ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

ФИТ

ИСиИ, 2 группа

Полынская Юлия

2 семестр

Вариант 6

**Лабораторная работа № 1. Указатели на функции**

**Указателем на функцию** называется **переменная**, которая содержит **адрес функции** (адрес точки входа в функцию).

4. В соответствии со своим вариантом *отделить корни* двух уравнений и вычислить их методом *дихотомии* для исходных данных из таблицы, приведенной ниже. Точность вычислений принять равной **e** = 0,001 для всех вариантов.

Операторы метода вычисления корня оформить в виде функции пользователя, уравнения записать также в виде функций пользователя. В основной программе предусмотреть ввод исходных данных, обращение к функции, реализующей метод дихотомии. В процессе выполнения программы определить корни двух уравнений. Использовать *указатель на функцию.*

2 – x2 + x,

sin2(x) + 0,2

#include "stdafx.h"

#include <iostream>

#include <math.h>

using namespace std;

float f1(float x);

float f2(float x);

float dihFunc(float(\*f)(float), float a, float b, float e);

int main()

{

setlocale(LC\_ALL, "Rus");

float a, b, e;

cout << "Введите a: "; cin >> a;

cout << "Введите b: "; cin >> b;

cout << "Введите e: "; cin >> e;

cout << "x1 = " << dihFunc(f1, a, b, e) << endl;

cout << "x2 = " << dihFunc(f2, a, b, e) << endl;

system("pause");

return 0;

}

float f1(float x) {

return pow(x, 3) + 3 \* x - 1;

}

float f2(float x) {

return exp(x) - 4;

}

float dihFunc(float(\*f)(float), float a, float b, float e) {

float x;

do

{

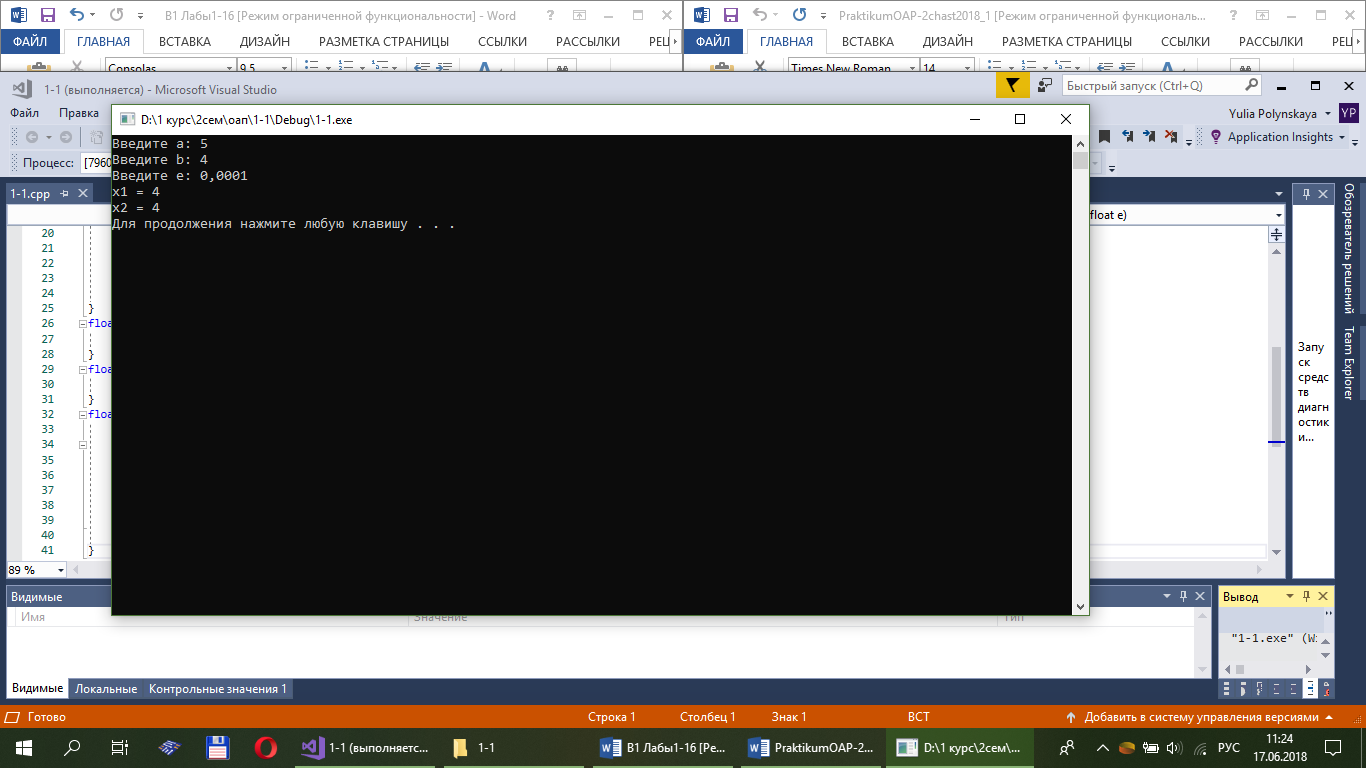
x = (a + b) / 2;

if ((f(a) \* f(x)) <= 0) b = x;

else a = x;

} while (abs(a - b) > 2 \* e);

return x;

}  
  
5. В соответствии со своим вариантом написать программы по условиям задач из таблицы ниже. Программа должна содержать функцию пользователя с *переменным числом параметров* и не менее трех обращений к ней с различным количеством параметров.

Написать функцию **as** с переменным числом параметров, которая находит сумму чисел типа **int** по формуле: ***S=a1\*a2-a2\*a3+a3\*a4-. . . .*** .  
#include "stdafx.h"

#include <iostream>

using namespace std;

void prost(int n, ...);

void FindSimple(int start, int end);

bool IsSimple(int number);

int main()

{

setlocale(LC\_ALL, "Rus");

prost(3, 1, 7,15);

cout << endl;

prost(5, 1, 7, 30, 50, 30);

cout << endl;

prost(7, 1, 7, 30, 50,60,20,40);

system("pause");

return 0;

}

void prost(int n, ...)

{

int \*p = &n;

for (int i = 1; i <= n; i++)

{

if (i + 1 > n)

return;

int start = \*(p + i);

int end = \*(p + i + 1);

if (start > end)

{

cout << i << " интервал:" << start << " - " << end << " не верен" << endl;

continue;

}

cout << i << " интервал:" << start << " - " << end << endl;

FindSimple(start, end);

cout << endl;

}

}

bool IsSimple(int number)

{

for (int i = 2; i <= number / 2; i++)

{

if (number % i == 0)

return false;

}

return true;

}

void FindSimple(int start, int end)

{

for (int i = start + 1; i<end; i++)

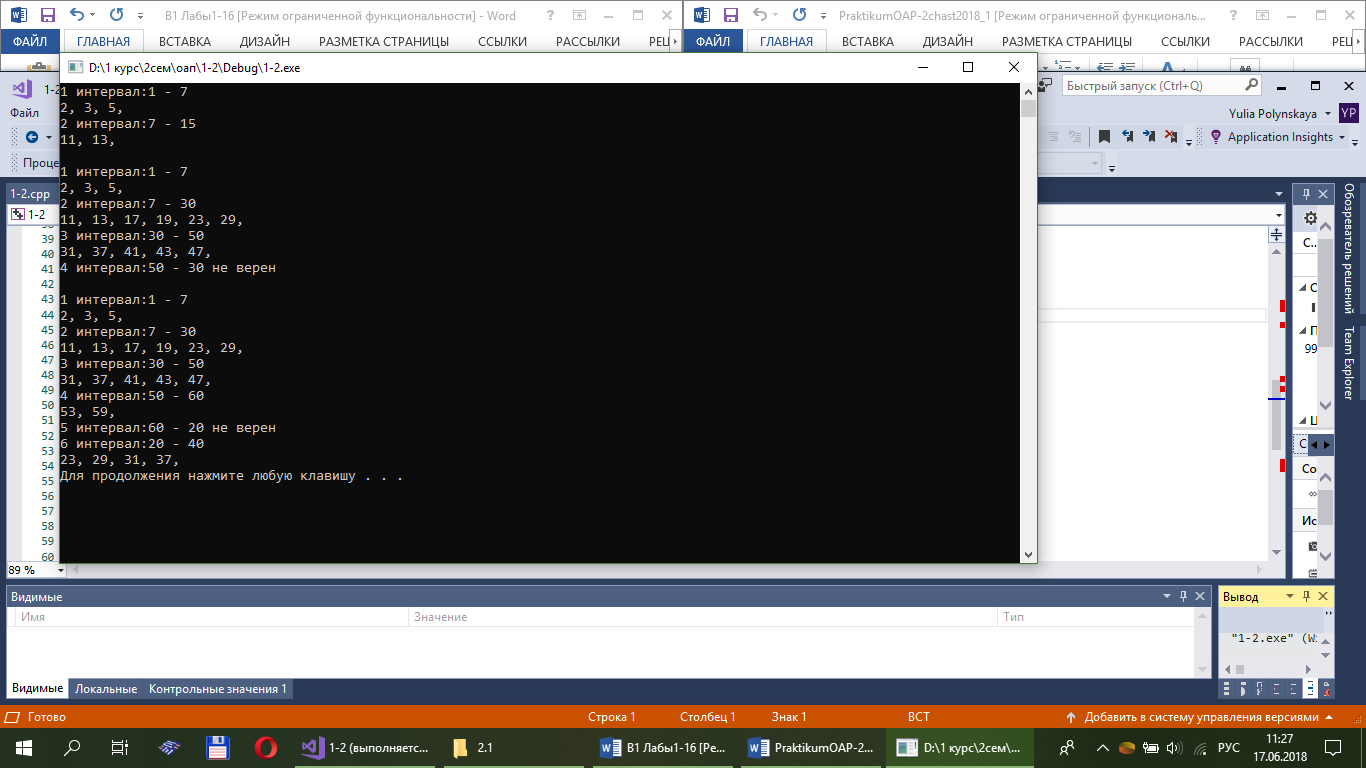
{

if (IsSimple(i))

cout << i << ", ";

}

}

  
**Лабораторная работа № 2. Работа с файлами на языке С**

*Файл* – это набор данных, размещенный на внешнем носителе и рассматриваемый в процессе обработки как единое целое. Каждый файл завершается маркером конца файла (end-of-file marker или EOF) или указанным числом байтов, записанным в служебную структуру данных.

В программу должна быть включена директива **#include <stdio.h>**.  
Файл открывается с помощью функци **fopen**, общий вид которой:

**fopen(const char \*filename, const char \*mode)**

Здесь **filename** − имя файла, **mode** − режим открытия файла.

Режимы открытия файла: **r** − открывает файл для чтения (если файл не существует, то вызов  завершается ошибкой); **w** − открывает пустой файл для записи (если файл существует, его содержимое удаляется); **a** − открывает файл для записи в конец файла (создает файл, если он не существует); **r+** − открывает для чтения и записи в существующий файл; **w+** − открывает пустой файл для чтения и записи (если файл существует, его содержимое удаляется); **a** − открывает файл для чтения и добавления в конец. Могут быть также добавлены символы **t** (открыть файл в текстовом режиме) либо **b** (открыть файл в бинарном режиме).

Используется также функция, общий вид которой:

**fopen\_s( FILE\*\* pFile, const char \*filename, const char \*mode)**, где **pFile** − указатель на файловый указатель.

Файл закрывается с помощью функции **fclose**.

Функции для строкового ввода и вывода: **fgets()** − чтение из файла одной строки текста полностью, если ее длина меньше 50 символов, **fputs()** − запись строки в файл (функция, в отличие от **puts()** сама не помещает в конце строки **'\n'**).  
С помощью функций **fwrite()** и **fread()** осуществляется блоковый ввод/вывод и работа с бинарными файлами.

Для записи информации служит функция:

**fwrite(void** **\*ptr**, **int** **size**, **int** n, **FILE \*pfile);**

где **\*ptr** – указатель на область памяти, в которой размещаются считываемые данные; **size** – размер одного считываемого элемента; **n** – количество считываемых элементов.

Для считывания информации используется функция, общий вид которой:

**fread( void \*ptr, int size, int n, FILE \*pfile);**5. В соответствии со своим вариантом разработать программы для условий, приведенных в таблице ниже, и изучить способы работы с файлами на языке С.

1. Компоненты файла **fA** – вещественные числа (положительные и отрицательные). Определить и вывести на экран порядковый номер того из них, которое наиболее близко к введенному пользователем целому числу.

#include "stdafx.h"

#include <stdio.h>

#include <iostream>

#include <Windows.h>

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

float val;

float \*mass = NULL;

int size = 0;

FILE \*file;

int err = fopen\_s(&file, "some.txt", "r");

if (err != NULL)

{

cout << "Ошибка чтения файла" << err << endl;

int somr;

cin >> somr;

return err;

}

cout << "Числа из файла" << endl;

while (!feof(file))

{

if (fscanf\_s(file, "%f\n", &val))

{

mass = (float\*)realloc(mass, (size + 1) \* sizeof(float));

mass[size] = val;

cout << mass[size] << " ";

size++;

}

}

cout << endl << "Введите вещественное число:" << endl;

float number;

cin >> number;

float nearest = mass[0], diff = abs(mass[0] - number);

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

if (abs(mass[i] - number) < diff) {

nearest = mass[i];

diff = abs(mass[i] - number);

}

}

for (int i = 0; i < size; i++)

if (mass[i] == nearest)

cout << "номер Ближайшего число из файла: " << i + 1 << endl;

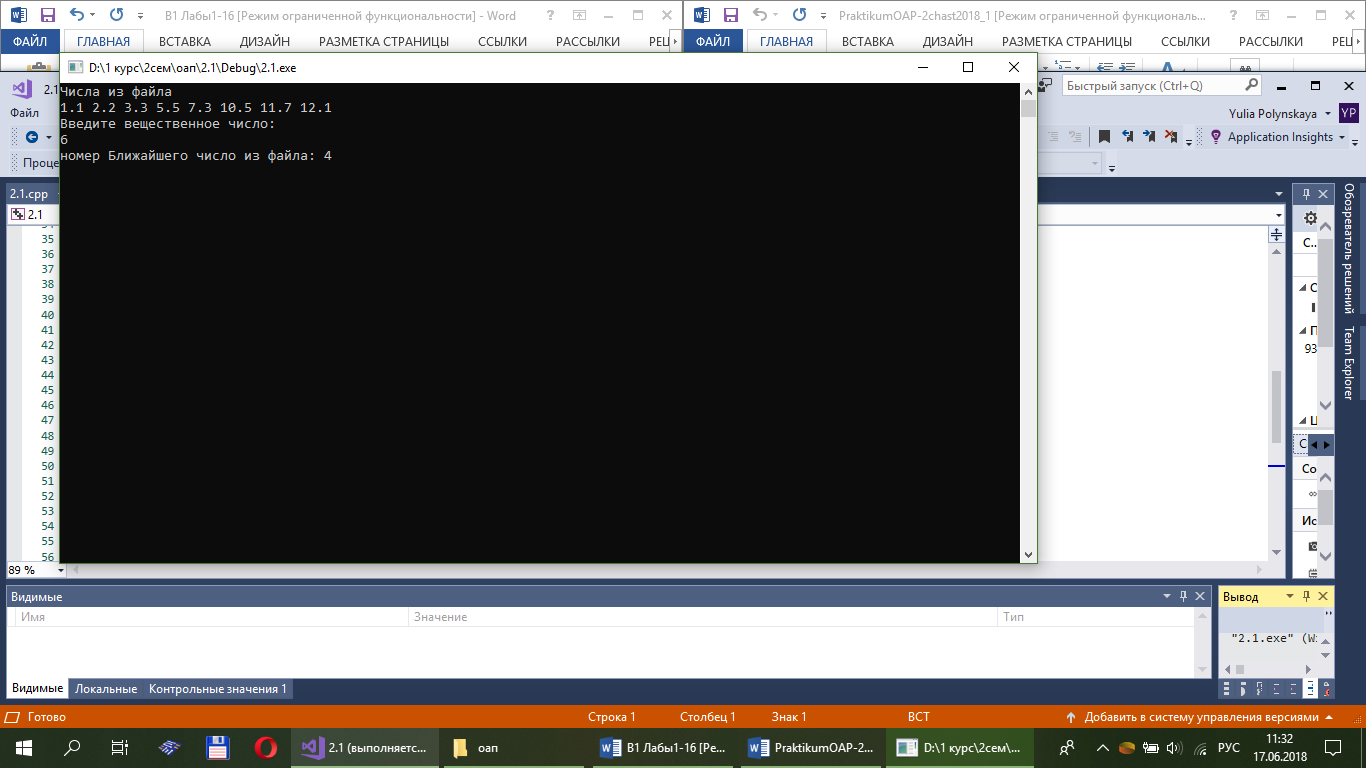
int somr;

cin >> somr;

free(mass);

fclose(file);

return 0;

}  
  
2. Создать текстовый файл **F1** не менее, чем из 6 строк, и записать в него информацию. Скопировать в файл **F2** только четные строки из **F1**.  
#include "stdafx.h"

#include <stdio.h>

#include <iostream>

#include <Windows.h>

#include <cstdio>

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

char str[256] = "";

FILE \*file1, \*file2;

int err = fopen\_s(&file1, "some.txt", "w");

if (err != NULL)

{

cout << "Ошибка чтения файла" << err << endl;

int somr;

cin >> somr;

return err;

}

cout << "введите строки(выход EOF)" << endl;

while (true)

{

gets\_s(str, sizeof(str));

if (strcmp(str, "EOF") == 0)

break;

fputs(str, file1);

fputs("\n", file1);

}

fclose(file1);

err = fopen\_s(&file2, "some2.txt", "w");

if (err != NULL)

{

cout << "Ошибка чтения файла" << err << endl;

int somr;

cin >> somr;

return err;

}

err = fopen\_s(&file1, "some.txt", "r");//открытие для чтения

if (err != NULL)

{

cout << "Ошибка чтения файла" << err << endl;

int somr;

cin >> somr;

return err;

}

int index = 1;

while (!feof(file1))

{

fgets(str, sizeof(str), file1);

if (index % 2 == 0)

fputs(str, file2);

index++;

}

int somr;

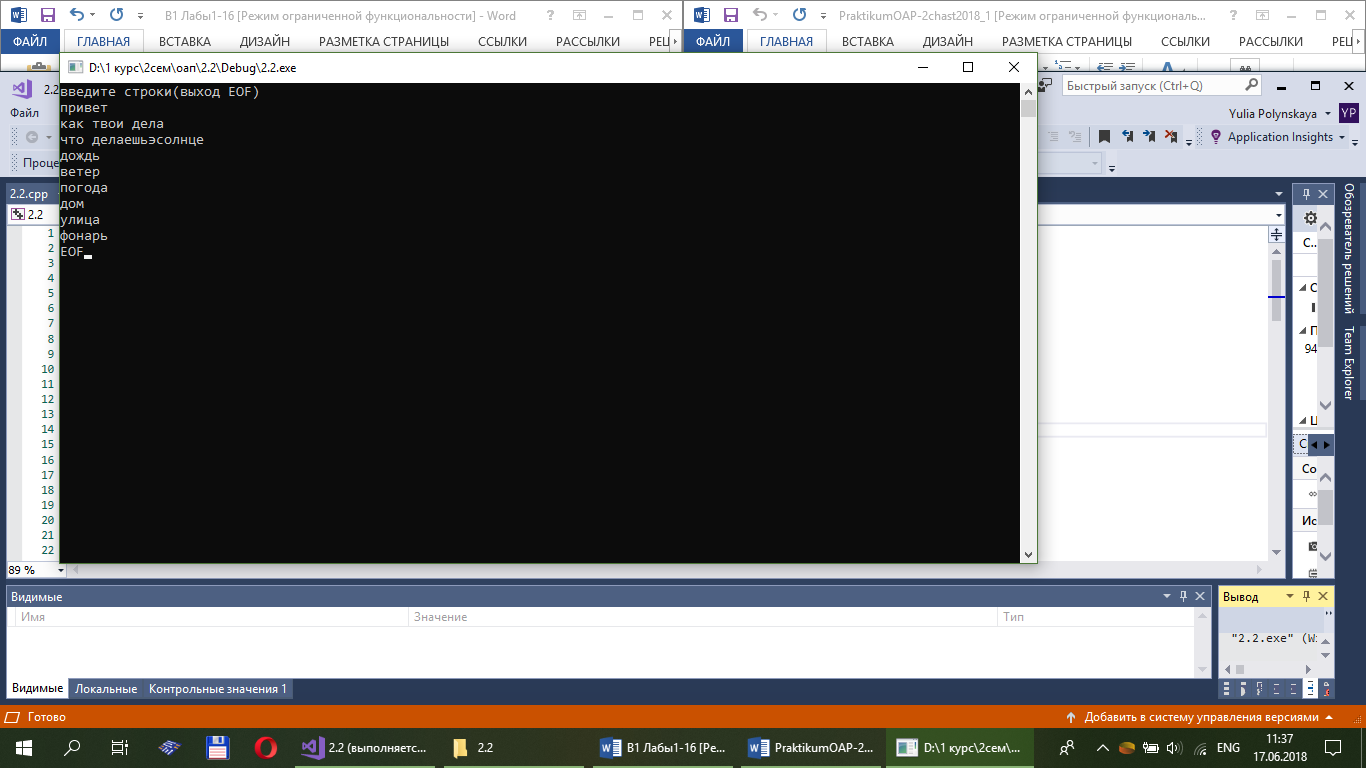
cin >> somr;

fclose(file1);

fclose(file2);

return 0;

}

  
**Лабораторная работа № 3. Работа с файлами на языке С++**

Для обработки файлов на языке C++ в программу должны быть включены заголовочные файлы <**iostream**> и <**fstream**>.

