**СОЗДАНИЕ И УНИЧТОЖЕНИЕ ОБЪЕКТОВ КЛАССА**

Цель работы: изучить синтаксис и семантику определения и вызова конструкторов и деструкторов; приобрести практические навыки создания и уничтожения объектов; изучить особенности применения различных видов конструкторов.

# Задание

Модифицируйте абстрактный тип данных, реализованный по заданию раздела IV, согласно варианту задания (см. раздел III), заменив соответствующие компонентные функции на систему конструкторов и деструктор.

Проверьте работоспособность АТД на тестовом наборе данных.

**РЕШЕНИЕ**

**Код**:

#include "stdafx.h"

#include <iostream>

#include <Windows.h>

using namespace std;

class Stac {

private:

class Node {//Вложенный класс который используется как компонент стека

public:

string date;

Node\* Next;

Node (string a = NULL, Node \*b = nullptr) {//конструктор класса node

this->date = a;

this->Next = b;

}

};

Node \*Head;

int size;

public:

Stac() {//контсруктор класса stac

Head = nullptr;

size = 0;

}

Stac(const Stac &a) {//Конструктор копирования

Head = NULL;

Node\* cur = a.Head;

int b = a.size;

while (b != 0) {

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

cur = cur->Next;

}

push(cur->date);

b--;

cur = a.Head;

}

}

~Stac() {//диструктор класса stac

Node\* tmp;

while (!emp()) {

tmp = Head;

Head = Head->Next;

delete(tmp);

size--;

}

}

bool emp() {//Проверка на пустоту

return Head == nullptr;

}

void push(string a) {//Добавление компонента в стек

if (emp()) {

Head = new Node(a);

size++;

}

else {

Node \*current = Head;

Head = new Node(a);

Head->Next = current;

size++;

}

}

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

if (!emp()) {

Node\* current = Head;

while (!emp()) {

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

Head = Head->Next;

}

Head = current;

cout << endl << size << endl;

}

else { cout << "Полка пуста" << endl; }

}

void pop() {//Удаление первого компонента стека

Node \*current = Head;

Head = Head->Next;

delete current;

size--;

}

void hideval(string 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;

}

void hiadnum(int num) {//Поиск компонента по позиций в стеке

int a = 1;

if (int i = num - 1 < size) {

Node\* cur = Head;

while (a < num) {

Head = Head->Next;

a++;

}

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

Head = cur;

}

else {

cout << "книги нет" <<endl;

}

}

};

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Stac Bib;

string name;

int n = 1;

while (n != 0) {

cout << "Выберите действие:" << endl

<< "0 - выход" << endl

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

<< "2 - снять первую книгу с полки" << endl

<< "3 - поиск интересующей книги" << endl

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

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

<< "6 - убрать все книги с полки" << endl

<< "7 - тест конструктора копирования" << endl;

cin >> n;

int a;

switch (n)

{

case (1):

cout << "Название книги:";

cin >> name;

Bib.push(name);

break;

case (2):

Bib.pop();

cout << "Книга убрана" << endl;

break;

case(3):

cout << "Название книги:";

cin >> name;

Bib.hideval(name);

break;

case(4):

cout << "Номер:";

cin >> a;

Bib.hiadnum(a);

break;

case(5):

Bib.print();

break;

case(6):

Bib.~Stac();

cout << "Книги удалены" << endl;

break;

case(7): {

Stac a(Bib);

cout << "Исходный стек:" << endl;

Bib.print();

cout << "Новый стек:" << endl;

a.print(); }

break;

default:

cout << "Команда не распознана" << endl;

break;

