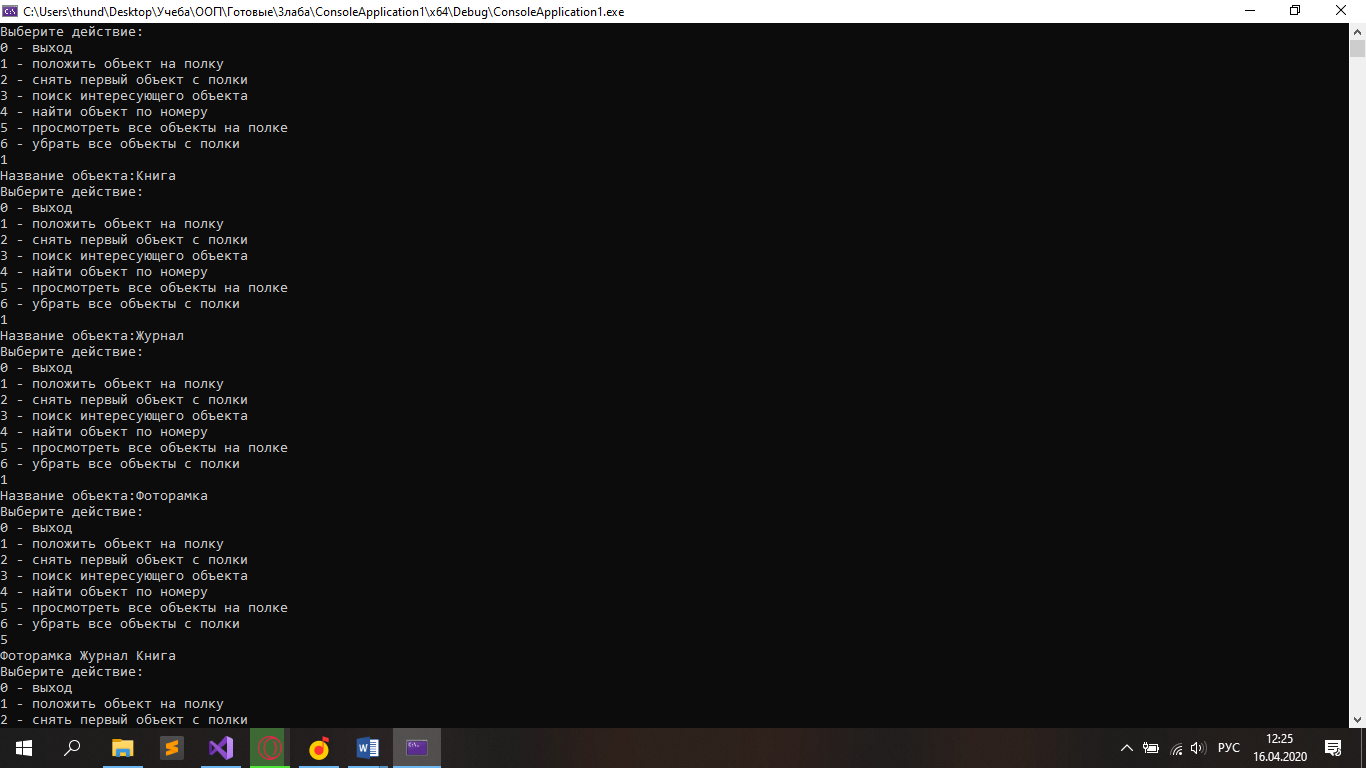
system("pause");

return 0;