Заголовочный файл <**fstream**> включает определения классов потоков: **ifstream** (для ввода в файл), **ofstream** (для вывода из файла),  **fstream**  (для ввода/вывода). Следует также использовать стандартное пространство имен **− using namespace std;**

Файловый ввод/вывод организован с помощью операций включения (**<<**) и извлечения (**>>**).

5. В соответствии со своим вариантом разработать программы для работы с файлами на языке **С++**. Для первой программы необходимо предварительно создать текстовый файл **FILE1** из нескольких строк и записать в него данные. Во второй программе ввод информации с клавиатуры и вывод в консольное окно должно осуществляться в главной функции, а запись в файл и чтение из файла − в функциях пользователя. Встроенные функции для работы со строками не использовать.  
1. Скопировать из файла **FILE1** в файл **FILE2** строки, начиная с **N** до **K**. Подсчитать количество согласных букв в файле **FILE2**.  
#include "stdafx.h"

#include <iostream>

#include <fstream>

#include <cstdio>

#include <string>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Rus");

int k, n;

FILE \*F1;

fopen\_s(&F1, "1.txt", "r");

ifstream in("1.txt");

ofstream out("2.txt");

cout << "Введите число n= ";

cin >> n;

cout << "Введите число k= ";

cin >> k;

if (!in) {

cout << "Файл fileOne.txt не найден" << endl;

}

if (!out) {

cout << "Файл fileTwo.txt не найден" << endl;

}

int i = 1;

string t;

char sog[] = "бвгджзклмнпрстфхцчшщ";

int answer = 0;

while (getline(in, t))

{

if (i <= k && i >= n && !feof(F1))

{

out << t << endl;

for (unsigned int i = 0; i < t.length(); i++)

{

for (int j = 0; j<strlen(sog); j++)

if (sog[j] == t[i])

{

answer++;

break;

}

}

}

i++;

}

fclose(F1);

in.close();

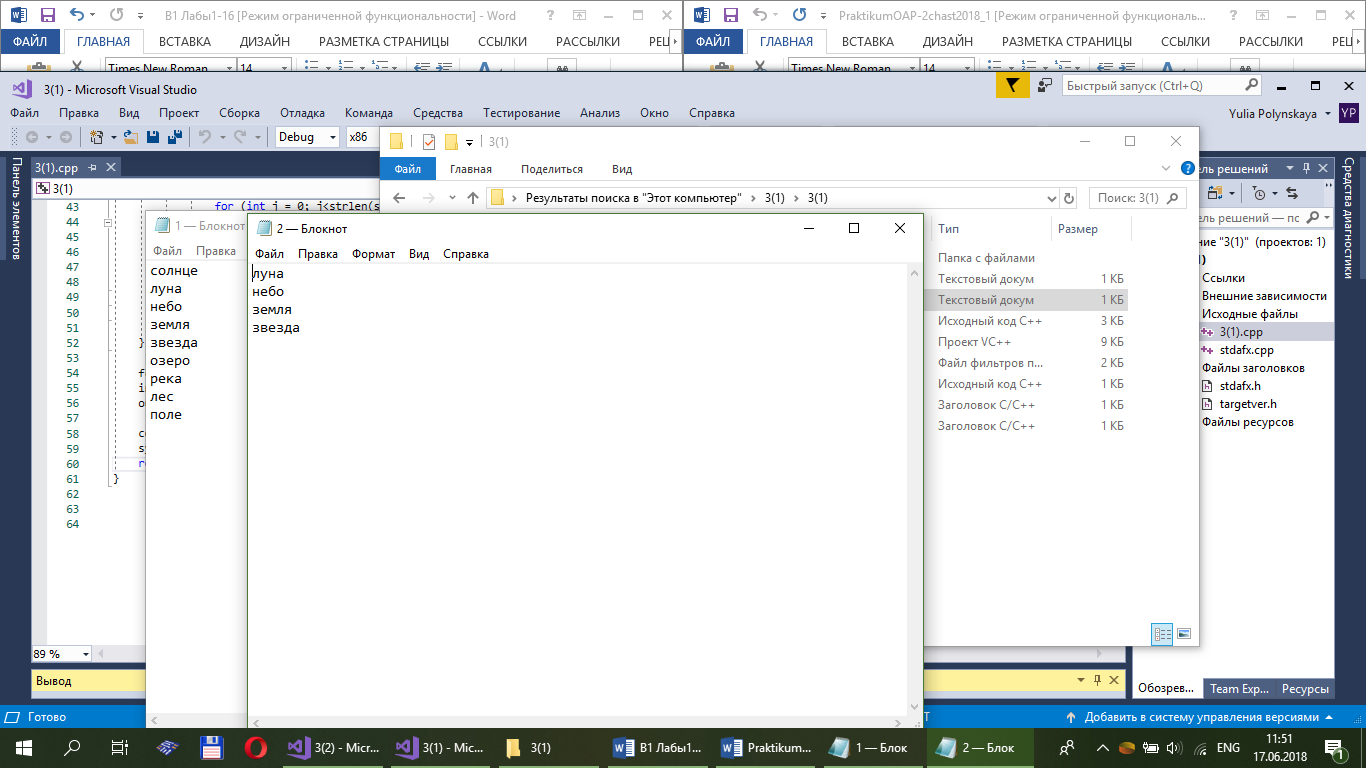
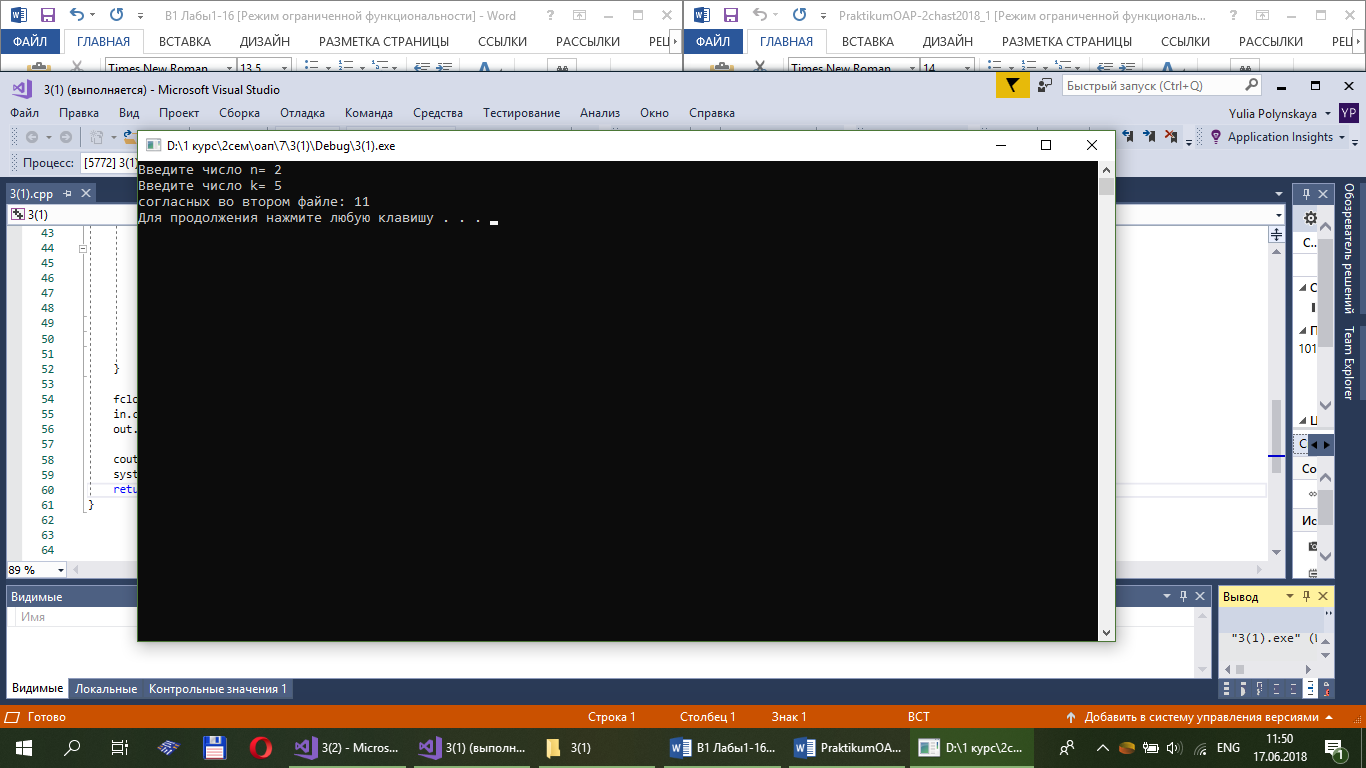
out.close();

cout << "согласных во втором файле: " << answer << endl;

system("pause");

return 0;

}

  
2. Ввести с клавиатуры строку символов, состоящую из групп цифр и нулей, и записать ее в файл. Прочитать из файла данные и вывести на экран группы с четным количеством символов.

#include "stdafx.h"

#include <iostream>

#include <fstream>

#include <cstdio>

#include <string>

using namespace std;

void write(char \*);

char\* read();

int main()

{

setlocale(LC\_ALL, "Rus");

char n[256];

cout << "Введите строку из чисел и нулей: ";

cin >> n;

write(n);

cout << "группы чисел с четным их количеством:" << endl;

char sub[256];

char \*str;

str = read();

if (strlen(str) != 0)

{

int prevZeroIndex = 0;

for (int i = 0; i < strlen(str); i++) {

if (str[i] == '0')

{

if (i != 0 && i != strlen(str))

{

memset(sub, 0, sizeof(sub) / sizeof(sub[0]));

for (int j = prevZeroIndex + 1, k = 0; j < i; j++, k++) {

sub[k] = str[j];

}

if (strlen(sub) % 2 == 0)

{

cout << sub << endl;

str = read();

}

prevZeroIndex = i;

continue;

}

}

}

}

system("pause");

return 0;

}

void write(char \* str) {

ofstream out("4.txt");

if (!out) {

cout << "Файл fileTwo.txt не найден" << endl;

}

out << str;

out.close();

}

char\* read() {

ifstream in("4.txt");

if (!in) {

cout << "Файл fileOne.txt не найден" << endl;

}

char str[256];

in >> str;

in.close();

return str;

## } Лабораторная работа № 4. Представление информации в виде структуры

*Структура* – это пользовательский тип данных, совокупность логически связанных данных различного типа, объединенных под одним идентификатором.

4. В соответствии со своим вариантом разработать программу для данных, приведенных в таблице ниже. Определить структурированный тип, разработать меню для работы с массивом структур.

В программу должны войти функции:

* ввод элементов структуры с клавиатуры;
* вывод элементов структуры в консольное окно;
* удаление заданной структурированной переменной;
* поиск информации;
* запись информации в файл;
* чтение данных из файла  
    
  **Склад.** Наименование товара, цена, количество, процент торговой надбавки (5, 10, 15, 20, 35, 30). Выбор по цене.

#include "stdafx.h"

#include <iostream>

#include <io.h>

#define sizeofWherehouse 256

#define nameLength 256

#define nameOfFile "base.bin"

using namespace std;

void input();

void change();

void print();

void find();

void del();

void save();

void read();

typedef struct Wherehouse

{

char name[nameLength];

float cost;

int count;

int percent;

};

struct Wherehouse ItemsOfWherehouse[sizeofWherehouse];

struct Wherehouse EmptyItem;

FILE \*f;

int currentSize = 0;

int main()

{

setlocale(LC\_CTYPE, "Russian");

int choice;

read();

do

{

system("cls");

cout << "1)Добавление новой записи" << endl;

cout << "2)Изменение записи" << endl;

cout << "3)Вывод записи(ей)" << endl;

cout << "4)Поиск по цене" << endl;

cout << "5)Удаление" << endl;

cout << "0)Выход" << endl << endl;

cout << "Введите номер операции: ";

cin >> choice;

switch (choice)

{

case 1: input(); break;

case 2: change(); break;

case 3: print(); break;

case 4: find(); break;

case 5: del(); break;

case 0: return 0;

default: cout << "попробуйте еще."; system("pause"); system("cls");

}

} while (choice != 0);

return 0;

}

void input()

{

system("cls");

setlocale(LC\_CTYPE, "Russian");

cout << "Ввод информации" << endl;

if (currentSize<sizeofWherehouse)

{

cout << "Запись номер: " << currentSize + 1 << endl;

cout << "Наименование товара: " << endl; cin >> ItemsOfWherehouse[currentSize].name;

cout << "Цена: " << endl; cin >> ItemsOfWherehouse[currentSize].cost;

cout << "Количество: " << endl; cin >> ItemsOfWherehouse[currentSize].count;

cout << "Процент торговой надбавки: " << endl; cin >> ItemsOfWherehouse[currentSize].percent;

currentSize++;

save();

}

else

cout << "Введено максимальное кол-во записей " << endl;

system("pause");

}

void print()

{

system("cls");

read();

setlocale(LC\_CTYPE, "Russian");

int vb, number;

cout << endl << "Введите: " << endl;

cout << "1-вывод одной записи" << endl;

cout << "2-вывод всех записей" << endl;

cin >> vb;

if (vb == 1)

{

cout << "Введите номер записи,которую нужно вывести: " << endl; cin >> number;

cout << "Наименование товара: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << ItemsOfWherehouse[number - 1].name << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Цена: ";

cout << ItemsOfWherehouse[number - 1].cost << endl;

cout << "Количество: ";

cout << ItemsOfWherehouse[number - 1].count << endl;

cout << "Процент торговой надбавки: ";

cout << ItemsOfWherehouse[number - 1].percent << endl;

}

if (vb == 2)

{

for (int i = 0; i < currentSize; i++)

{

cout << "номер товара: " << i + 1 << endl;

cout << "Наименование товара: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << ItemsOfWherehouse[i].name << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Цена: ";

cout << ItemsOfWherehouse[i].cost << endl;

cout << "Количество: ";

cout << ItemsOfWherehouse[i].count << endl;

cout << "Процент торговой надбавки: ";

cout << ItemsOfWherehouse[i].percent << endl;

}

}

system("pause");

system("cls");

}

void del()

{

system("cls");

setlocale(LC\_CTYPE, "Russian");

int n;

cout << "Введите номер записи,которую нужно удалить: " << endl;

cout << "Если удалить все записи,введите " << sizeofWherehouse + 1 << endl; cin >> n;

if (n <= sizeofWherehouse)

{

for (int i = (n - 1); i<currentSize; i++)

ItemsOfWherehouse[i] = ItemsOfWherehouse[i + 1];

currentSize--;

}

else

if (n >= sizeofWherehouse + 1)

{

for (int i = 0; i<sizeofWherehouse; i++)

ItemsOfWherehouse[i] = EmptyItem;

currentSize = 0;

}

save();

system("pause");

}

void find()

{

system("cls");

setlocale(LC\_CTYPE, "Russian");

int c = 0, i = 0;

float costToFind;

cout << "введите цену: " << endl;

cin >> costToFind;

for (int i = 0; i<currentSize; i++)

{

if (ItemsOfWherehouse[i].cost == costToFind)

{

cout << "номер товара: " << i + 1 << endl;

cout << "Наименование товара: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << ItemsOfWherehouse[i].name << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Цена: ";

cout << ItemsOfWherehouse[i].cost << endl;

cout << "Количество: ";

cout << ItemsOfWherehouse[i].count << endl;

cout << "Процент торговой надбавки: ";

cout << ItemsOfWherehouse[i].percent << endl;

cout << endl;

}

}

{ if (ItemsOfWherehouse[i].cost != costToFind)

cout << "Таких товаров нет" << endl; }

system("pause");

system("cls");

}

void change()

{

system("cls");

int n, per;

setlocale(LC\_CTYPE, "Russian");

cout << "Введите номер строки" << endl; cin >> n;

do

{

cout << "Введите: " << endl;

cout << "1)для изменения наименования" << endl;

cout << "2)для изменения цены" << endl;

cout << "3)для изменения количества" << endl;

cout << "4)для изменения процентов" << endl;

cout << "5)отмена\n";

cin >> per;

switch (per)

{

case 1: cout << "наименование ";

cin >> ItemsOfWherehouse[n - 1].name; break;

case 2: cout << "цена ";

cin >> ItemsOfWherehouse[n - 1].cost; break;

case 3: cout << "кол-во ";

cin >> ItemsOfWherehouse[n - 1].count; break;

case 4: cout << "процент(5,10,15,20,25,30) ";

cin >> ItemsOfWherehouse[n - 1].percent; break;

}

} while (per != 5);

save();

system("pause");

system("cls");

}

void save()

{

if (!fopen\_s(&f, nameOfFile, "w"))

{

for (int i = 0; i < currentSize; i++)

fwrite(&ItemsOfWherehouse[i], sizeof(ItemsOfWherehouse[i]), 1, f);

}

fclose(f);

}

void read() {

Wherehouse buf;

Wherehouse temp;

if (!fopen\_s(&f, nameOfFile, "r"))

{

currentSize = 0;

if (\_filelength(\_fileno(f)) != 0)

while (!feof(f))

{

fread(&buf, sizeof(buf), 1, f);

if (strcmp(buf.name, temp.name) != 0)

{

ItemsOfWherehouse[currentSize] = buf;

temp = buf;

currentSize++;

}

}

fclose(f);