}

}

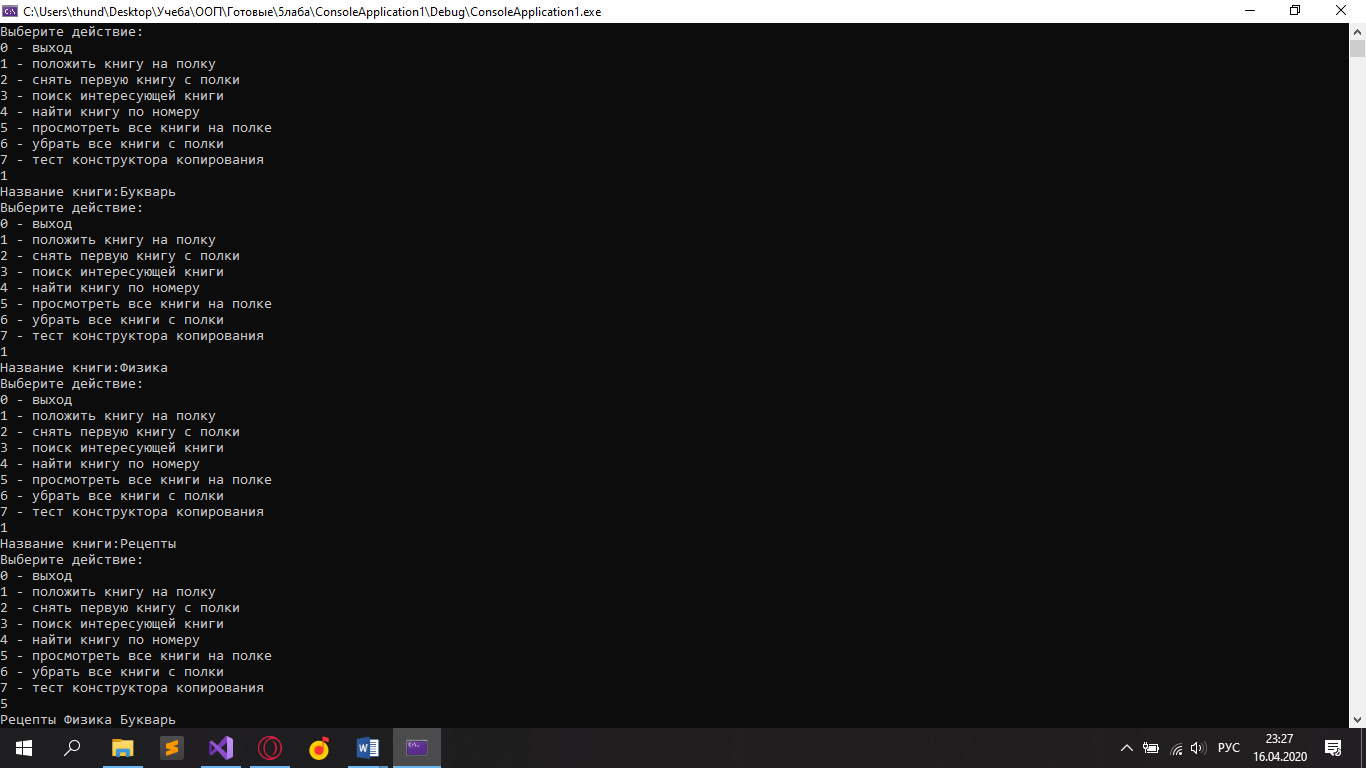
system("pause");

return 0;