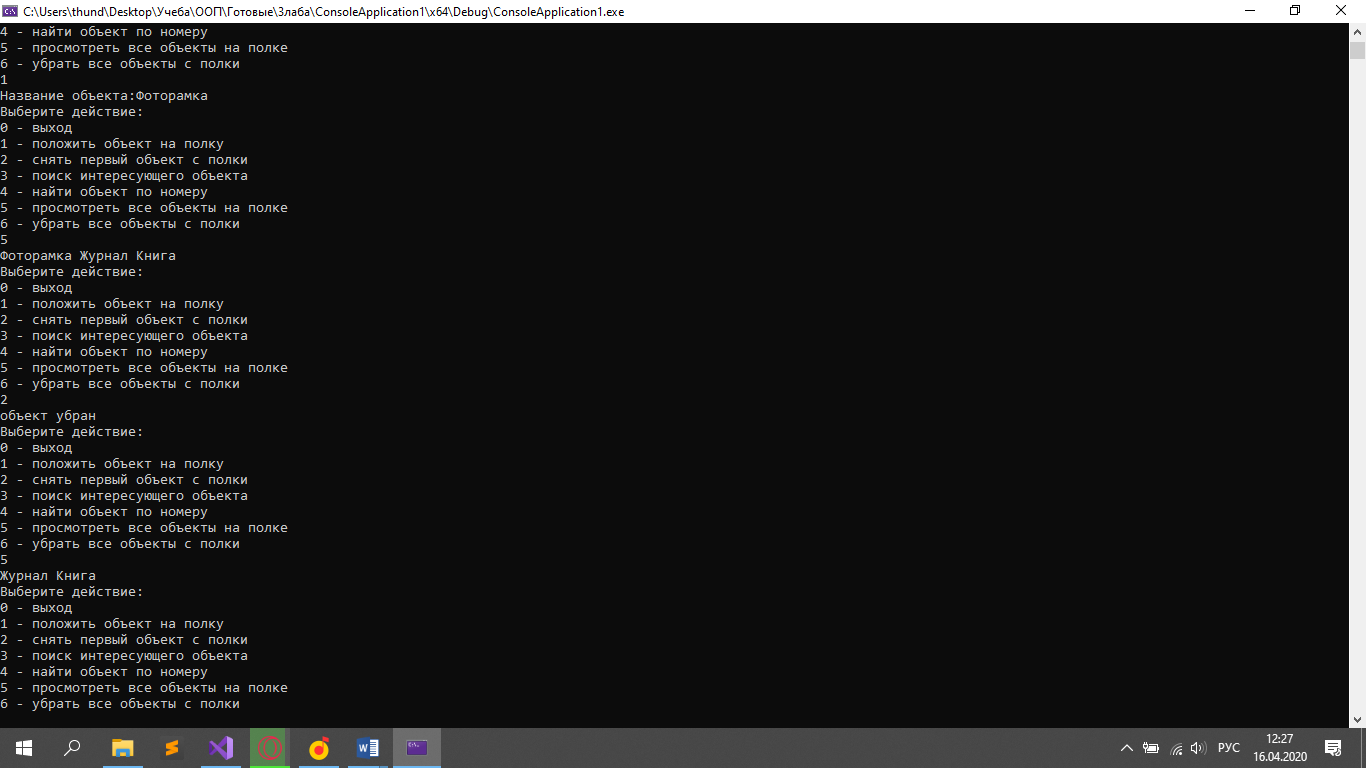
}

**Тесты**:

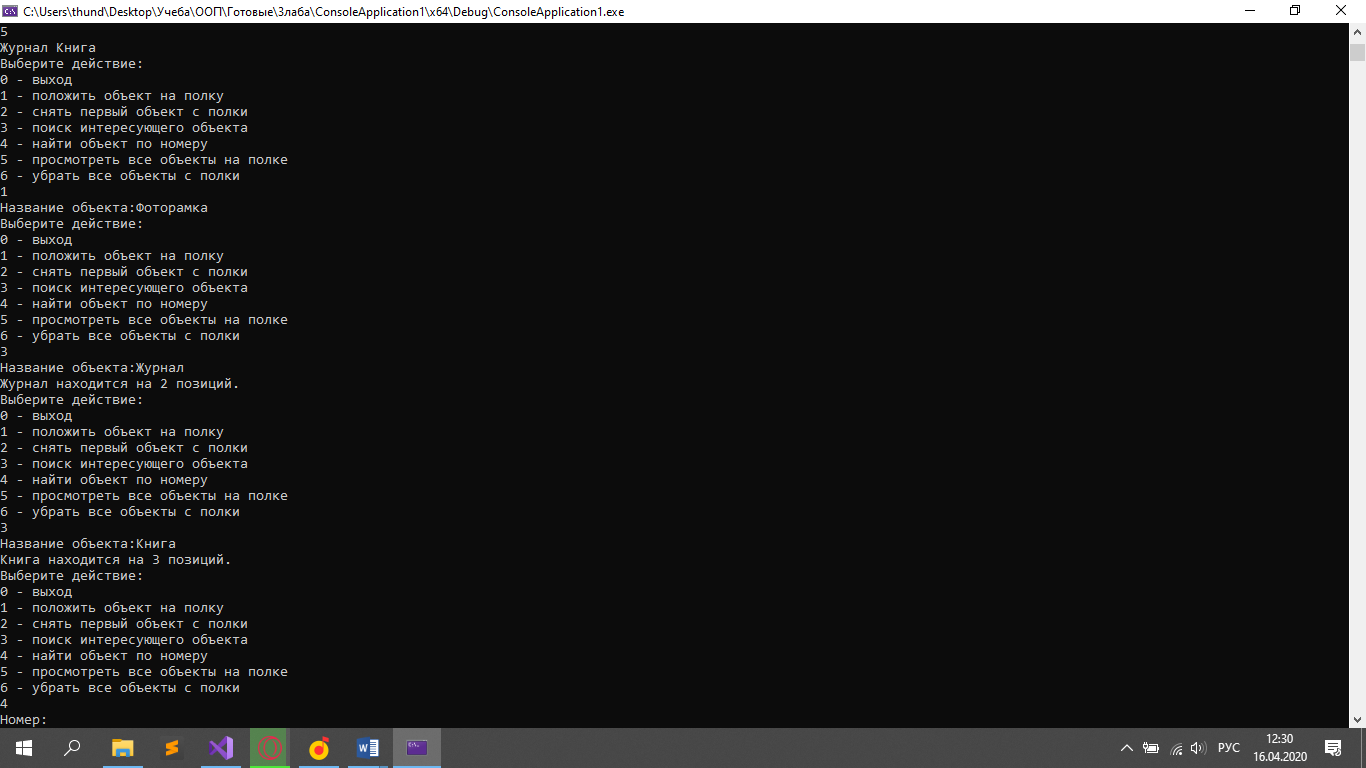
1. Добавление объектов на полку и просмотр содержимого