}

else

{

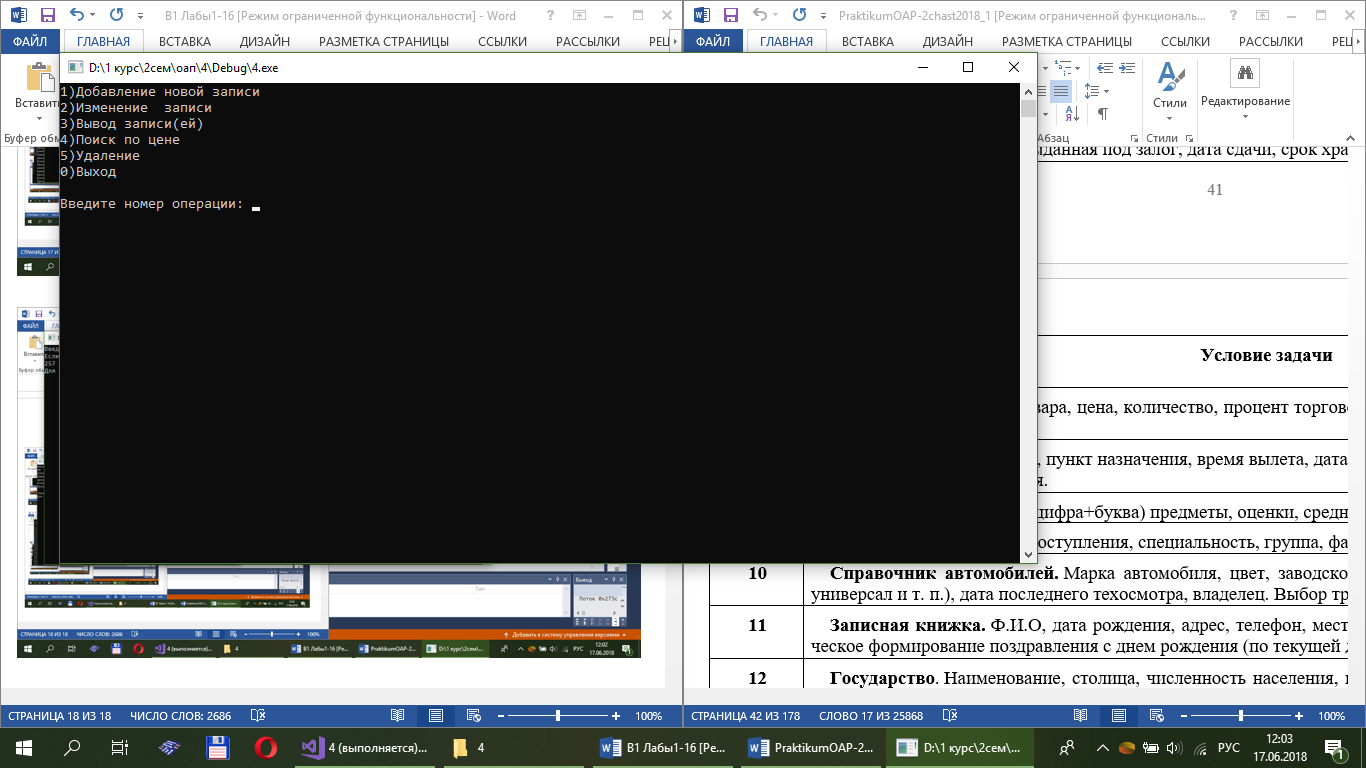
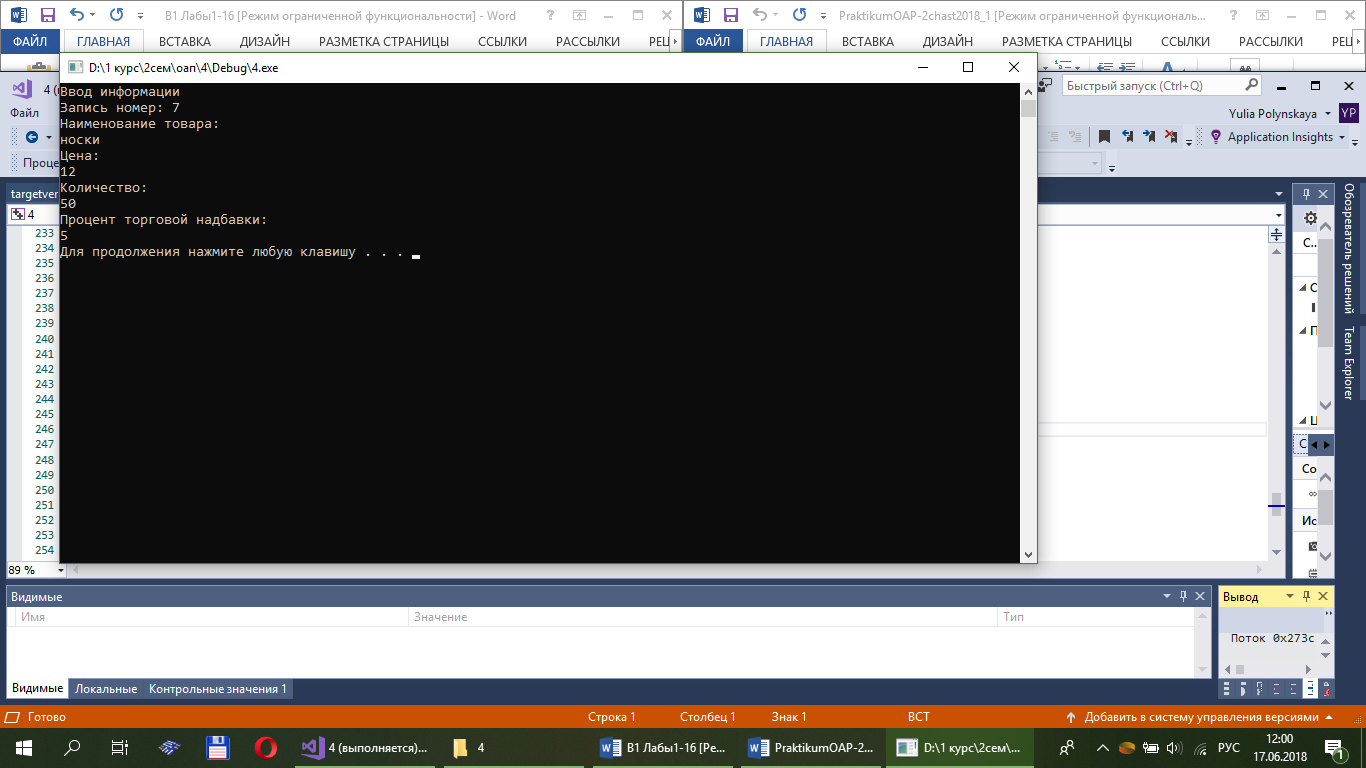
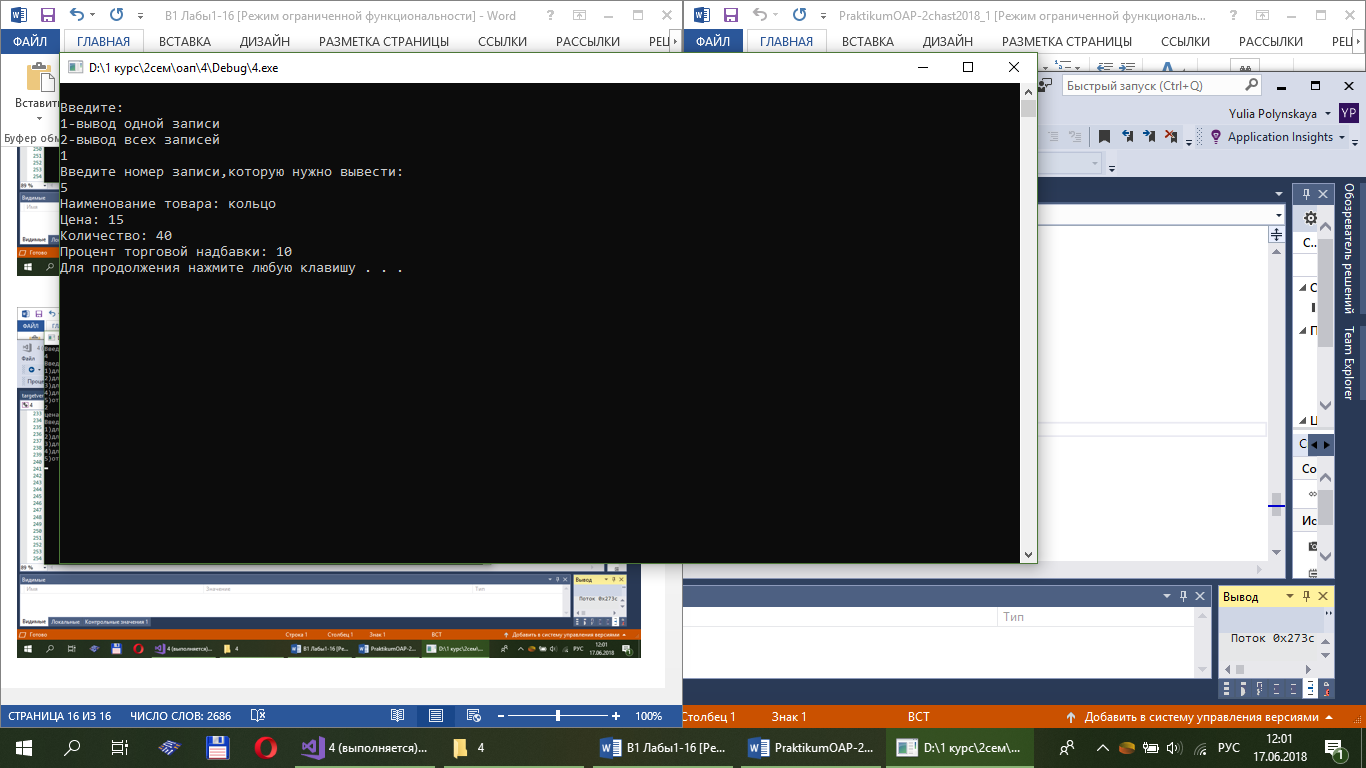
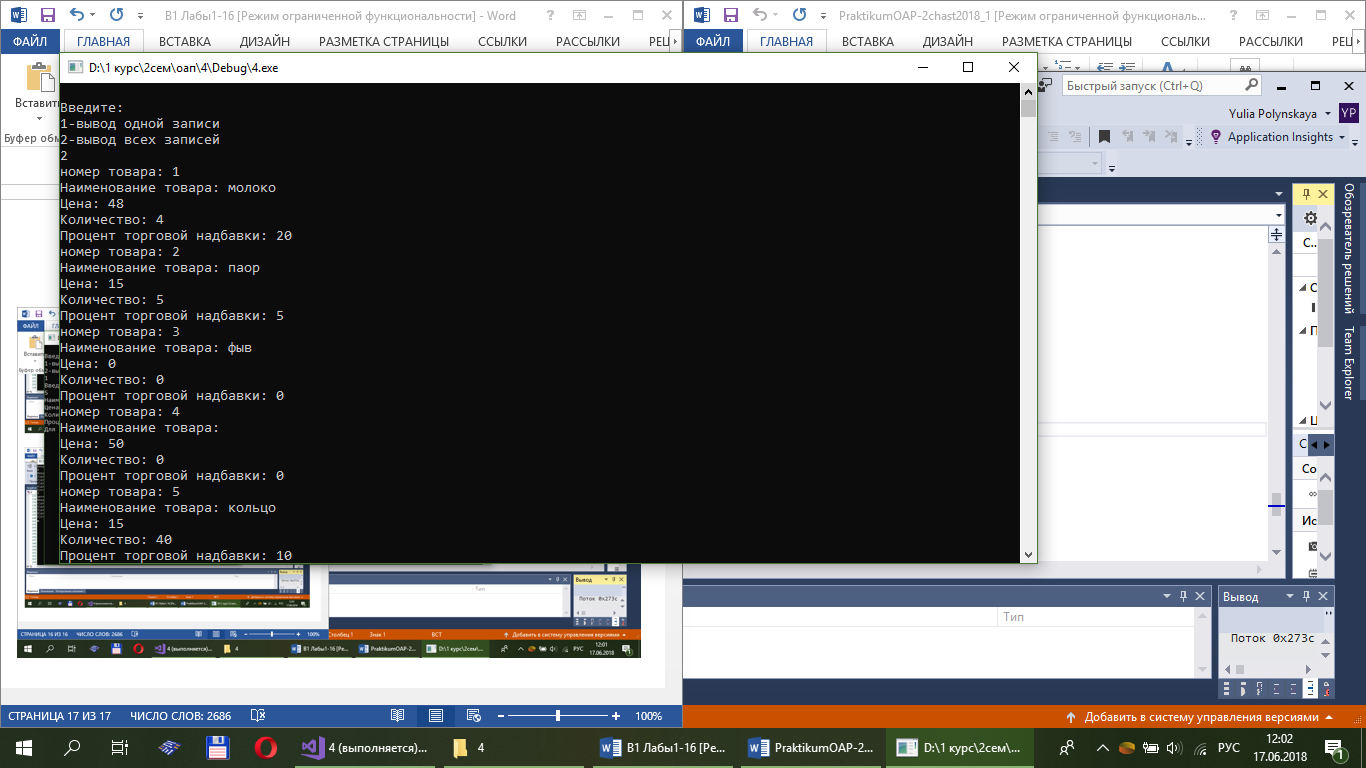
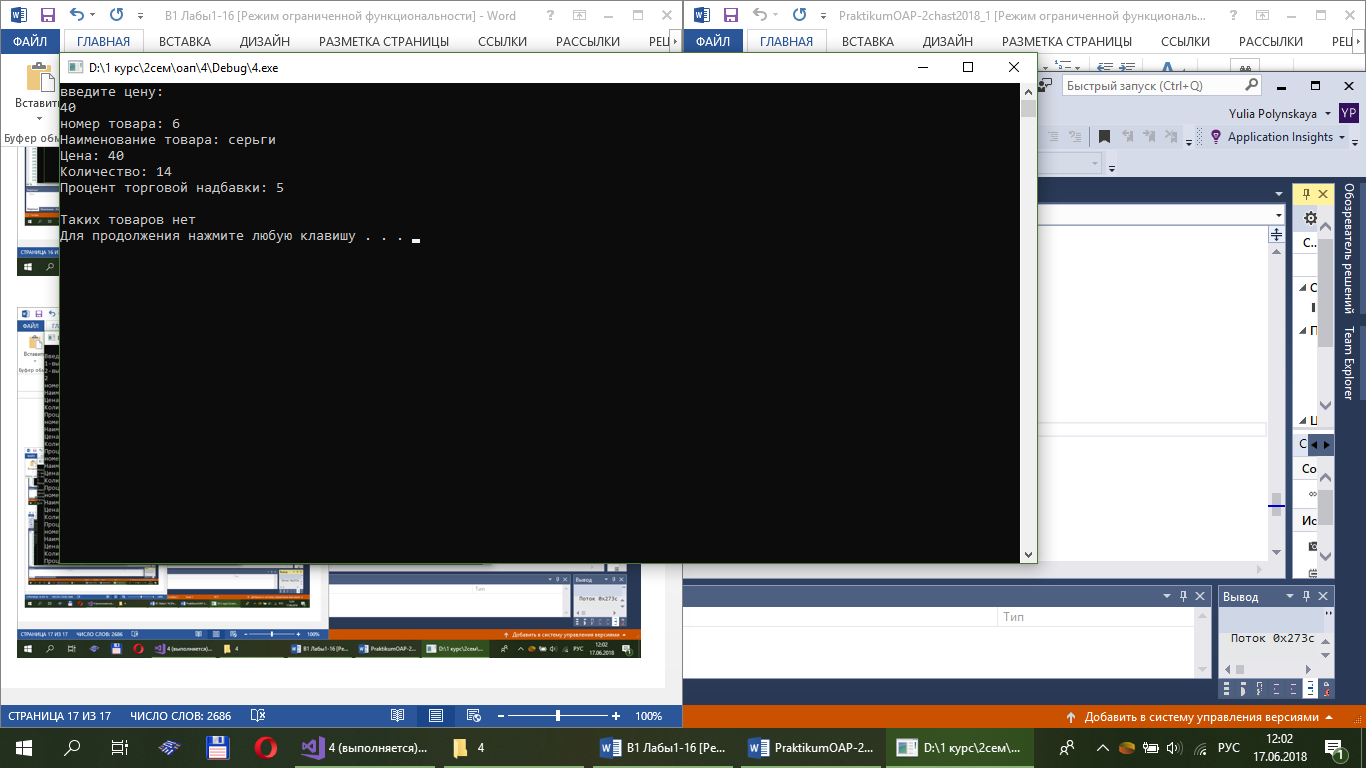
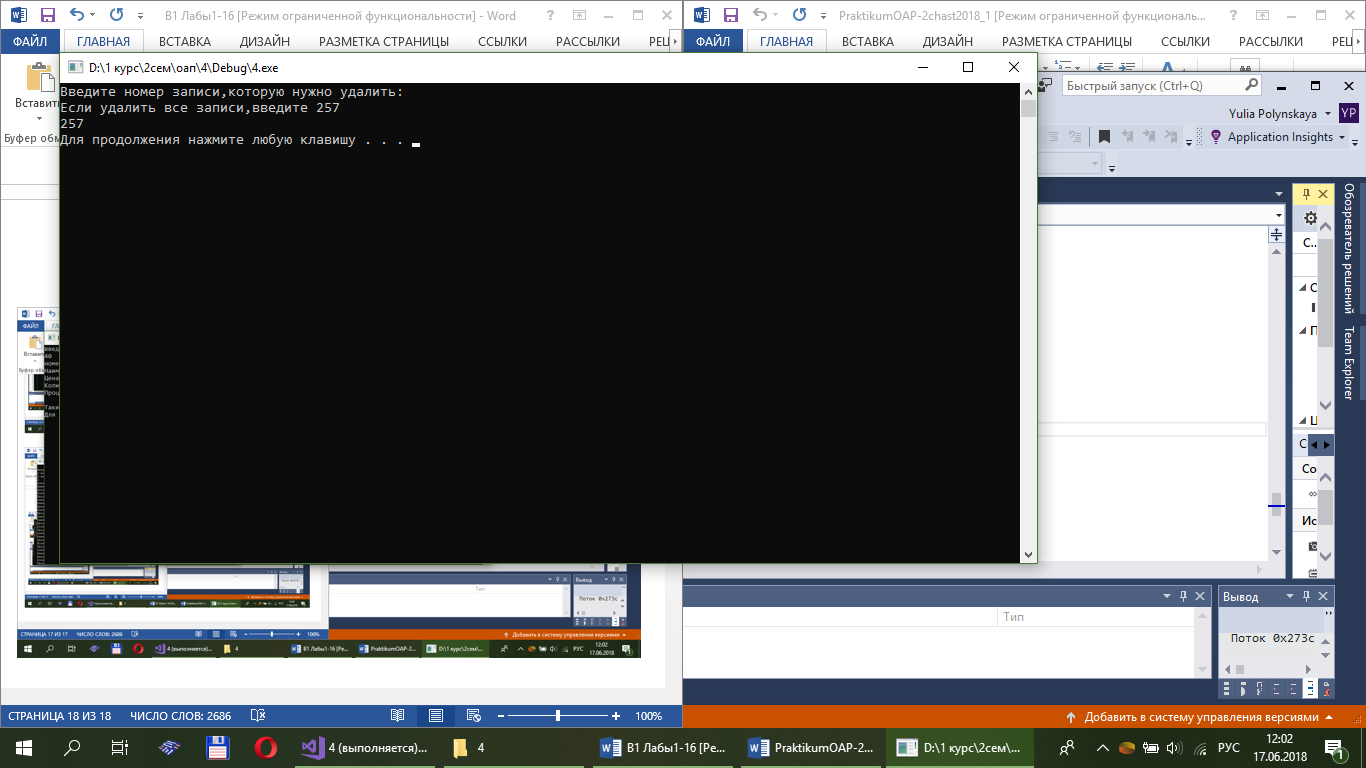
cout << "Ошибка открытия файла или файл пуст, добавте элементы";

system("pause");

system("cls");

return;

}

}  
 **       
**Лабораторная работа № 5. Объединения, перечисления, битовые поля**

*Объединение* – это поименованная совокупность данных разных типов, размещаемых в *одной и той же области памяти*, размер которой достаточен для хранения наибольшего элемента. Объединение подобно структуре, однако в каждый момент времени может использоваться *только один* из элементов объединения.  
*Перечисления* используются в тех случаях, когда данные можно представить в виде нумерованного списка.

В примере создается перечисление **Emonths**, включающее названия месяцев.

Здесь перечисление представляет собой ряд целых чисел от 1 до 12. Например, значение переменной **months**, после того как ей была присвоена константа **December**, стало равным 12.

Поскольку названию каждого месяца соответствует определенное числовое значение, то элементы перечисления могут участвовать в арифметических операциях**.**  Элементом структуры может быть *битовое поле,* обеспечивающее доступ к отдельным битам памяти, которые позволяют рационально использовать память с помощью хранения данных в минимально требуемом количестве битов. Элементы битового поля должны быть объявлены как тип **int** или **unsigned**.

4. В соответствии со своим вариантом разработать программу с использованием *перечислений* и *битовых полей* для работы с данными из таблицы, приведенной ниже. Реализовать функции ввода с клавиатуры, вывода на экран, удаления, поиска элементов. Интерфейс пользователя осуществить в виде меню.