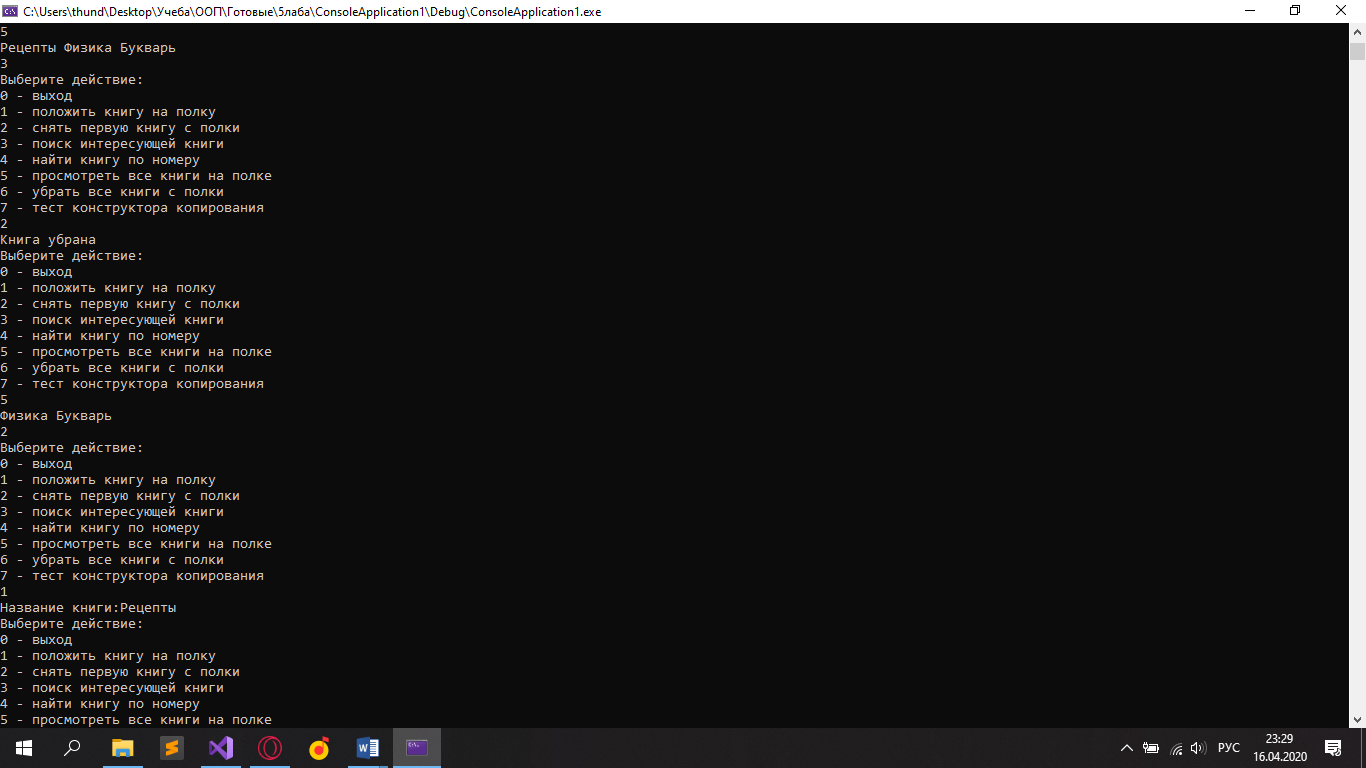
}

**Тесты**:

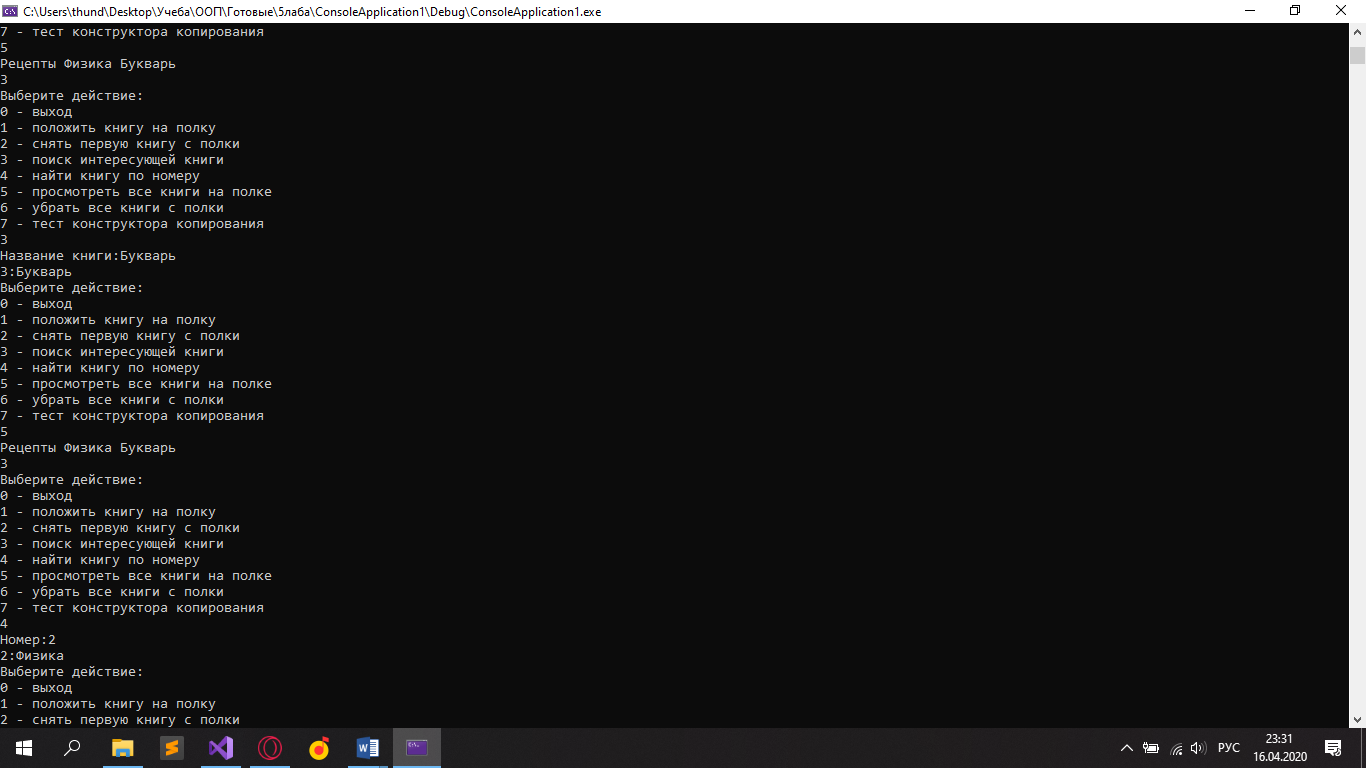
1. Добавление новых объектов



