# РАЗРАБОТКА АБСТРАКТНЫХ ТИПОВ ДАННЫХ

**Цель работы**: изучить способы определения классов, правила доступа к элементам; приобрести практические навыки работы с объектами класса; изучить принципы и механизмы создания абстрактных типов данных.

**ЗАДАНИЕ**

Разработайте алгоритм и программу, реализующую абстрактный тип данных (АТД) – класс, согласно варианту задания (см. раздел III). Предусмотрите закрытую реализацию и открытый интерфейс. Интерфейс должен содержать псевдоконструкторы и псевдодеструктор, функции присваивания, вывода содержимого и обработки (сортировка, поиск, сравнение, арифметические действия и т.д.).

**РЕШЕНИЕ**

**Код**:

#include <iostream>

#include <Windows.h>

using namespace std;

class Stack {

private://Закрытые элементы класса

struct Node {//Компонент стека

string date;//значение компонента

Node\* Next;//Ссылка на следующий компонент

};

Node\* Head;//"Голова" стека

int size = 0;//Счетчик компонентов стека

public:

void initStac() {//псевдоконструктор

Head = nullptr;

}

void initStac(string a) {// псевдоконструктор копирования

Head = new Node;

Head->date = a;

Head->Next = nullptr;

size++;

}

void deleteStac() {//псевдодиструктор

Node\* current;

while (!emp()) {

current = Head;

Head = Head->Next;

delete(current);

}

}

bool emp() {//проверка на пустоту

return Head == nullptr;

}

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

if (emp()) {

initStac(a);

}

else {

Node\* current = Head;

Head = new Node;

Head->date = 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

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);

Stack Bib;

Bib.initStac();

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;

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.deleteStac();

cout << "Книги удалены" << endl;

break;

}

}

system("pause");

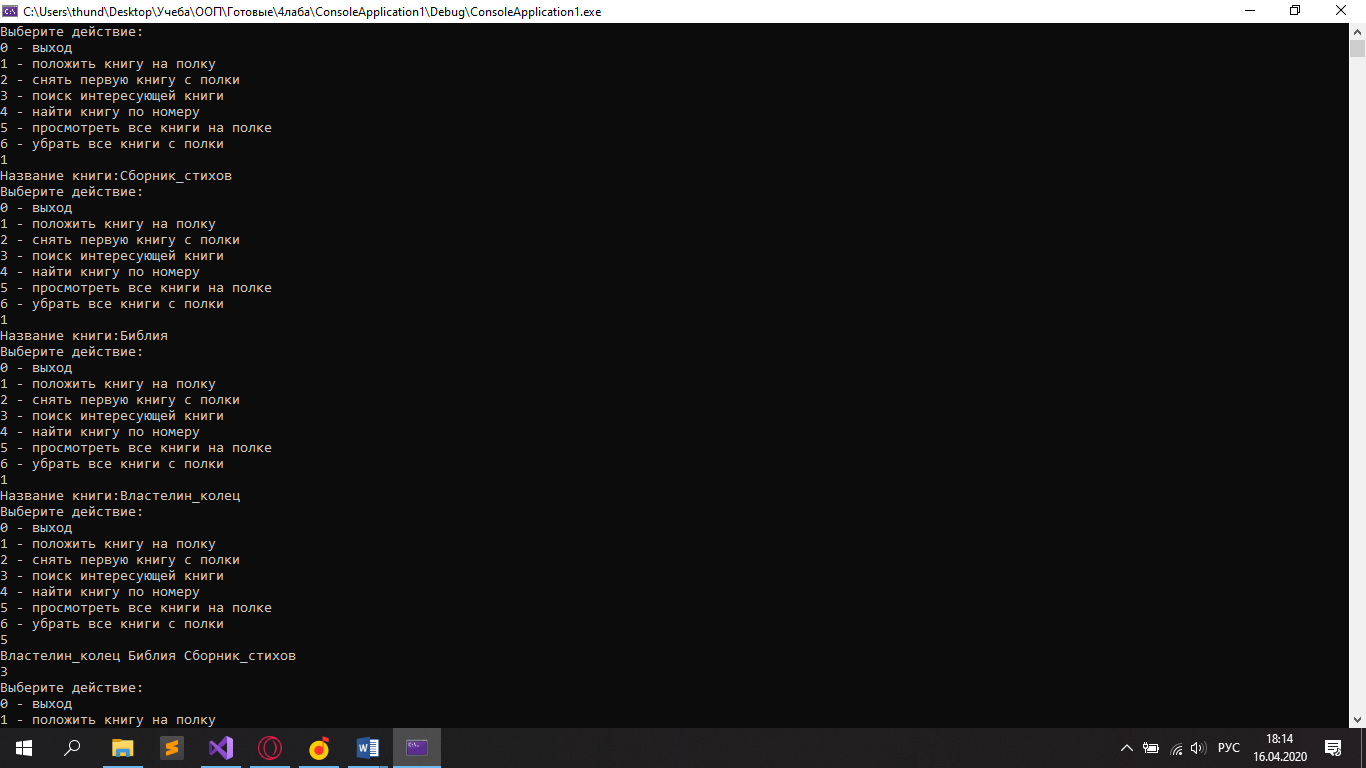
return 0;