**Справочник автомобилей.** Марка автомобиля, цвет, заводской номер, дата выпуска, тип кузова (седан, универсал и т. п.), дата последнего техосмотра, владелец. Выбор транспортных средств по номеру. Дату выпуска реализовать с помощью битового поля, марку − с помощью перечисления.  
#include "stdafx.h"

#include "windows.h"

#include <iostream>

#include <locale.h>

#define zap 30

#define maxYear 2018

unsigned day, month, year,year1;

using namespace std;

enum Marks {

Ford = 1, Nissan, Opel

} mark;

char marks[][256] = { "Ford","Nissan","Opel" };

struct Date

{

unsigned day : 8;

unsigned months : 8;

unsigned year : 16;

unsigned year1 : 16;

};

struct gibd

{

int marka;

char cvet[10];

int zav;

Date date;

Date teh;

};

struct gibd Skl[zap];

struct gibd bad;

int sch = 0;

void enter()

{

cout << "Ввод информации" << endl;

if (sch<zap)

{

cout << "Запись номер: " << sch + 1 << endl;

cout << "Марка машины(Ford=1,Nissan=2, Opel=3): " << endl;

int mar = 0;

cin >> Skl[sch].marka;

cout << "Цвет машины: " << endl;

cin >> Skl[sch].cvet;

cout << "Заводской номер: " << endl;

cin >> Skl[sch].zav;

cout << "Дата выпуска: " << endl;

cin >> day;

while (day > 31)

{

cout << "число дней не может привышать 31, введите корректное значение: " << endl;

cin >> day;

}

Skl[sch].date.day = day;

cin >> month;

while (month > 12)

{

cout << "число месяцев в году не привышает 12, введите корректное значение: " << endl;

cin >> month;

}

Skl[sch].date.months = month;

cin >> year;

while (year >

maxYear)

{

cout << "год не может быть позже " << maxYear << ", введите корректное значение: " << endl;

cin >> year;

}

Skl[sch].date.year = year;

cout << "Дата последнего техосмотра: " << endl;

cin >> day;

while (day > 31)

{

cout << "число дней не может привышать 31, введите корректное значение: " << endl;

cin >> day;

}

Skl[sch].teh.day = day;

cin >> month;

while (month > 12)

{

cout << "число месяцев в году не привышает 12, введите корректное значение: " << endl;

cin >> month;

}

Skl[sch].teh.months = month;

cin >> year1;

while (year1 <year )

{

cout << "год не может быть раньше " << year << ", введите корректное значение: " << endl;

cin >> year1;

}

Skl[sch].teh.year1 = year1;

sch++;

}

else

cout << "Введено максимальное количество записей" << endl;

}

void out()

{

int vb, nom;

cout << "Введите: " << endl;

cout << "1-вывод одной записи" << endl;

cout << "2-вывод всех записей" << endl;

cin >> vb;

if (vb == 1)

{

cout << "Введите номер записи,которую нужно вывести: " << endl;

cin >> nom;

cout << "Марка машины: ";

cout << marks[Skl[nom - 1].marka] << endl;

cout << "Цвет машины: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[nom - 1].cvet << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Заводской номер: " << Skl[nom - 1].zav << endl;

cout << "Дата выпуска: " << Skl[nom - 1].date.day << "." << Skl[nom - 1].date.months << "." << Skl[nom - 1].date.year << endl;

cout << "Дата последнего техосмотра: " << Skl[nom - 1].teh.day << "." << Skl[nom - 1].teh.months << "." << Skl[nom - 1].teh.year << endl;

}

if (vb == 2)

{

for (int i = 0; i<sch; i++)

{

setlocale(LC\_CTYPE, "Russian");

cout << "Марка машины: " << marks[Skl[i].marka - 1] << endl;

cout << "Цвет машины: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].cvet << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Заводской номер: " << Skl[i].zav << endl;

cout << "Дата выпуска: " << Skl[i].date.day << "." << Skl[i].date.months << "." << Skl[i].date.year << endl;

cout << "Дата последнего техосмотра: " << Skl[i].teh.day << "." << Skl[i].teh.months << "." << Skl[i].teh.year << endl;

}

}

system("pause");

}

void del()

{

int n;

cout << "Введите номер записи,которую нужно удалить: " << endl;

cout << "Если удалить все записи,введите 100" << endl;

cin >> n;

if (n != 100)

{

for (int i = (n - 1); i<sch; i++)

Skl[i] = Skl[i + 1];

sch--;

}

if (n == 100)

{

for (int i = 0; i<zap; i++)

Skl[i] = bad;

}

}

void poisk()

{

int vb, c = 0, i = 0;

int name\_1;

cout << endl << "Выбор по номеру " << endl;

cout << "Введите номер операции" << endl;

cin >> vb;

if (vb == 1)

{

cout << "Введите заводской номер: " << endl;

cin >> name\_1;

for (int i = 0; i<sch; i++)

{

if (Skl[i].zav == name\_1)

{

setlocale(LC\_CTYPE, "Russian");

cout << "Марка машины: " << marks[Skl[i].marka] << endl;

cout << "Цвет машины: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].cvet << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Заводской номер: " << Skl[i].zav << endl;

cout << "Дата выпуска: " << Skl[i].date.day << "." << Skl[i].date.months << "." << Skl[i].date.year << endl;

cout << "Дата последнего техосмотра: " << Skl[i].teh.day << "." << Skl[i].teh.months << "." << Skl[i].teh.year << endl;

break;

}

}

}

system("pause");

}

void main()

{

setlocale(LC\_CTYPE, "Russian");

int r;

do

{

system("cls");

cout << "Записей пока нет" << endl;

cout << "Введите:" << endl;

cout << "1-для ввода новой записи" << endl;

cout << "2-для вывода записи(ей)" << endl;

cout << "3-для удаления записи(ей)" << endl;

cout << "4-для поиска информации" << endl;

cout << "5-для выхода" << endl;

cin >> r;

switch (r)

{

case 1: enter(); break;

case 2: out(); break;

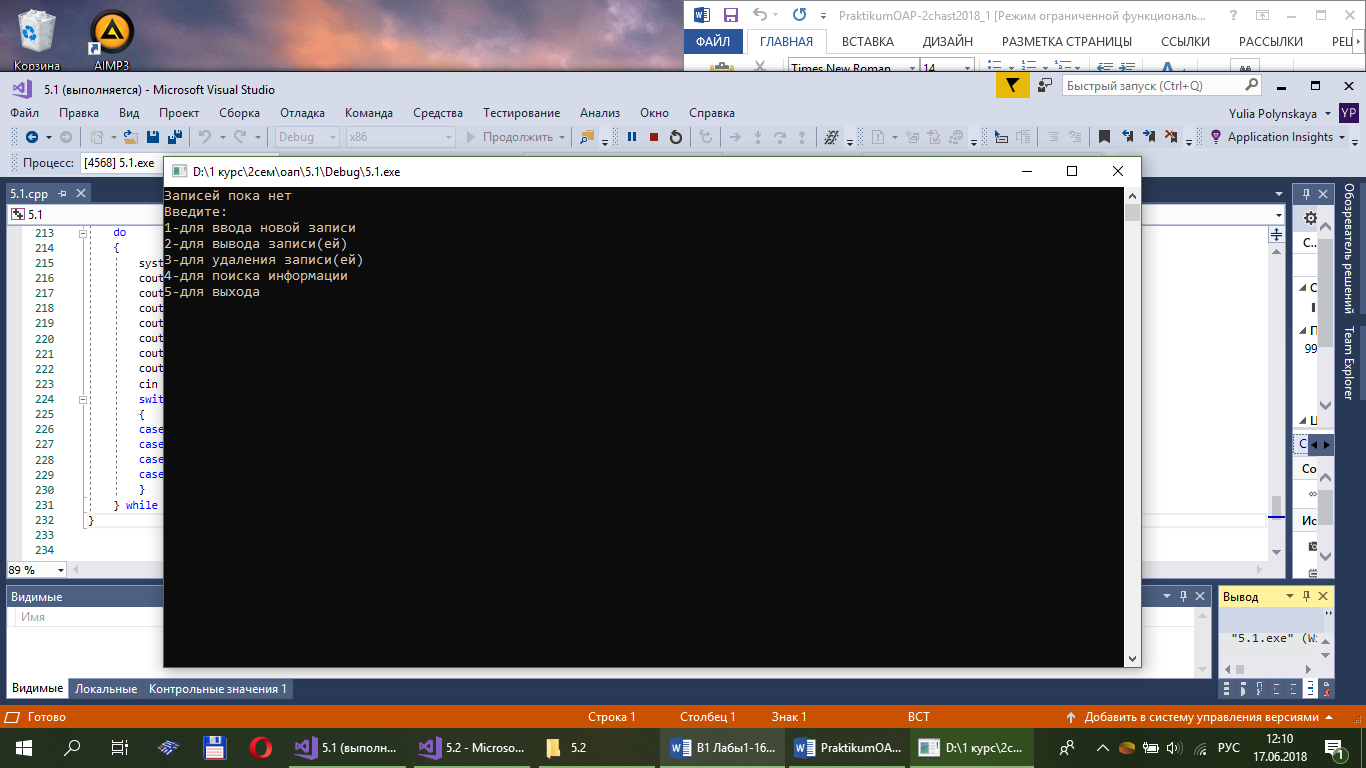
case 3: del(); break;

case 4: poisk(); break;

}

} while (r != 5);

}

  
**Отдел кадров**. Ф.И.О. работника, образование, специальность, подразделение, должность, оклад, дата поступления на предприятие. Выбор по должности.  
#include "stdafx.h"

#include <iostream>

#include <string>

#define zap 30

#define maxYear 2018

using namespace std;

unsigned day, month, year;

union Utypes //объединение

{

int hh;

int choice;

}types;

struct gibd

{

char fio[255];

char obr[255];

struct Date //битовое

{

unsigned day : 8;

unsigned month : 8;

unsigned year : 16;

} date;

char spec[255];

char podrazd[255];

char dolgnost[255];

int oklad;

};

struct gibd Skl[zap];

struct gibd bad;

int sch = 0;

void enter()

{

cout << "Ввод информации" << endl;

if (sch<zap)

{

cout << "Запись номер: " << sch + 1 << endl;

cout << "ФИО работника: " << endl;

cin >> Skl[sch].fio;

cout << "Образование: " << endl;

cin >> Skl[sch].obr;

cout << "Специалность: " << endl;

cin >> Skl[sch].spec;

cout << "Подразделение: " << endl;

cin >> Skl[sch].podrazd;

cout << "Должность: " << endl;

cin >> Skl[sch].dolgnost;

cout << "Оклад: " << endl;

cin >> Skl[sch].oklad;

cout << "Дата поступления на предприятие: " << endl;

cin >> types.hh;

while (types.hh > 31)

{

cout << "число дней не может привышать 31, введите корректное значение: " << endl;

cin >> types.hh;;

}

Skl[sch].date.day = types.hh;

cin >> types.hh;

while (types.hh > 12)

{

cout << "число месяцев в году не привышает 12, введите корректное значение: " << endl;

cin >> types.hh;

}

Skl[sch].date.month = types.hh;

cin >> types.hh;

while (types.hh > maxYear)

{

cout << "год не может быть позже " << maxYear << ", введите корректное значение: " << endl;

cin >> types.hh;

}

Skl[sch].date.year = types.hh;

sch++;

}

else

cout << "Введено максимальное количество записей" << endl;

system("pause");

}

void out()

{

int nom;

cout << "Введите: " << endl;

cout << "1-вывод одной записи" << endl;

cout << "2-вывод всех записей" << endl;

cin >> types.choice;

if (types.choice == 1)

{

cout << "Введите номер записи,которую нужно вывести: " << endl;

cin >> nom;

setlocale(LC\_CTYPE, "Russian");

cout << "Запись номер: " << nom << endl;

cout << "ФИО работника: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[nom - 1].fio << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Образование: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[nom - 1].obr << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Специалность: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[nom - 1].spec << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Подразделение: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[nom - 1].podrazd << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Должность: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[nom - 1].dolgnost << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Оклад: " << Skl[nom - 1].oklad << endl;

cout << "Дата поступления на предприятие: "

<< Skl[nom - 1].date.day << "." << Skl[nom - 1].date.month << "." << Skl[nom - 1].date.year << endl;

system("pause");

}

if (types.choice == 2)

{

for (int i = 0; i<sch; i++)

{

setlocale(LC\_CTYPE, "Russian");

cout << "Запись номер: " << i + 1 << endl;

cout << "ФИО работника: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].fio << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Образование: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].obr << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Специалность: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].spec << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Подразделение: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].podrazd << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Должность: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].dolgnost << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Оклад: " << Skl[i].oklad << endl;

cout << "Дата поступления на предприятие: "

<< Skl[i].date.day << "." << Skl[i].date.month << "." << Skl[i].date.year << endl;

}

system("pause");

}

}

void del()

{

int n;

cout << "Введите номер записи,которую нужно удалить: " << endl;

cout << "Если удалить все записи,введите 100" << endl;

cin >> n;

if (n != 100)

{

for (int i = (n - 1); i<sch; i++)

Skl[i] = Skl[i + 1];

sch--;

}

if (n == 100)

{

for (int i = 0; i<zap; i++)

Skl[i] = bad;

}

system("pause");

}

void poisk()

{

int c = 0, i = 0;

char name\_1[100];

cout << endl << "Введите: " << endl;

cout << "Выбор по должности номер:" << endl;

cin >> types.choice;

if (types.choice == 1)

{

cout << "Введите должность: " << endl;

cin >> name\_1;

for (int i = 0; i<sch; i++)

{

if (strcmp(Skl[i].dolgnost, name\_1) == 0)

{

setlocale(LC\_CTYPE, "Russian");

cout << "Запись номер: " << i + 1 << endl;

cout << "ФИО работника: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].fio << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Образование: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].obr << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Специалность: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].spec << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Подразделение: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].podrazd << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Должность: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << Skl[i].dolgnost << endl;

setlocale(LC\_CTYPE, "Russian");

cout << "Оклад: " << Skl[i].oklad << endl;

cout << "Дата поступления на предприятие: "

<< Skl[i].date.day << "." << Skl[i].date.month << "." << Skl[i].date.year << endl;

}

}

}

system("pause");

}

int main()

{

setlocale(LC\_CTYPE, "Russian");

do

{

system("cls");

cout << "Введите:" << endl;

cout << "1-для ввода новой записи" << endl;

cout << "2-для вывода записи(ей)" << endl;

cout << "3-для удаления записи(ей)" << endl;

cout << "4-для поиска информации" << endl;

cout << "5-для выхода" << endl;

cin >> types.choice;

switch (types.choice)

{

case 1: system("cls"); enter(); break;

case 2: system("cls"); out(); break;

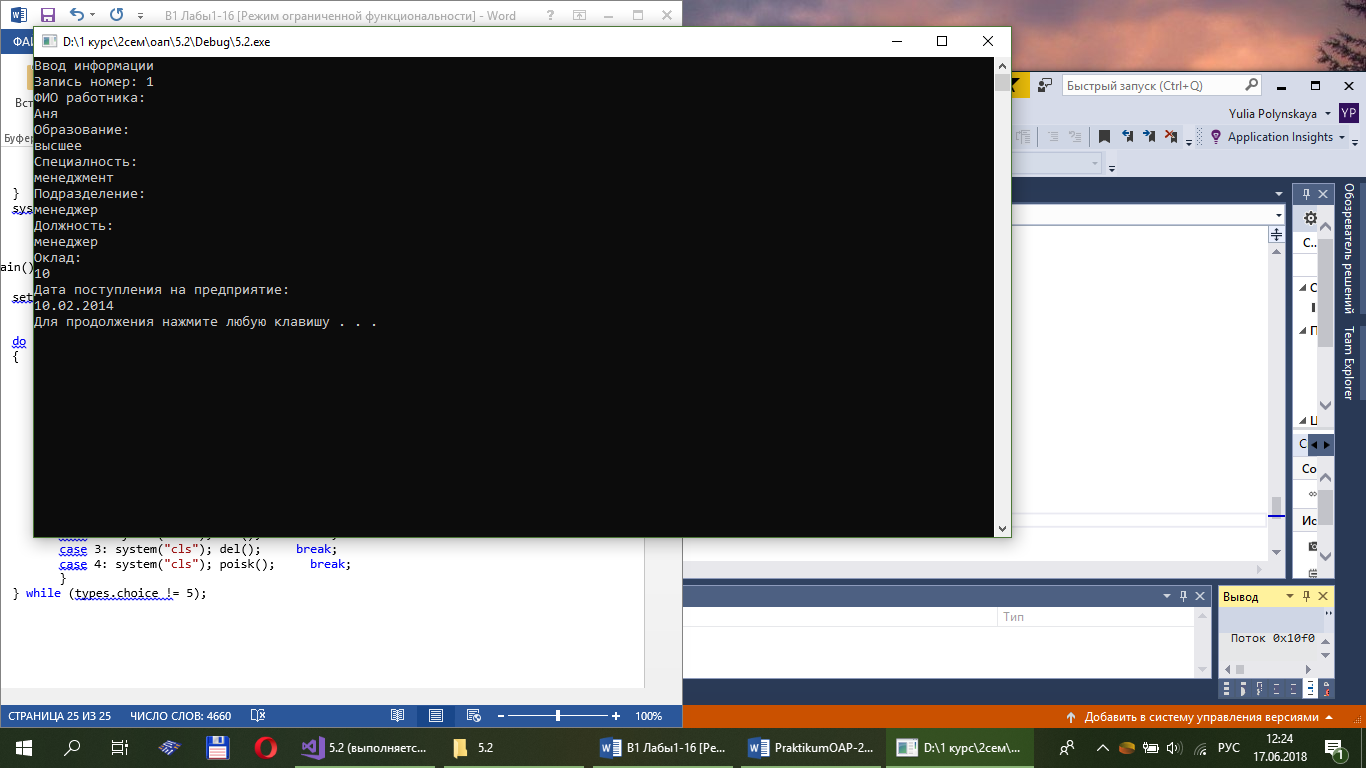
case 3: system("cls"); del(); break;

case 4: system("cls"); poisk(); break;

}

} while (types.choice != 5);

}



**Лабораторная работа № 6. Динамические структуры данных. Односвязные списки  
->указатель косвенной адресации**

**Список** представляет собой линейную последовательность переменных, каждая из которых связана указателями со своими соседями. Списки бывают *односвязные* (каждый элемент имеет указатель на следующий); *двусвязные* (каждый элемент имеет указатель на следующий и на предыдущий); *двусвязные циклические* (первый и последний элементы ссылаются друг на друга).  
Ключевое слово **nullptr** используется для обнуления указателей.  
4. В соответствии со своим вариантом разработать программу с использованием *односвязного списка* по данным, представленным в таблице ниже.

Программа должна содержать меню с пунктами: добавление элемента, удаление элемента, поиск элемента, вывод списка в консольное окно, запись списка в файл, считывание списка из файла.

Создать список, содержащий элементы целого типа. Найти сумму отрицательных двухзначных элементов или выдать сообщение, что таких элементов нет.