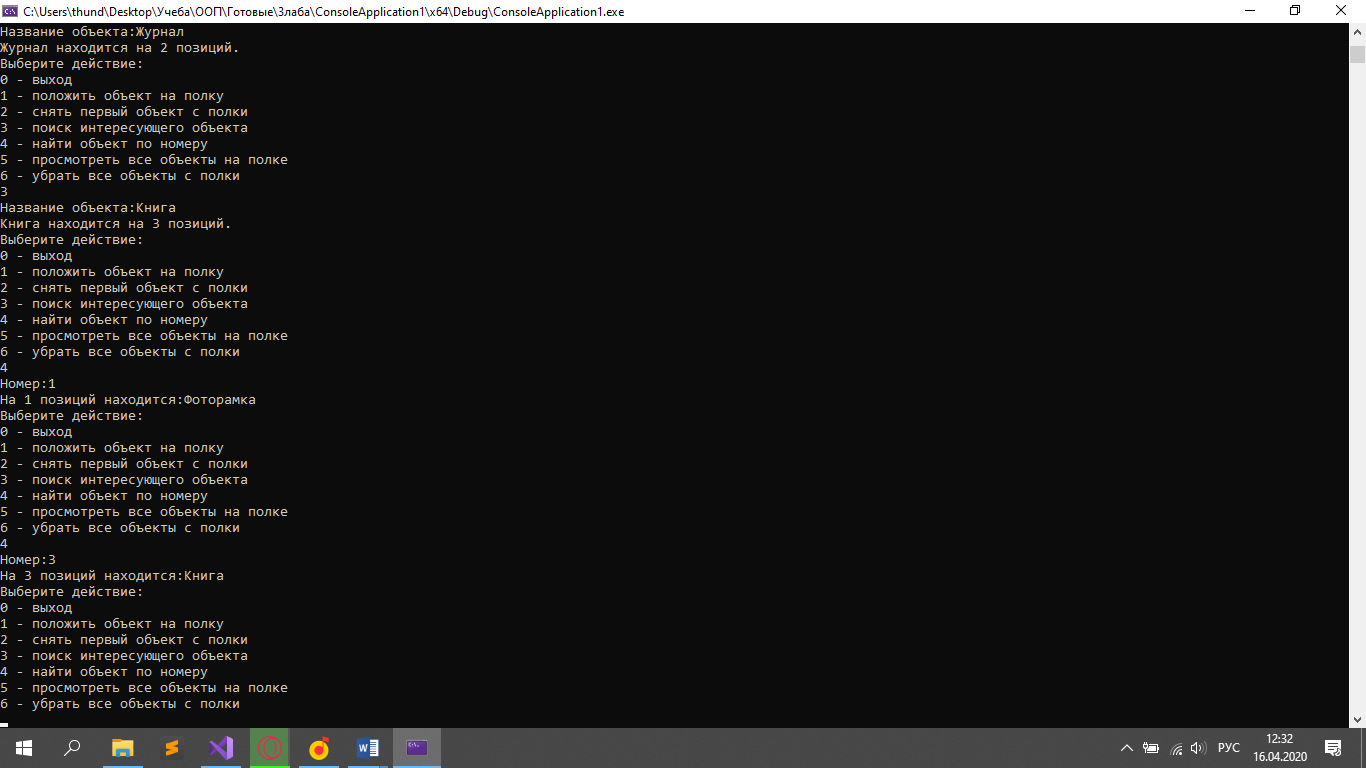
1. Снятие первого объекта с полки



1. Поиск интересующего объекта



1. Поиск объекта по номеру



1. Удаление всех объектов