;

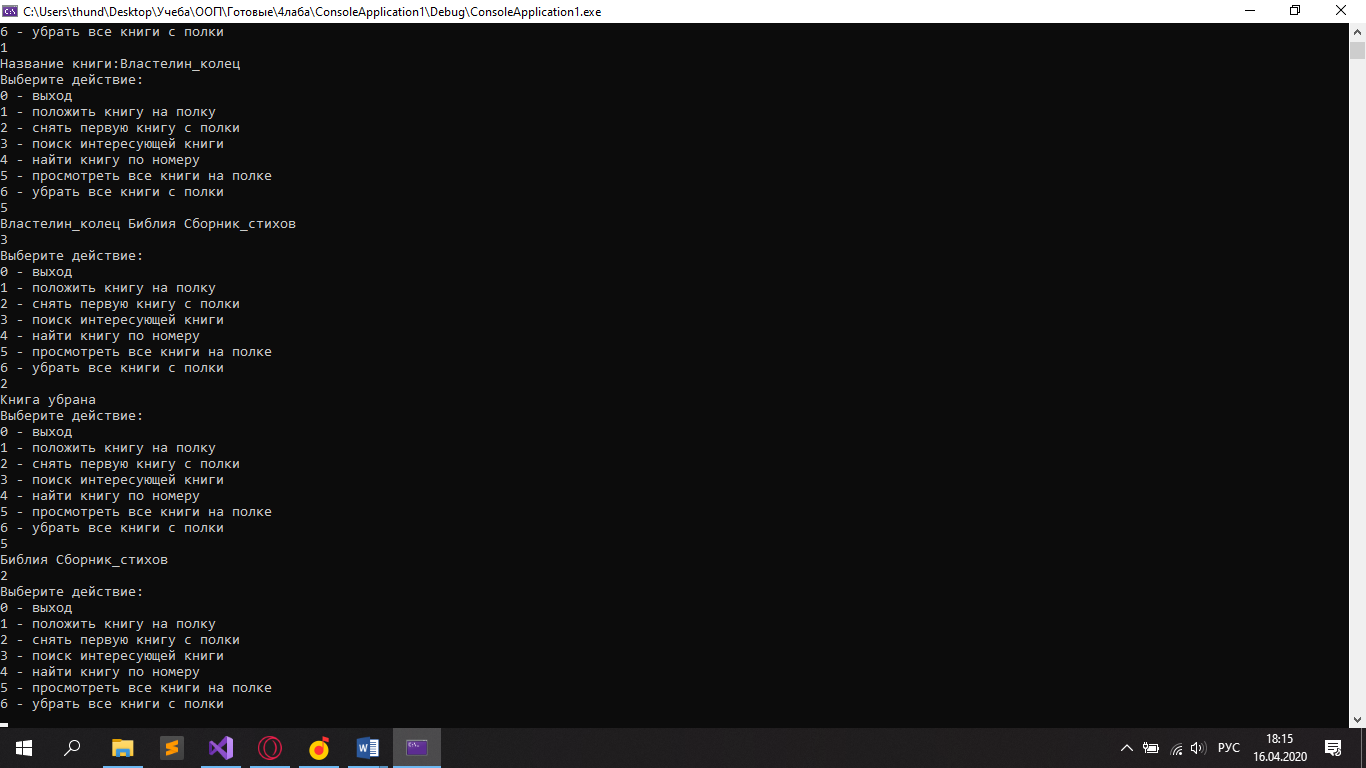
}

**Тесты**:

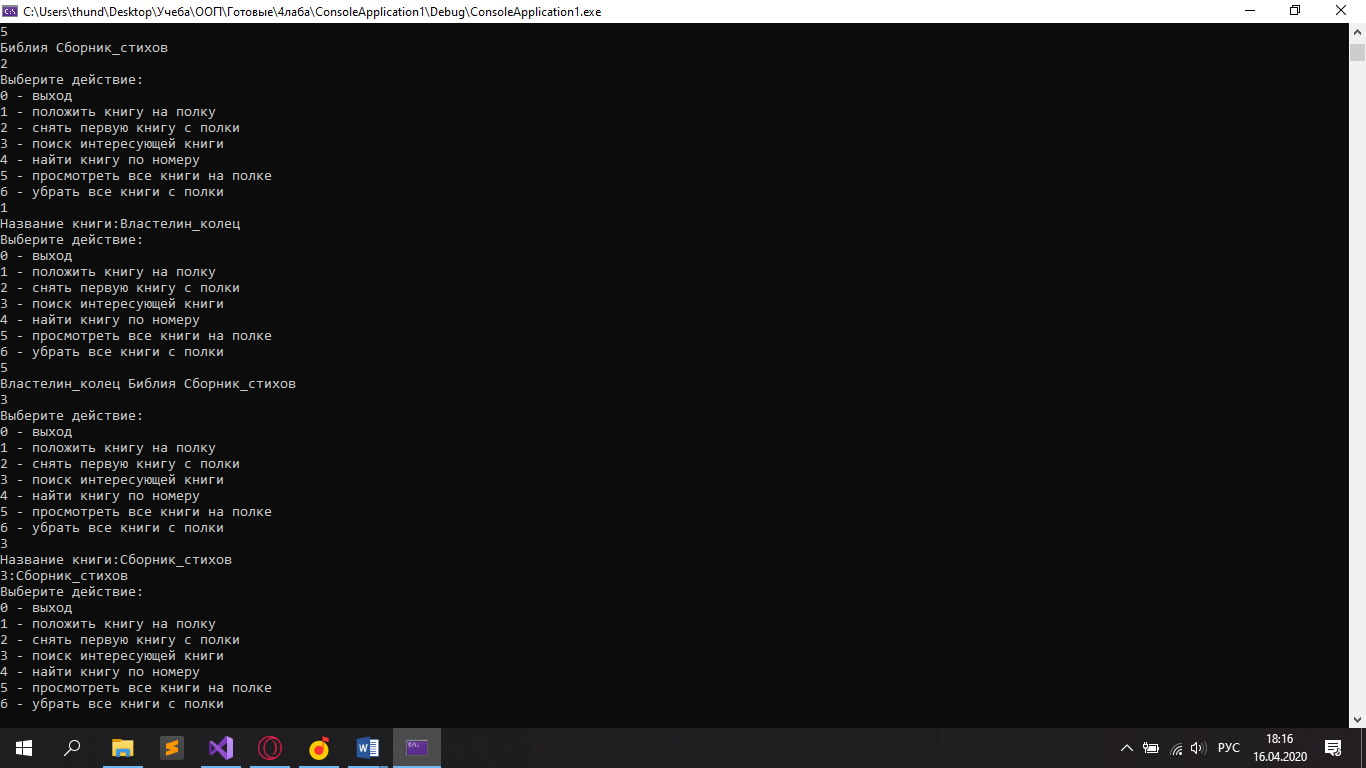