#include "stdafx.h"

#include <iostream>

#include <fstream>

#include <math.h>

using namespace std;

struct list

{

int number;

list \*next;

};

void insert(list \*&p, int value);

void printList(list \*p);

void toFile(list \*&p);

void fromFile(list \*&p);

void menu(void);

int del(list \*&, int value);

void SumOfNegetive(list \*p);

void Find(list \*p, int value);

int main()

{

setlocale(LC\_CTYPE, "Russian");

list \*first = nullptr;

int choice; int value;

menu();

cin >> choice;

while (choice != 8)

{

switch (choice)

{

case 1: cout << "Введите число ";

cin >> value;

insert(first, value);

printList(first);

break;

case 2: toFile(first);

break;

case 3: fromFile(first);

break;

case 4: cout << "Введите число ";

cin >> value;

del(first, value);

printList(first);

break;

case 5: SumOfNegetive(first);

break;

case 6: printList(first);

break;

case 7: cout << "Введите число ";

cin >> value;

Find(first, value);

break;

default: cout << "Неправильный выбор" << endl;

menu(); break;

}

cin >> choice;

}

return 0;

}

void insert(list \*&p, int value)

{

list \*newP = new list;

if (newP != NULL)

{

newP->number = value;

newP->next = p;

p = newP;

}

else

cout << "Операция добавления не выполнена" << endl;

}

void printList(list \*p)

{

if (p == NULL)

cout << "Список пуст" << endl;

else

{

cout << "Список:" << endl;

while (p != NULL)

{

cout << " -> " << p->number;

p = p->next;

}

cout << " -> NULL" << endl;

}

}

void toFile(list \*&p)

{

list \*temp = p;

list buf;

ofstream frm("mList.dat");

if (frm.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

while (temp)

{

buf = \*temp;

frm.write((char \*)&buf, sizeof(list));

temp = temp->next;

}

frm.close();

cout << "Список записан в файл mList.dat\n";

}

void fromFile(list \*&p)

{

list buf, \*first = nullptr;

ifstream frm("mList.dat");

if (frm.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

frm.read((char \*)&buf, sizeof(list));

while (!frm.eof())

{

insert(first, buf.number);

cout << " -> " << buf.number;

frm.read((char \*)&buf, sizeof(list));

}

cout << " -> NULL" << endl;

frm.close();

p = first;

cout << "\nСписок считан из файла mList.dat\n";

}

int del(list \*&p, int value)

{

list \*previous, \*current, \*temp;

if (value == p->number)

{

temp = p;

p = p->next; // отсоединить узел

delete temp; //освободить отсоединенный узел

return value;

}

else

{

previous = p;

current = p->next;

while (current != NULL && current->number != value)

{

previous = current;

current = current->next; // перейти к следующему

}

if (current != NULL)

{

temp = current;

previous->next = current->next;

free(temp);

return value;

}

}

return 0;

}

void SumOfNegetive(list \*p) {

int sum = 0;

while (p != NULL)

{

if (p->number < 0)

if (abs(p->number) >= 10 && abs(p->number) <= 99)

sum += p->number;

p = p->next;

}

if (sum != 0)

cout << "Сумма отрицательных двузначных:" << sum << endl;

else

cout << "в списке нет таких чисел" << endl;

}

void Find(list \*p, int value) {

while (p != NULL)

{

if (p->number == value)

{

cout << "Число: " << value << " найдено в списке." << endl;

return;

}

else

p = p->next;

}

cout << "в списке нет таких чисел" << endl;

}

void menu(void)

{

cout << "Сделайте выбор:" << endl;

cout << " 1 - Ввод числа" << endl;

cout << " 2 - Запись списка в файл" << endl;

cout << " 3 - Чтение списка из файла" << endl;

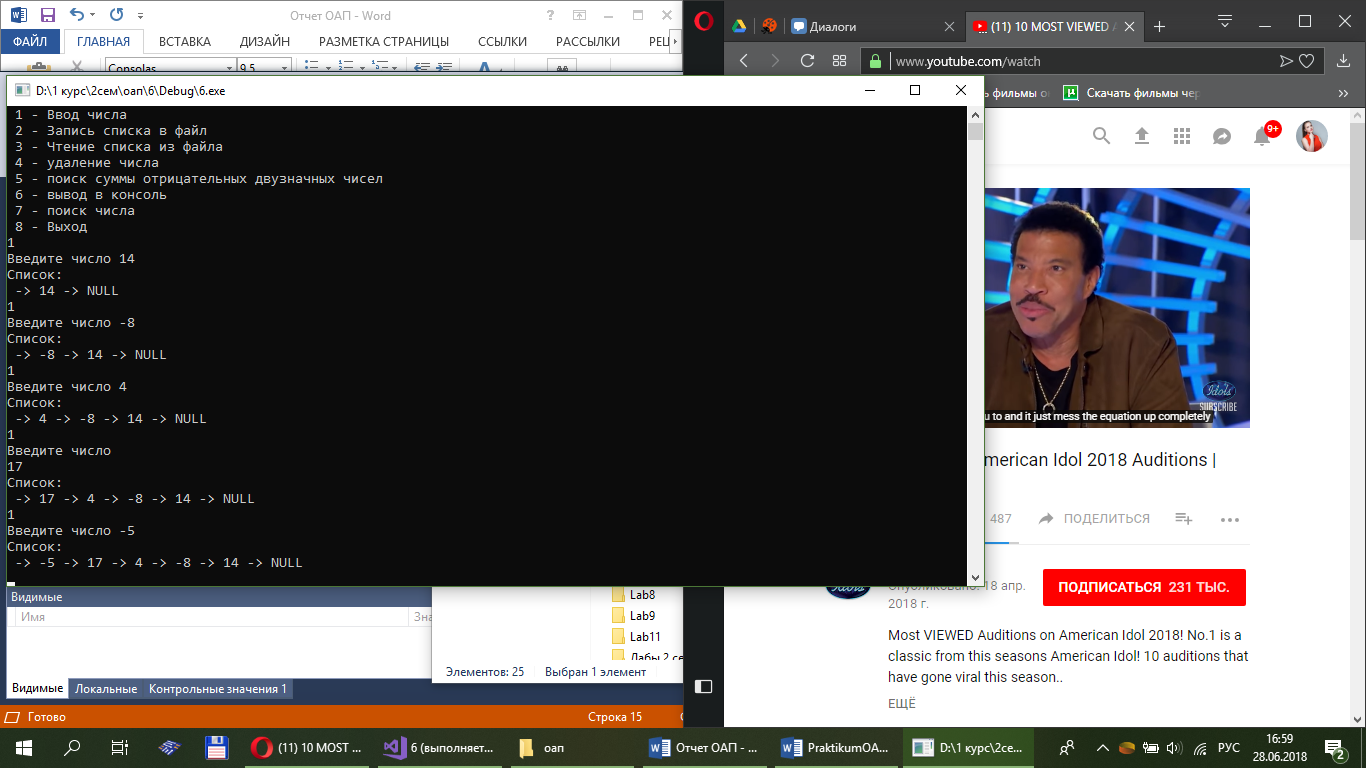
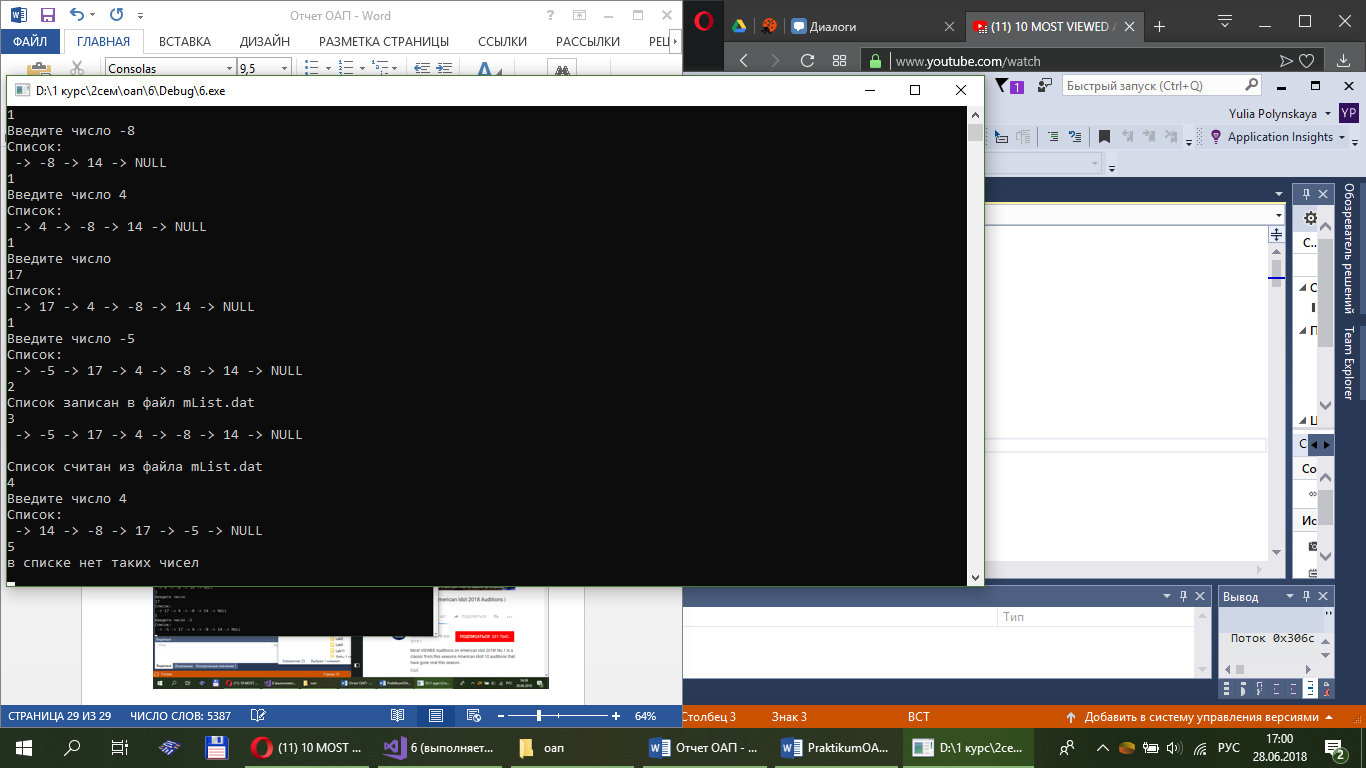
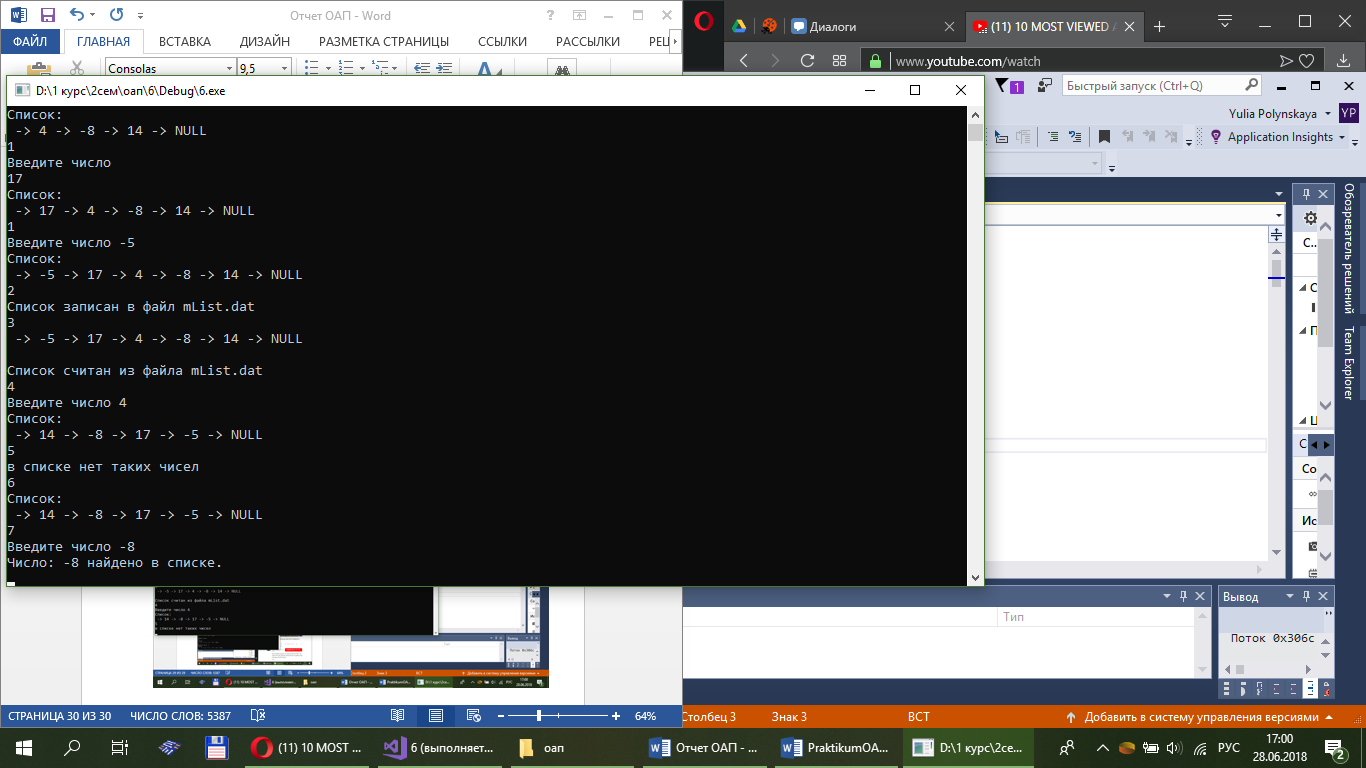
cout << " 4 - удаление числа" << endl;

cout << " 5 - поиск суммы отрицательных двузначных чисел" << endl;

cout << " 6 - вывод в консоль" << endl;

cout << " 7 - поиск числа" << endl;

cout << " 8 - Выход" << endl;

}  
  
  
  
**Лабораторная работа № 7. Полустатические структуры данных: стеки**

При удалении мы копируем наш список и последний элемент становится первым

**Стеком** называется одномерная структура данных, включение и исключение элементов которого осуществляется с помощью указателя стека в соответствии с правилом "последним введен, первым выведен" (last-in, first-out− **LIFO**). Стек может быть реализован статически (на основе *массива*) и динамически (на основе *списка*).

*Первая* часть − программный модуль с главной функцией (пусть он имеет имя **proectStack**) − организует вывод меню и обращения к функциям работы со стеком, описанным в файле **myStack.cpp**.

Для добавления *второго* программного модуля с именем **myStack.cpp**, содержащего функции для манипуляции со стеком (добавление, удаление и вывод элементов),  
*Третья* часть – это заголовочный файл **myStack.h**, который содержит описание структуры стека и прототипы программ обработки его элементов.

4. Создать проект, демонстрирующий работу со стеком, организованным на основе *списка*,в соответствии со своим вариантом для данных из таблицы ниже. Все операции со стеком реализовать через функции. Дополнить проект функциями очистки стека **clear()**, сохранения в файл и считывания из файла.

Проект должен содержать три части: главная функция, файл с функциями работы со стеком и заголовочный файл. Создать интерфейс в виде меню.

Разработать функцию, которая подсчитывает количество элементов стека, у которых равные "соседи".  
list.h  
#pragma once

#include <iostream>

using namespace std;

struct ElementStak

{

int d;

struct ElementStak \*next;

};

struct Stack {

void push(int d);

int pop();

ElementStak\*tail = NULL;

};

list.cpp  
#include "stdafx.h"

#include <iostream>

#include "List.h"

void Stack::push(int d)

{

ElementStak \*pv = new ElementStak;

pv->d = d; // значение помещается в стек

pv->next = tail;

tail = pv;

};

int Stack::pop()

{

int tmp = tail->d; // извлекается в tmp значение в вершине стека

ElementStak \*pv = tail;// запоминается указатель на вершину стека

tail = tail->next; // вершиной становится предшеств. элемент

delete pv; // освобождается память,

return tmp;

};  
7.cpp  
#include "stdafx.h"

#include <ctime>

#include <iostream>

#include "List.h"

#include <fstream>

using namespace std;

#define SIZE 20

void menu();

void toFile(Stack \*&p);

void fromFile(Stack \*&p);

void clear(Stack \*&p);

void findNeighbors(Stack \*&p);

int main()

{

setlocale(LC\_ALL, "rus");

srand((unsigned)time(NULL));

Stack \*MyStk = new Stack;

menu();

int choice; int value;

cin >> choice;

while (choice != 6)

{

switch (choice)

{

case 1: cout << "Введите число ";

cin >> value;

MyStk->push(value);

break;

case 2: toFile(MyStk);

break;

case 3: fromFile(MyStk);

break;

case 4: clear(MyStk);

break;

case 5: findNeighbors(MyStk);

break;

default: cout << "Неправильный выбор" << endl;

menu(); break;

}

cout << " ";

cin >> choice;

}

system("pause");

return 0;

}

void menu()

{

cout << "Сделайте выбор:" << endl;

cout << " 1 - Ввод числа" << endl;

cout << " 2 - Запись стека в файл" << endl;

cout << " 3 - Чтение стека из файла" << endl;

cout << " 4 - Очистка стека" << endl;

cout << " 5 - Количество элементов стека, у которых равные \"соседи\"." << endl;

cout << " 6 - Выход" << endl;

}

void toFile(Stack \*&p)

{

ElementStak \*temp = p->tail;

ofstream frm("mList.dat");

if (frm.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

while (temp)

{

frm.write((char \*)&temp->d, sizeof(Stack));

temp = temp->next;

}

frm.close();

cout << "Список записан в файл mList.dat\n";

}

void fromFile(Stack \*&p)

{

ElementStak buf;

Stack \*first = new Stack;

ifstream frm("mList.dat");

if (frm.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

frm.read((char \*)&buf, sizeof(Stack));

while (!frm.eof())

{

first->push(buf.d);

frm.read((char \*)&buf, sizeof(Stack));

}

frm.close();

p = first;

cout << "\nСписок считан из файла mList.dat\n";

}

void clear(Stack\*& p)

{

ElementStak \*temp = p->tail;

while (temp->next != NULL)

{

ElementStak \*buf = temp;

temp = temp->next;

delete buf;

}

delete temp;

p->tail = NULL;

cout << "Стек очищен" << endl;

}

void findNeighbors(Stack \*&p) {

Stack \*stack = p;

if (stack->tail)

{

Stack \*MyStk2 = new Stack;

int kol = 0, a = stack->pop(), b = stack->pop(), c = stack->pop();

while (stack->tail != NULL)//пока не извлеч. все эл. из стека

{

if (a == c)//если соседи совпад.

kol++;

MyStk2->push(a);//запис. левого соседа

a = b;//новый левый- текущий

b = c;//текущий-правый сосед

c = stack->pop();//новым правым соседом становится еще один новый извлеч. элемент

}

if (a == c)

kol++;

MyStk2->push(a);

MyStk2->push(b);

MyStk2->push(c);

while (MyStk2->tail != NULL)

{

stack->push(MyStk2->pop());//извлек. из 2, помещ. в 1

}

cout << "Cтек:\t";

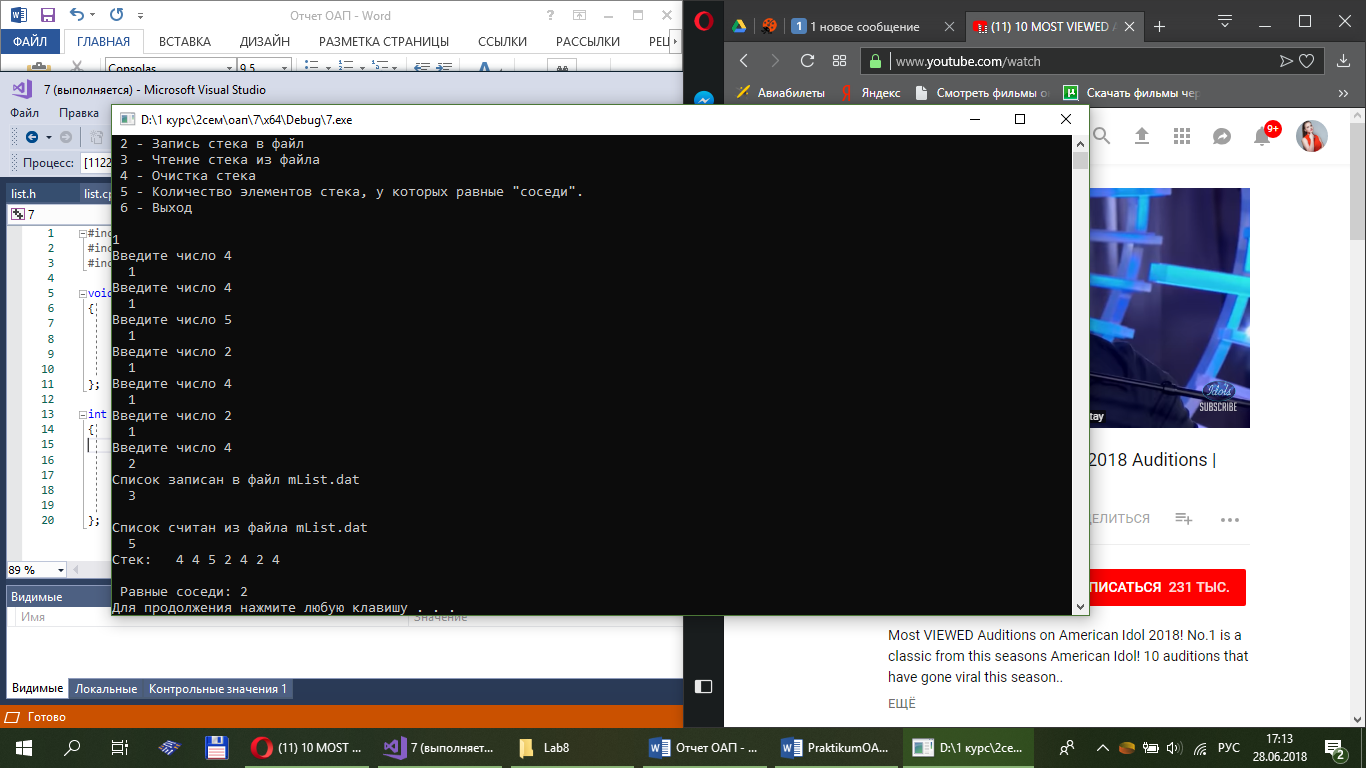
while (stack->tail != NULL)

cout << stack->pop() << " "; cout << endl;

cout << "\n Равные соседи: " << kol << endl;

system("pause");

}

}  
  
**Лабораторная работа № 8. Полустатические структуры данных: очереди**

**Очередь** − одномерная структура данных, для которых загрузка или извлечение элементов осуществляется с помощью указателей начала извлечения (**head**) и конца (**tail**) очереди в соответствии с правилом **FIFO** ("**first-in, first-out**" − "первым введен, первым выведен"), т. е. включение производится с одного, а исключение – с другого конца.