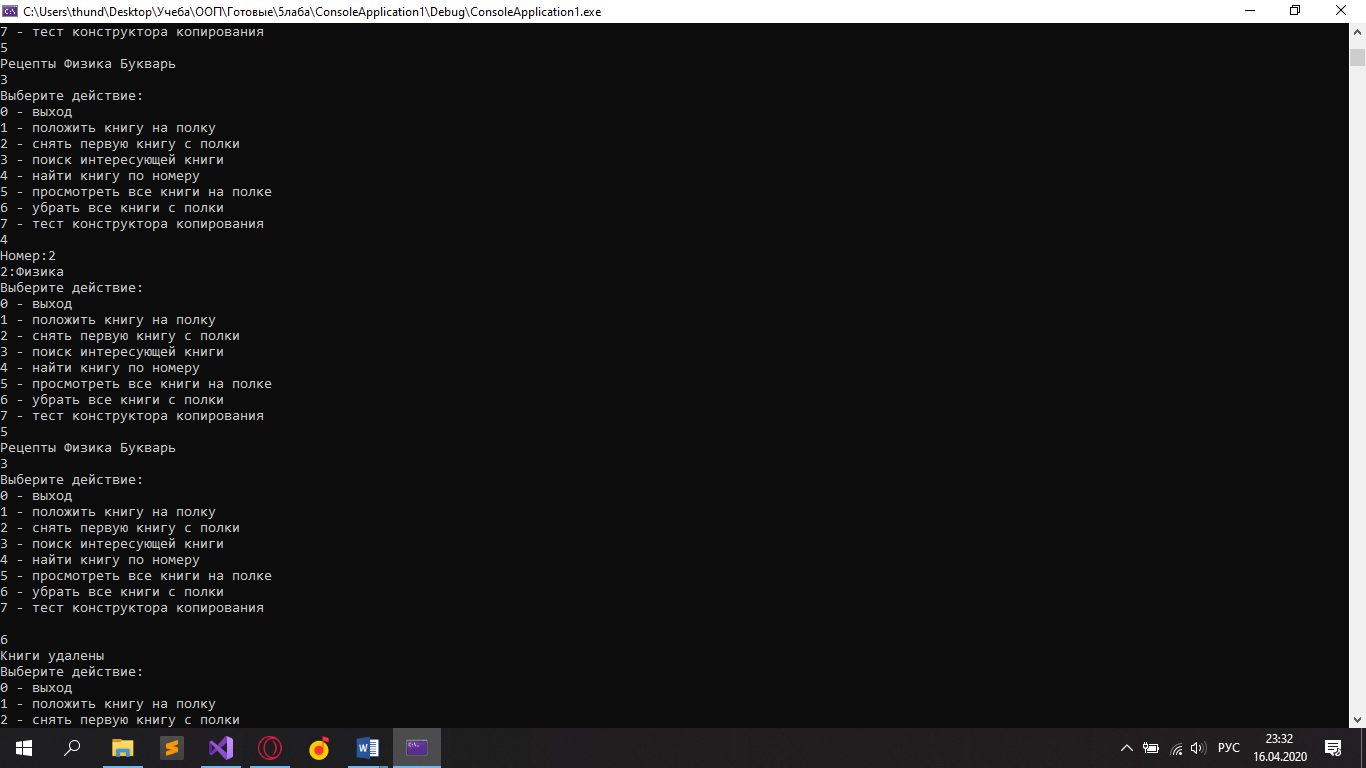
1. Удаление первого объекта на полке



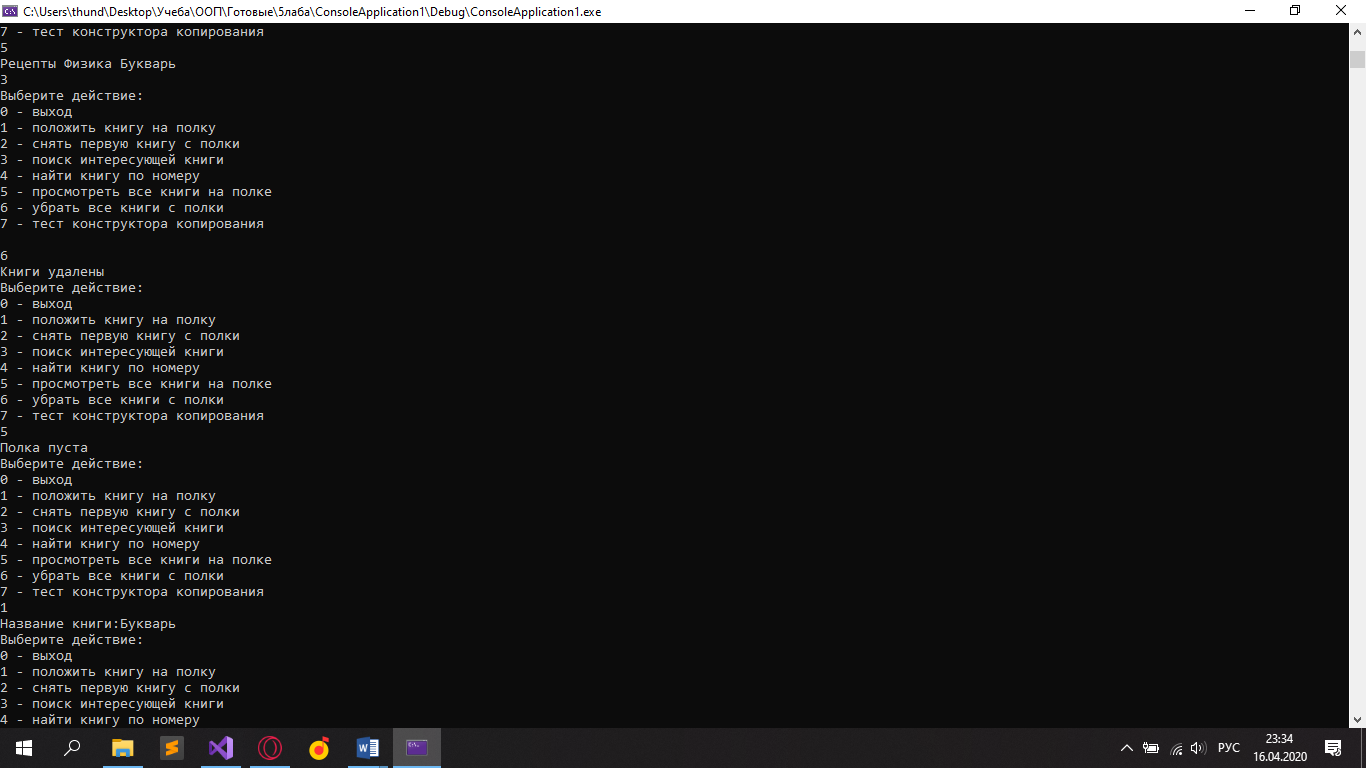
1. Поиск объекта по названию



1. Поиск объекта по номеру на полке



1. Удаление всех объектов



1. Тест конструктора копирования