1. Добавление новых объектов



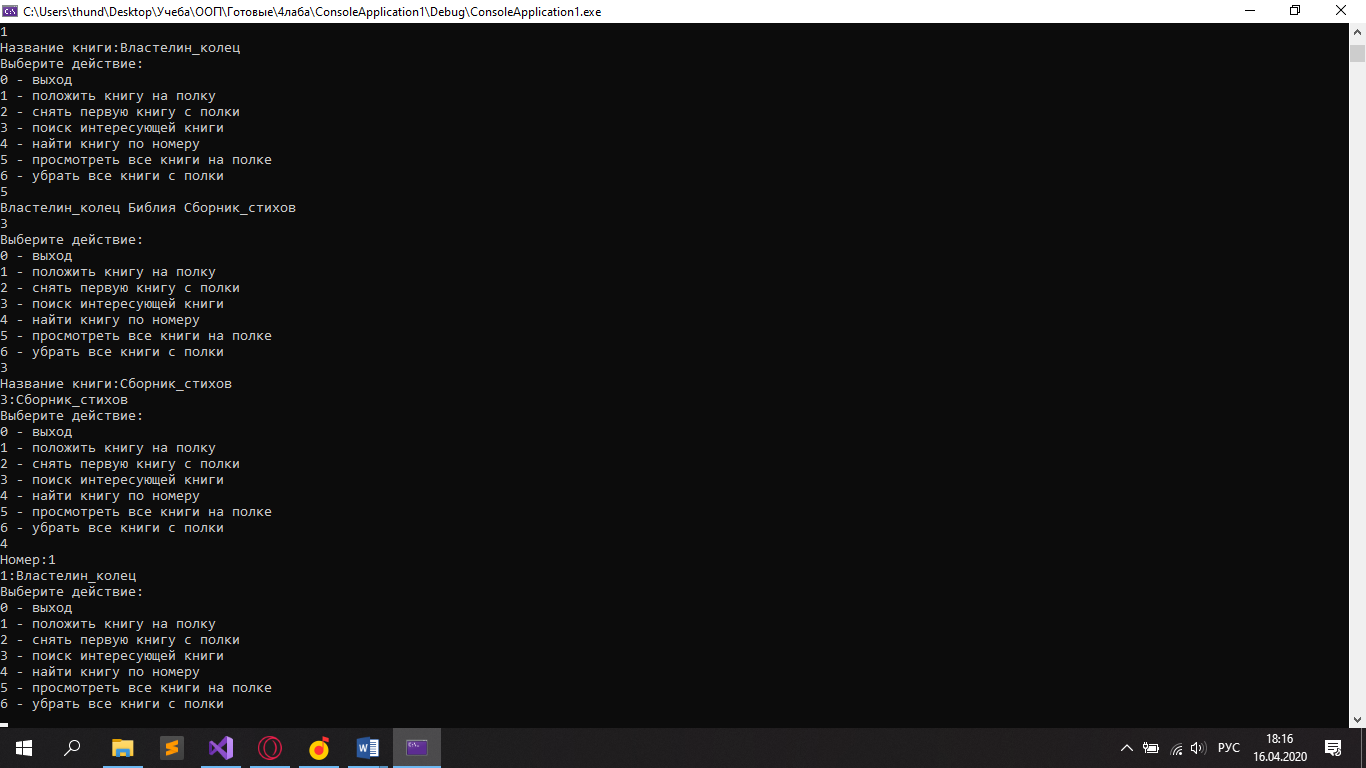
1. Удаление первого объекта на полке



1. Поиск объекта по названию



1. Поиск объекта по номеру на полке



1. Удаление всех объектов