4. Создать проект из нескольких файлов, демонстрирующий работу с очередью. В соответствии со своим вариантом выполнить задание из таблицы, представленной ниже.

Разработать меню и реализовать все операции с очередью через функции. Максимальный размер очереди ввести с клавиатуры.

Создать очередь для целых чисел и функции для ввода, вывода, удаления и определения размера очереди. Вывести элементы очереди до первого нулевого (включительно). Найти максимальный элемент очереди.  
Lab8.h  
struct Number

{

int info;

Number \*next;

};

void create(Number \*\*begin, Number \*\*end, int p); //формирование элементов очереди

void view(Number \*begin); //функция вывода элементов очереди

Number \*view\_befor\_first\_zero\_elem(Number \*begin); //определение первого нулевого элемента

void show\_count\_after\_zero\_elem(Number \*begin); //вывод количества элементов после нулевого элемента

void clean(Number \*\*begin);

int get\_count(Number\* begin);//размер очереди

Number\* get\_max\_number(Number \*begin);  
Lab8.cpp  
#include "stdafx.h"

#include "Lab8.h"

#include<iostream>

using namespace std;

int menu();

int main()

{

Number \*begin = nullptr, \*end, \*t;

Number\* zeroElem = nullptr;

setlocale(LC\_CTYPE, "Rus");

while (true)

{

switch (menu())

{

case 1:

{

t = new Number;

int p, size;

cout << "\nВведите размер очереди =";

cin >> size;

cout << "Введите номер= ";

cin >> p;

cin.ignore(cin.rdbuf()->in\_avail());

cin.sync();

t->info = p; //первый элемент

t->next = nullptr;

begin = end = t;

for (int i = 1; i < size; i++) //создание очереди

{

cout << "Введите номер= ";

cin >> p;

cin.ignore(cin.rdbuf()->in\_avail());

cin.sync();

create(&begin, &end, p);

}

break;

}

case 2:

{

if (begin == nullptr) //вывод на экран

cout << "Нет элементов" << endl;

else

view(begin);

}

break;

case 3:

{

zeroElem = view\_befor\_first\_zero\_elem(begin); //определение первого нулевого элемента

break;

}

case 4:

{

if (zeroElem == nullptr)

{

cout << "Сначала выполните пункт 3" << endl;

break;

}

show\_count\_after\_zero\_elem(zeroElem); //вывод количества элементов после нулевого элемента

break;

}

case 5:

{

Number\* maxElem = get\_max\_number(begin);

if (maxElem == nullptr)

{

cout << "Наибольший элемент: не найден" << endl;

}

cout << "Наибольший элемент: " << maxElem->info << endl;

break;

}

case 6:

{

cout << "Размер очереди: " << get\_count(begin) << endl;

break;

}

case 7:

{

clean(&begin);

break;

}

case 8:

{

system("pause");

exit(0);

break;

}

default:

break;

}

}

}

int menu()

{

char s[80];

int c;

cout << endl;

cout << "1. Создание очереди" << endl;

cout << "2. Вывод очереди" << endl;

cout << "3. Вывод элементов до нулевого" << endl;

cout << "4. Вывод количества элементов после нулевого" << endl;

cout << "5. Наибольший элемент" << endl;

cout << "6. Размер очереди" << endl;

cout << "7. Очистка очереди" << endl;

cout << "8. Выход" << endl;

cout << endl;

do

{

cout << "Ваш выбор: ";

cin.sync();

gets\_s(s);

cout << endl;

c = atoi(s);

}

while (c < 0 || c > 8);

return c;

}  
  
Funcs.cpp  
#include "stdafx.h"

#include "Lab8.h"

#include <iostream>

using namespace std;

void create(Number\*\* begin, Number\*\* end, int p) //Формирование элементов очереди

{

Number\* t = new Number;

t->next = nullptr;

if (\*begin == nullptr)

\*begin = \*end = t;

else

{

t->info = p;

(\*end)->next = t;

\*end = t;

}

}

void view(Number\* begin) //Вывод элементов очереди

{

Number\* t = begin;

if (t == nullptr)

{

cout << "Номер пуст\n";

return;

}

while (t != nullptr)

{

cout << t->info << endl;

t = t->next;

}

}

Number\* view\_befor\_first\_zero\_elem(Number\* begin) //Опрделение элементов до 0

{

Number\* t = begin;

int min;

if (t == nullptr)

{

cout << "Номер пуст\n";

return nullptr;

}

min = t->info;

while (t != nullptr)

{

if (min != 0)

{

cout << min << "\t";

t = t->next;

min = t->info;

}

if (min == 0)

{

cout << min << "\t" << endl;

return t;

}

}

}

void show\_count\_after\_zero\_elem(Number\* begin) // после минимального элемента

{

Number\* t = begin;

int count = 0;

while (t != nullptr)

{

count++;

t = t->next;

}

cout << "Количество элементов после 0: " << count - 1 << endl;

}

int get\_count(Number\* begin)//размер очереди

{

Number\* t = begin;

int count = 0;

while (t != nullptr)

{

count++;

t = t->next;

}

return count;

}

Number\* get\_max\_number(Number\* begin)

{

Number \*t = begin, \*mn = nullptr;

if (t == nullptr)

{

cout << "Номер пуст\n";

return nullptr;

}

int max = t->info;

while (t != nullptr)

{

if (t->info >= max)

{

max = t->info;

mn = t;

}

t = t->next;

}

return mn;

}

void clean(Number\*\* begin)

{

Number\* t;

t = new Number;

while (\*begin != nullptr)

{

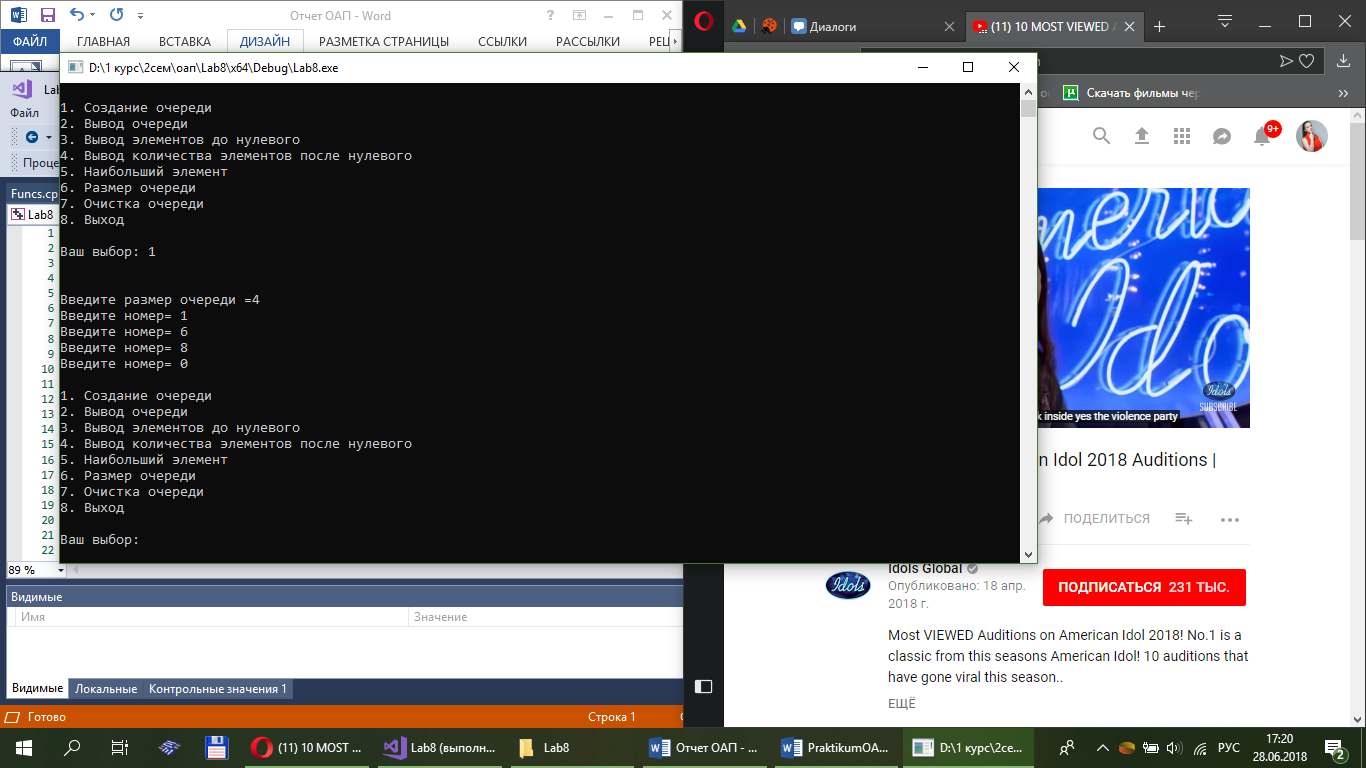
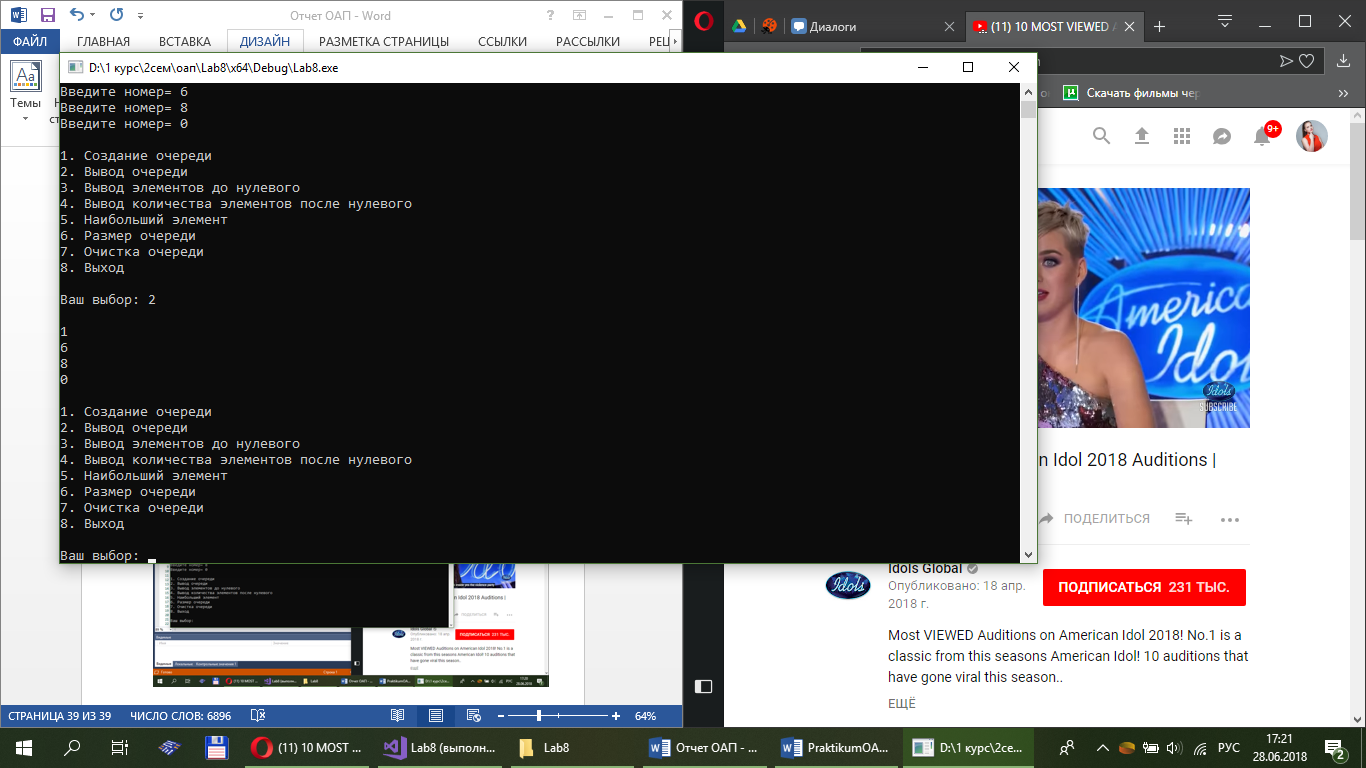
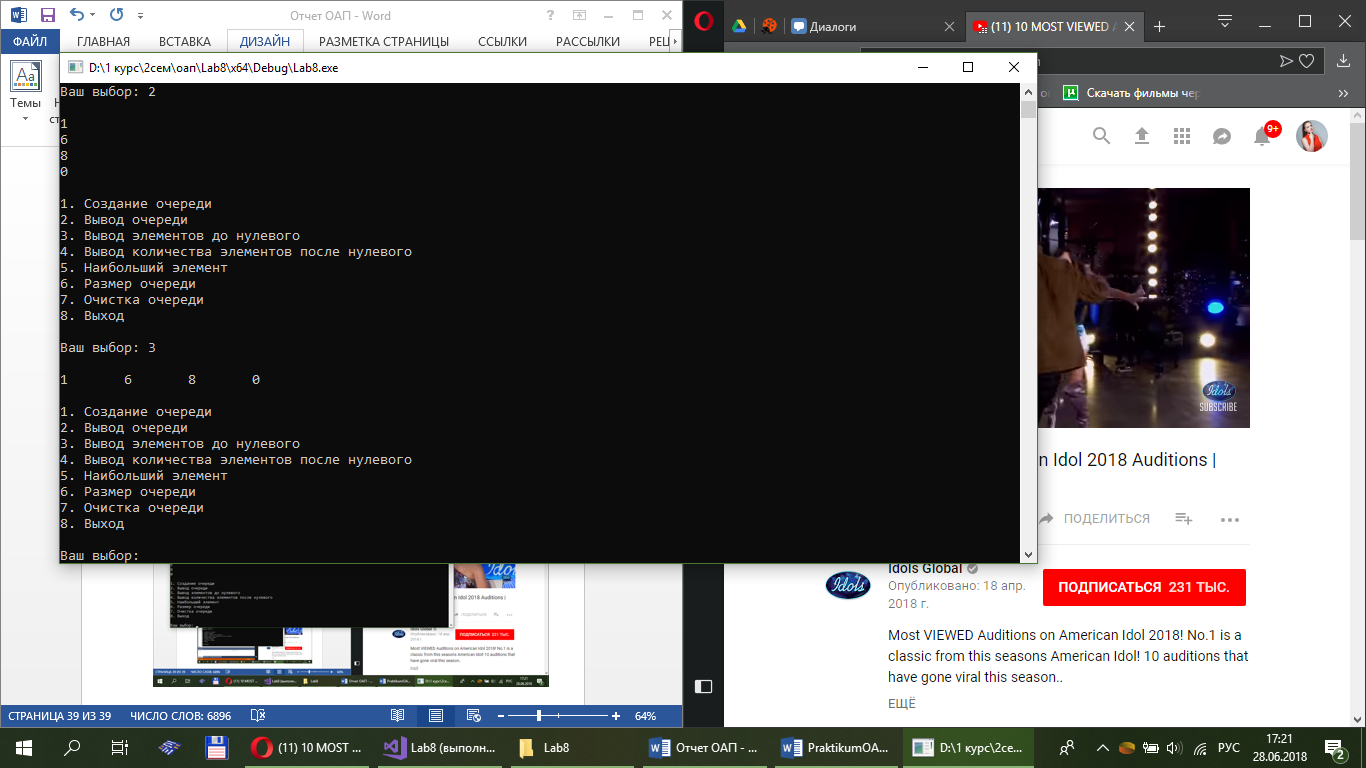
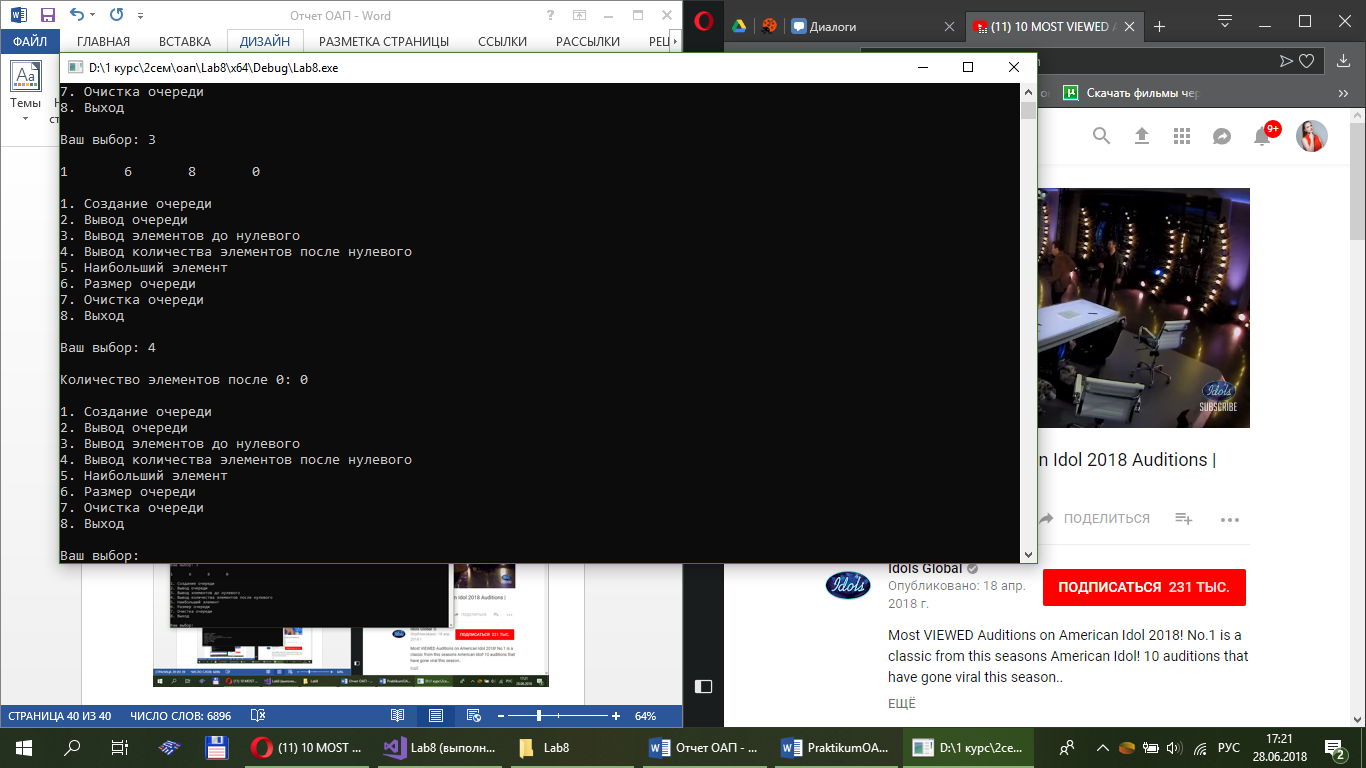
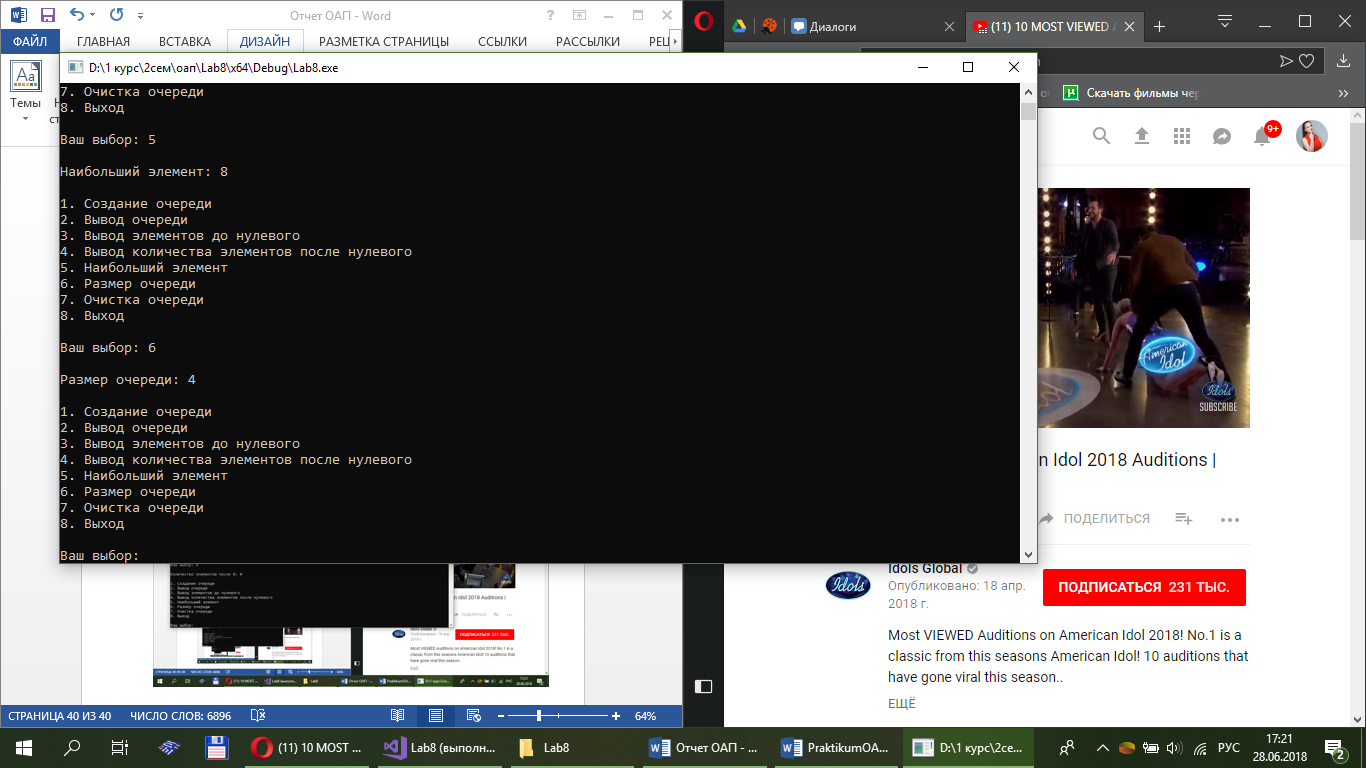
t = \*begin;

\*begin = (\*begin)->next;

delete t;

}

}

     
  
**Лабораторная работа № 9. Двусвязные списки**

В *двусвязных списках* каждый элемент имеет указатели на следующий и на предыдущий элементы.

2. Дополнить программу функцией в соответствии со своим вариантом из таблицы, представленной ниже.   
**findMin()** – функция поиска минимального  элемента списка по одному из выбранных полей.  
#include "stdafx.h"

#include <iostream>

#include <fstream>

using namespace std;

const unsigned int NAME\_SIZE = 30;

const unsigned int CITY\_SIZE = 20;

struct Address

{

char name[NAME\_SIZE];

char city[CITY\_SIZE];

Address\* next;

Address\* prev;

};

int menu();

void insert(Address\* e, Address\*\* phead, Address\*\* plast);

Address\* setElement();

void outputList(Address\*\* phead, Address\*\* plast);

void findByName(char name[NAME\_SIZE], Address\*\* phead);

void deleteByName(char name[NAME\_SIZE], Address\*\* phead, Address\*\* plast);

void writeToFile(Address\*\* phead);

void readFromFile(Address\*\* phead, Address\*\* plast);

void findMin(bool isName, Address\* phead);

int main()

{

Address\* head = nullptr;

Address\* last = nullptr;

setlocale(LC\_CTYPE, "Rus");

while (true)

{

switch (menu())

{

case 1: insert(setElement(), &head, &last);

break;

case 2:

{

char dname[NAME\_SIZE];

cout << "Введите имя: ";

cin.getline(dname, NAME\_SIZE - 1, '\n');

cin.ignore(cin.rdbuf()->in\_avail());

cin.sync();

deleteByName(dname, &head, &last);

}

break;

case 3: outputList(&head, &last);

break;

case 4:

{

char fname[NAME\_SIZE];

cout << "Введите имя: ";

cin.getline(fname, NAME\_SIZE - 1, '\n');

cin.ignore(cin.rdbuf()->in\_avail());

cin.sync();

findByName(fname, &head);

}

break;

case 5:

{

writeToFile(&head);

break;

}

case 6:

{

readFromFile(&head, &last);

break;

}

case 7:

{

cout << "По какому полю ищем? " << endl << "1. По имени" << endl << "2. По Городу" << endl;

char s[80];

int c;

cin.sync();

gets\_s(s);

cout << endl;

c = atoi(s);

switch (c)

{

case 1:

{

findMin(true, head);

break;

}

case 2:

{

findMin(false, head);

break;

}

default:

{

cout<<"Не верный выбор"<<endl;

break;

}

}

break;

}

case 8: exit(0);

default: exit(1);

}

}

}

int menu()

{

char s[80];

int c;

cout << endl;

cout << "1. Ввод имени" << endl;

cout << "2. Удаление записи по имени" << endl;

cout << "3. Вывод на экран" << endl;

cout << "4. Поиск записи по имени" << endl;

cout << "5. Запись в файл" << endl;

cout << "6. Чтение из файла" << endl;

cout << "7. Поиск минимальной длинны поля" << endl;

cout << "8. Выход" << endl;

cout << endl;

do

{

cout << "Ваш выбор: ";

cin.sync();

gets\_s(s);

cout << endl;

c = atoi(s);

}

while (c < 0 || c > 8);

return c;

}

void insert(Address\* e, Address\*\* phead, Address\*\* plast) //Добавление в конец списка

{

Address\* p = \*plast;

if (\*plast == nullptr)

{

e->next = nullptr;

e->prev = nullptr;

\*plast = e;

\*phead = e;

return;

}

p->next = e;

e->next = nullptr;

e->prev = p;

\*plast = e;

}

Address\* setElement() // Создание элемента и ввод его значений с клавиатуры

{

Address\* temp = new Address();

if (!temp)

{

cerr << "Ошибка выделения памяти памяти";

return nullptr;

}

cout << "Введите имя: ";

cin.getline(temp->name, NAME\_SIZE - 1, '\n');

cin.ignore(cin.rdbuf()->in\_avail());

cin.clear();

cout << "Введите город: ";

cin.getline(temp->city, CITY\_SIZE - 1, '\n');

cin.ignore(cin.rdbuf()->in\_avail());

cin.clear();

temp->next = nullptr;

temp->prev = nullptr;

return temp;

}

void outputList(Address\*\* phead, Address\*\* plast)

{

Address\* t = \*phead;

while (t)

{

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << t->name << ' ' << t->city << endl;

t = t->next;

}

cout << "" << endl;

setlocale(LC\_CTYPE, "Russian");

}

void findByName(char name[30], Address\*\* phead)

{

Address\* t = \*phead;

while (t)

{

if (!strcmp(name, t->name)) break;

t = t->next;

}

if (!t)

cerr << "Имя не найдено" << endl;

else

{

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout << t->name << ' ' << t->city << endl;

setlocale(LC\_CTYPE, "Russian");

}

}

void deleteByName(char name[NAME\_SIZE], Address\*\* phead, Address\*\* plast)

{

struct Address\* t = \*phead;

while (t)

{

if (!strcmp(name, t->name)) break;

t = t->next;

}

if (!t)

cerr << "Имя не найдено" << endl;

else

{

if (\*phead == t)

{

\*phead = t->next;

if (\*phead)

(\*phead)->prev = nullptr;

else

\*plast = nullptr;

}

else

{

t->prev->next = t->next;

if (t != \*plast)

t->next->prev = t->prev;

else

\*plast = t->prev;

}

delete t;

cout << "Элемент удален" << endl;

}

}

void writeToFile(Address\*\* phead)

{

struct Address\* t = \*phead;

FILE\* fp;

errno\_t err = fopen\_s(&fp, "mlist", "wb");

if (err)

{

cerr << "Файл не открывается" << endl;

exit(1);

}

cout << "Сохранение в файл" << endl;

while (t)

{

fwrite(t, sizeof(struct Address), 1, fp);

t = t->next;

}

fclose(fp);

}

void readFromFile(Address\*\* phead, Address\*\* plast)

{

struct Address\* t;

FILE\* fp;

errno\_t err = fopen\_s(&fp, "mlist", "rb");

if (err)

{

cerr << "Файл не открывается" << endl;

exit(1);

}

while (\*phead)

{

\*plast = (\*phead)->next;

delete \*phead;

\*phead = \*plast;

}

\*phead = \*plast = nullptr;

cout << "Загрузка из файла" << endl;

while (!feof(fp))

{

t = new Address();

if (!t)

{

cerr << "Ошибка выделения памяти" << endl;

return;

}

if (1 != fread(t, sizeof(struct Address), 1, fp)) break;

insert(t, phead, plast);

}

fclose(fp);

}

void findMin(bool isNmae, Address\* phead)

{

Address\* current = phead;

Address\* minNode = nullptr;

if (current == nullptr)

{

cout << "Список пуст" << endl;

return;

}

minNode = current;

int minLenght = 0;

if (isNmae)

{

minLenght = strlen(current->name);

}

else

{

minLenght = strlen(current->city);

}

while (current != nullptr)

{

int currentFieldLength = 0;

if (isNmae)

{

currentFieldLength = strlen(current->name);

}

else

{

currentFieldLength = strlen(current->city);

}

if (currentFieldLength < minLenght)

{

minLenght = currentFieldLength;

minNode = current;

}

current = current->next;

}

if (minNode == nullptr)

{

cout << "Элемент не найден" << endl;

return;

}

cout << "Элемент с минимальной длинной ";

isNmae ? cout << "имени" << endl: cout << "названия города: " << endl;

cout << "Имя: " ;

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout<< minNode->name << endl;

setlocale(LC\_CTYPE, "Russian");

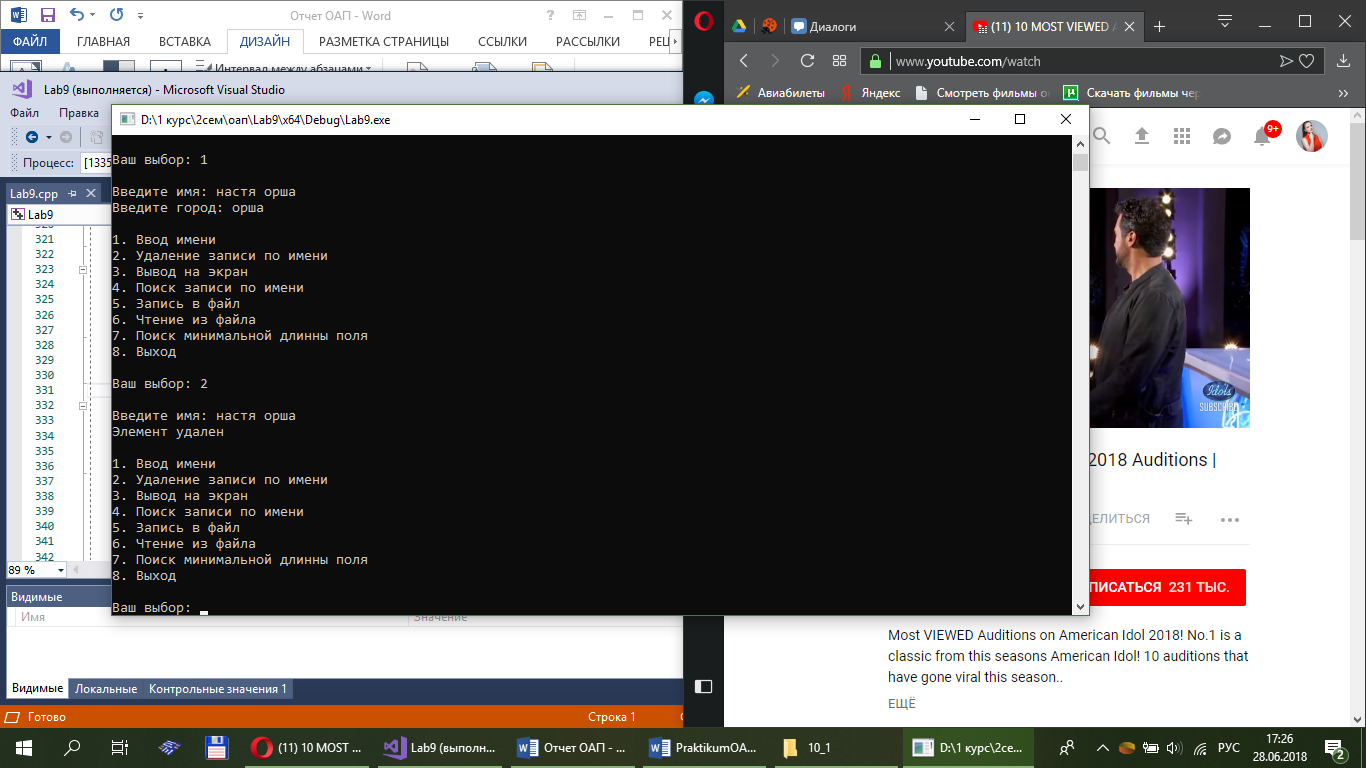
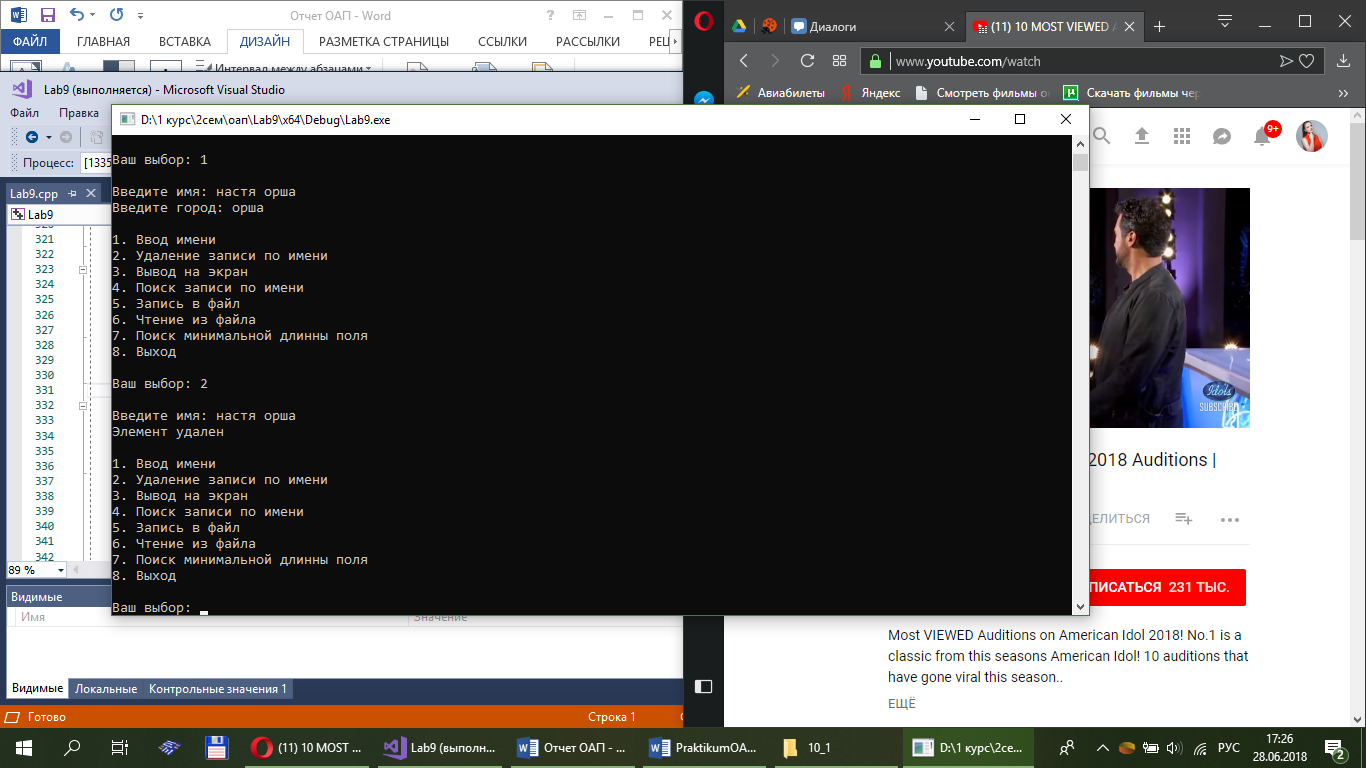
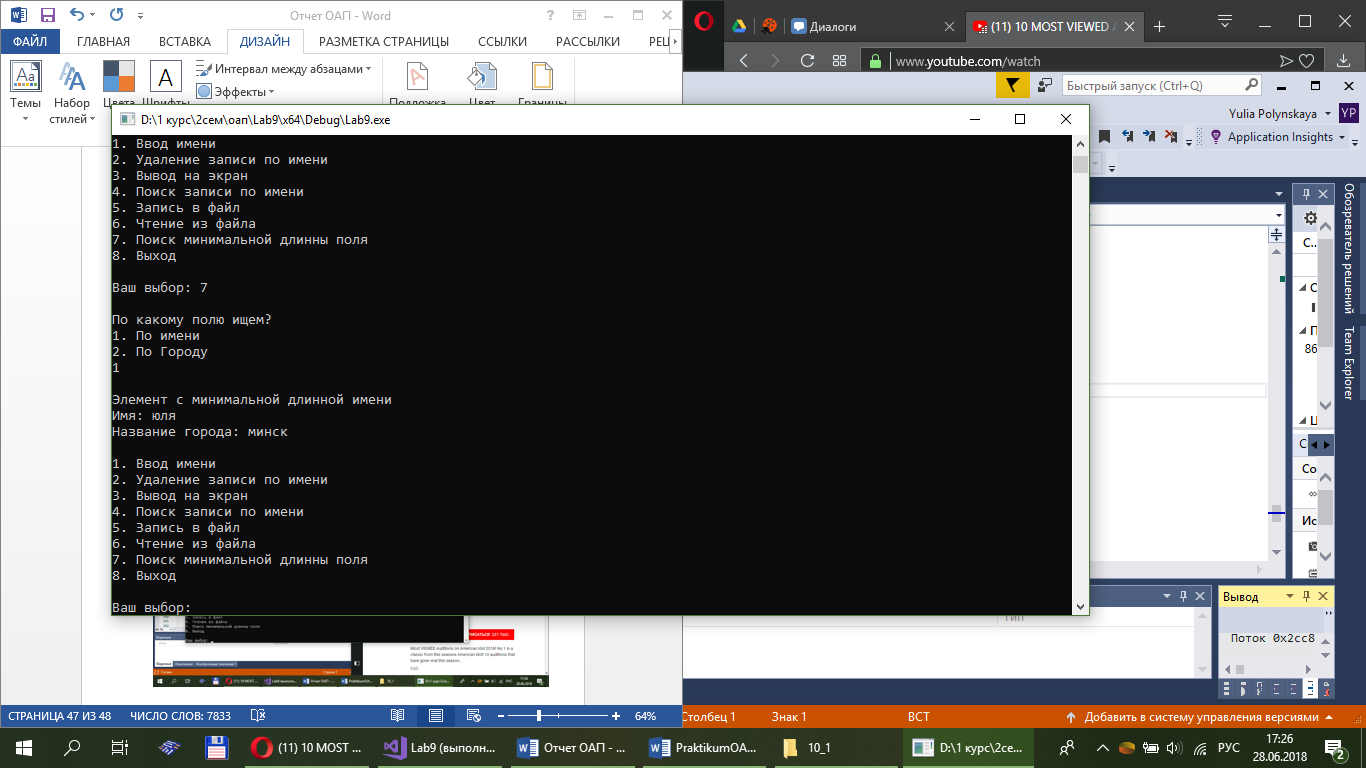
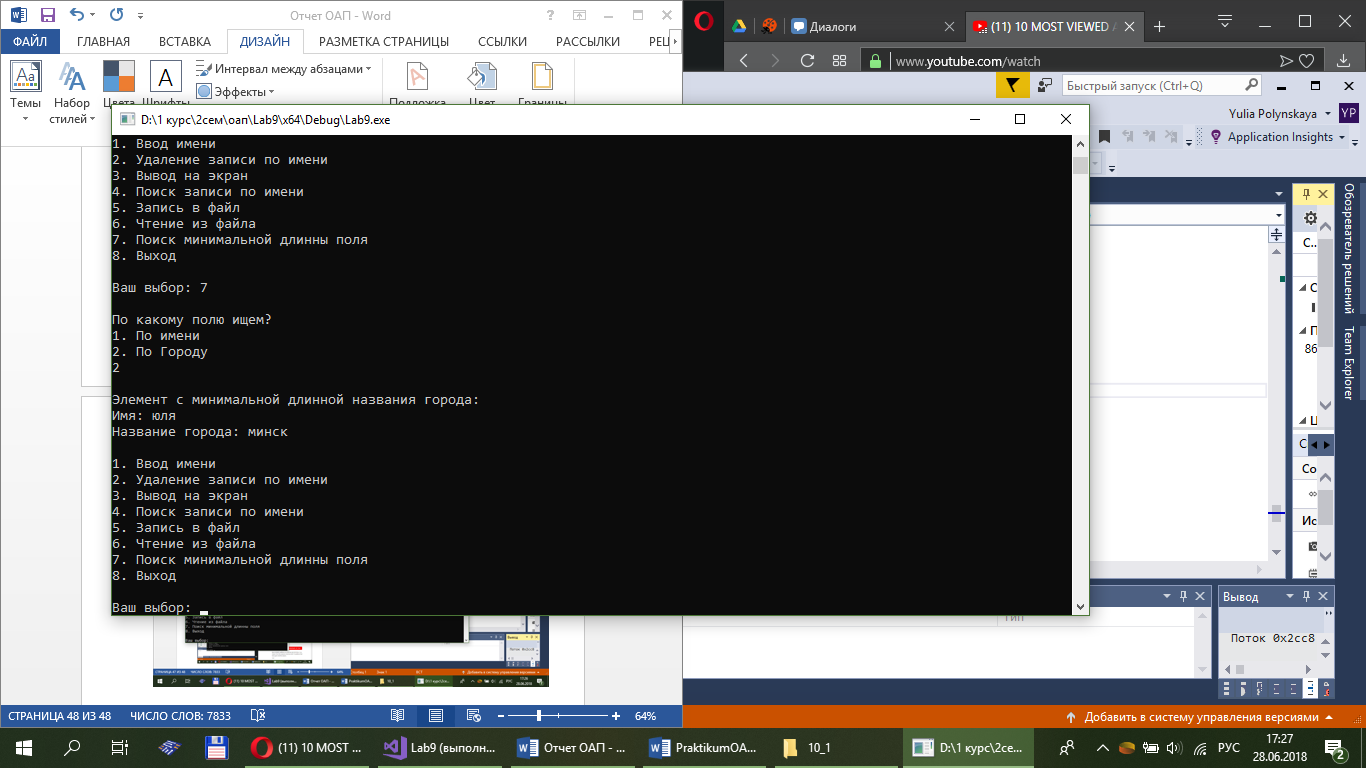
cout << "Название города: ";

setlocale(LC\_CTYPE, "Russian\_Russia.866");

cout<< minNode->city << endl;

setlocale(LC\_CTYPE, "Russian");

}

     
**Лабораторная работа № 10. Рекурсивные алгоритмы**

*Рекурсивная функция* − это функция c такой организацией работы, при которой она вызывает сама себя. Рекурсия должна иметь внутри себя условие завершения. Рекурсивная функция (программа) может быть *линейной* (функция содержит единственный вызов самой себя), *смешанной* (две или более функций вызывают друг друга попеременно), *ветвящейся* (когда рекурсивный вызов содержится в теле функции более одного раза, либо производится в цикле), *вложенной* (имеется вызов функции внутри обращения к самой функции).

4. В соответствии со своим вариантом выполнить задания из таблицы, представленной ниже. В некоторых заданиях имеются ошибки: не выполняется условие завершения рекурсии. Изменить условие такой задачи с тем, чтобы рекурсия выполнялась.

Разработать программу, реализующую рекурсивный алгоритм вычисления **A(m, n)** для любых целых не отрицательных **m** и **n**.  
#include "stdafx.h"

#include <iostream>

using namespace std;

int c(int m, int n);

int main()

{

setlocale(LC\_ALL, "rus");

int a, n = 5, m = 3;

a = c(m, n);

cout << "Результат:" << a << endl;

system("pause");

}

int c(int m, int n)

{

if (m < 0 || n < 0) return -1;

if (m == 0)

return n + 1;

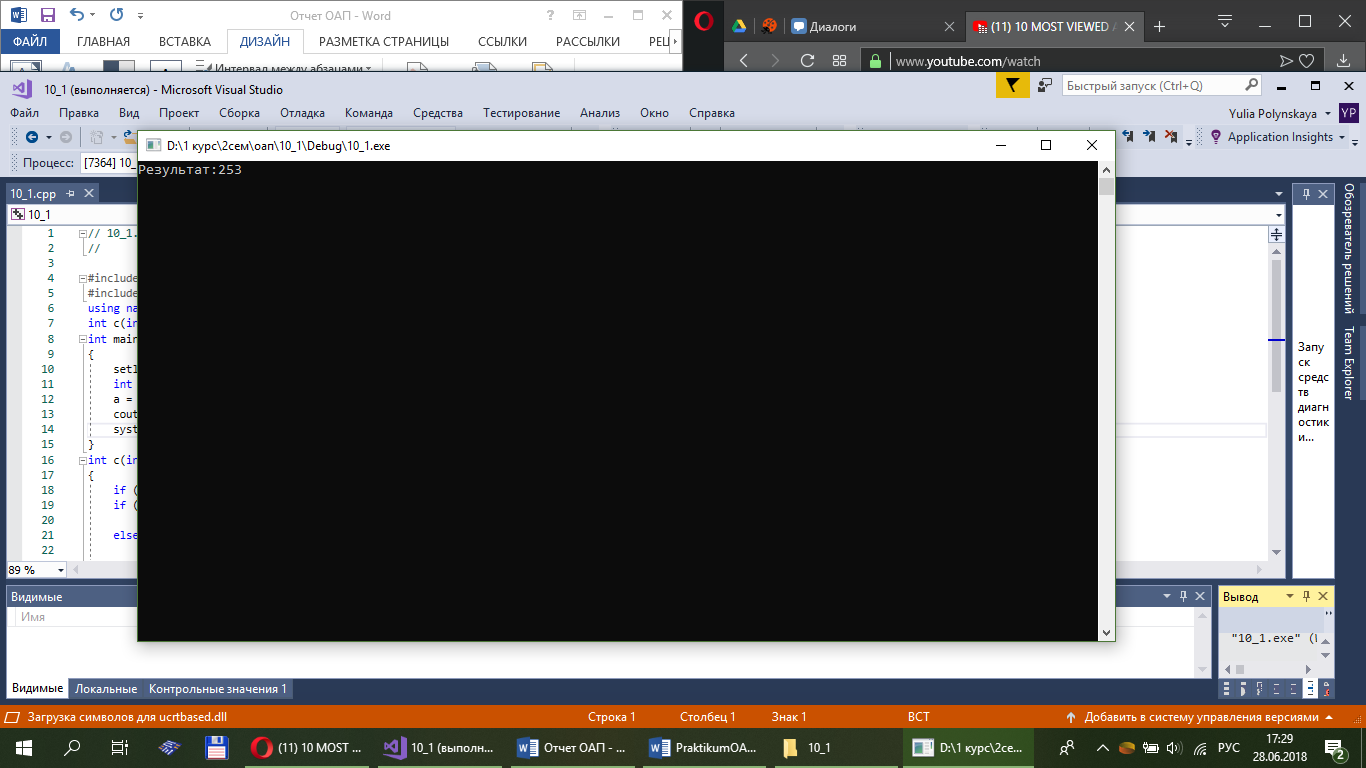
else

if (m > 0 && n == 0)

return c(m - 1, 1);

else

return c(m - 1, c(m, n - 1));

}  
  
**Лабораторная работа № 11. Бинарные деревья**

**Дерево** − это структура, имеющая следующие свойства:

– существует единственный элемент (узел, вершина), на который не ссылается никакой другой и который называется *корнем*;

– начиная с корня и следуя по определенной цепочке указателей, можно осуществить доступ к любому элементу структуры;

– на каждый элемент, кроме корня, имеется единственная ссылка.

**Бинарное дерево поиска** – это упорядоченное дерево, каждая вершина которого имеет **не более двух** поддеревьев: в левом поддереве содержатся ключи, имеющие значения, **меньшие**, чем значение данного узла, в правом поддереве содержатся ключи, имеющие значения, **большие**, чем значение данного узла.  
3. Разработать программу работы с бинарным деревом, в которую включить основные функции манипуляции данными и функцию в соответствии со своим вариантом из таблицы, представленной ниже.  
В текстовом файле построчно записаны целые числа. Написать функцию, которая по файлу F, все элементы которого различны, строит  соот­ветствующее бинарное дерево поиска T, и вторую функцию, которая записывает в файл G элементы дерева поиска Т в порядке их возрастания.  
#include "stdafx.h"

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

//В текстовом файле построчно записаны целые числа.Написать функцию,

//которая по файлу F, все элементы которого различны, строит

//соот¬ветствующее бинарное дерево поиска T, и вторую функ - цию,

//которая записывает в файл G элементы дерева поиска Т

//в порядке их возрастания.

ifstream fin("text.txt");

ofstream fout("myFile.txt");

struct Node

{

int data; //Информационное поле

Node \*left, \*right; //Указатели на левую и правую ветви дерева

};

Node\* tree = nullptr;

void insert(int a, Node\*\* t) //Добавление элемента a

{

if ((\*t) == nullptr) //если дерева нет, то создается элемент

{

(\*t) = new Node;

(\*t)->data = a;

(\*t)->left = (\*t)->right = nullptr;

return;

}

if (a > (\*t)->data) //дерево есть, если а больше текущего

insert(a, &(\*t)->right); //то элемент помещается вправо

else

insert(a, &(\*t)->left); //иначе - влево

}

void toFile(Node\* t, int u) //Вывод на экран

{

if (t == nullptr) return;

toFile(t->left, ++u); //левое поддерево

fout << t->data << endl;

u--;

toFile(t->right, ++u); // правое поддерево

}

void toConsloe(Node\* t, int level) //Вывод дерева

{

if (t)

{

toConsloe(t->right, level + 1); //вывод правого поддерева

for (int i = 0; i < level; i++)

cout << " ";

cout << t->data<<endl;

toConsloe(t->left, level + 1); //вывод левого поддерева

}

}

void main()

{

setlocale(LC\_CTYPE, "Russian");

int temp;

while (!fin.eof())

{

fin >> temp;

insert(temp, &tree);

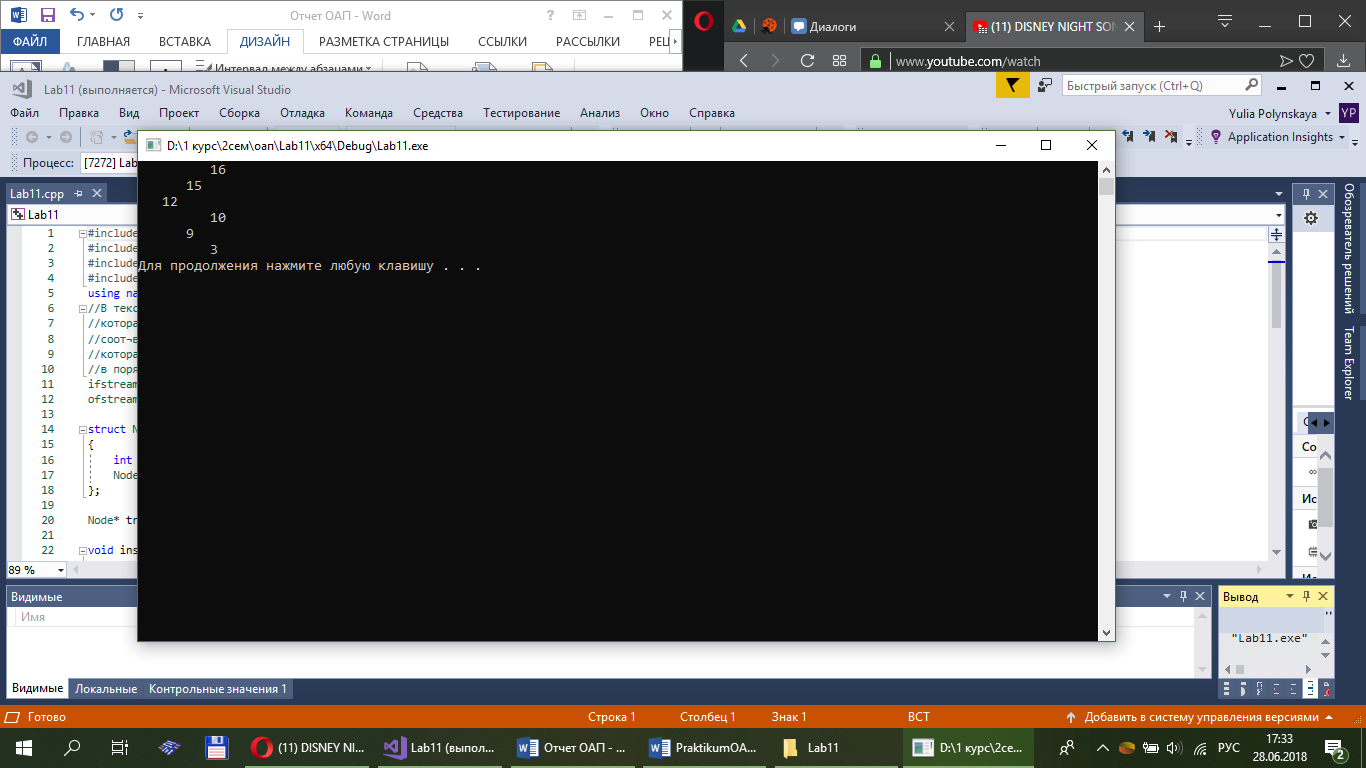
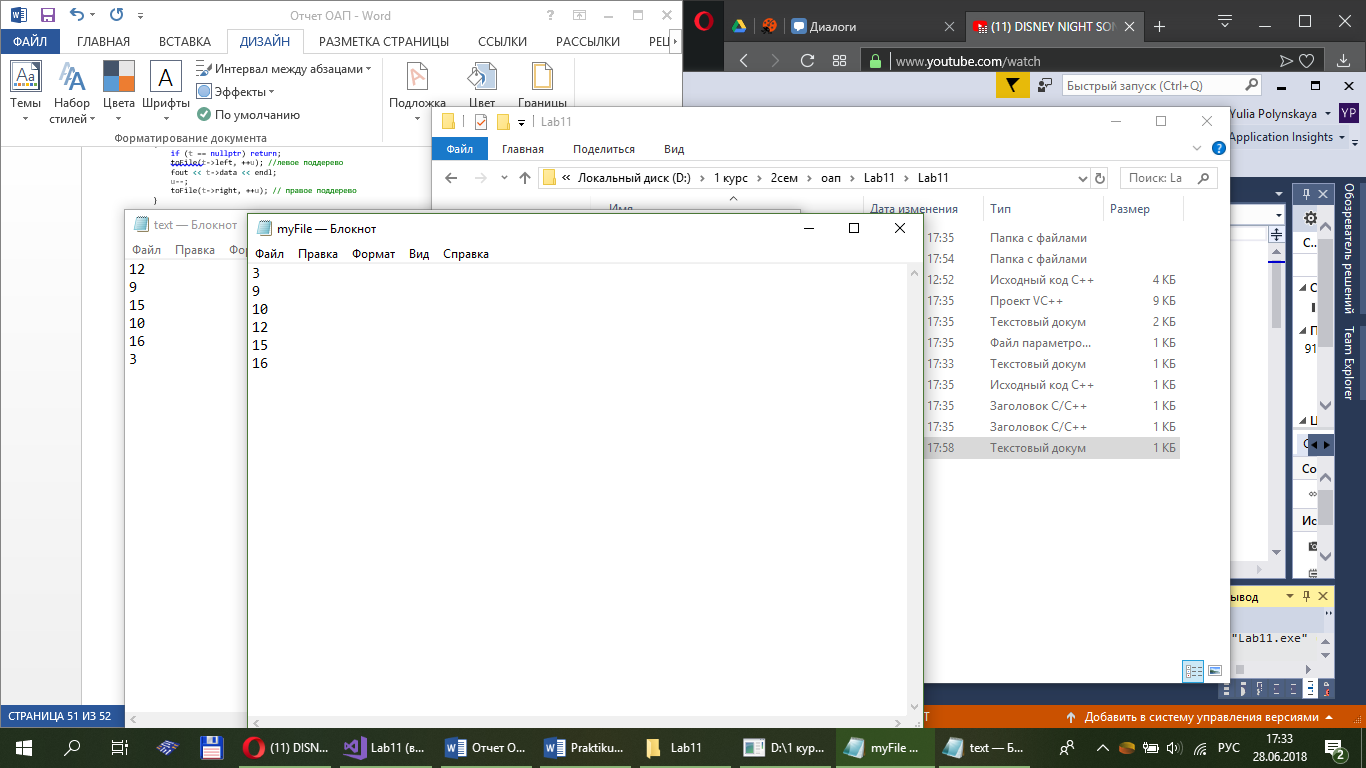
}

toFile(tree, 0);

toConsloe(tree,1);

system("pause");

}

  
  
  
**Лабораторная работа № 12.** [**Разработка проекта с использованием бинарного дерева**](#Лаб13)

Проект с использованием бинарного дерева в данной работе состоит из трех частей. Предполагается, что в дереве не более 10 уровней. Дублирование ключей не допускается.

4. Добавить к проекту  функции смешанного и нисходящего обхода дерева с выводом  на консоль, проверки сбалансированности дерева и функцию в соответствии с вариантом из таблицы, представленной в лабораторной работе № 11, изменив ее так, чтобы функция соответствовала проекту данной лабораторной работы.

**Tree.h**#pragma once

#include <iostream>

using namespace std;

namespace btree

{

struct NodeTree

{

int key;

};

enum CMP { LESS = -1, EQUAL = 0, GREAT = 1 };

struct Node //Узел бинарного списка

{

Node\* Parent; // указатель на родителя

Node\* Left; // указатель на левую ветвь

Node\* Right; // указатель на правую ветвь

void\* Data; // данные

Node(Node\* p, Node\* l, Node\* r, void\* d) // конструктор

{

Parent = p; Left = l; Right = r; Data = d;

}

Node\* Next(); // следующий по ключу

Node\* Prev(); // предыдущий по ключу

Node\* Min(); // минимум в поддереве

Node\* Max(); // максимум в поддереве

void ScanDown(void(\*f)(void\* n)); // обход поддерева сверху вниз

void Scan(int(\*f)(void\* n));

void ScanLevel(void(\*f)(void\* n), int);

void DescScan(void(\*f) (void\* n));

int GetLevel();

int ViewTreeN(int level);

void ScanByLevel(void(\*f)(void\* n));

void MixedScan(void(\*f) (void\* n));

};

struct Object // Интерфейс бинарного дерева

{

Node\* Root; // указатель на корень

CMP(\*Compare)(void\*, void\*); // функция сравнения

Object(CMP(\*f)(void\*, void\*))

{

Root = NULL; Compare = f;

};

Node\* Max() { return Root->Max(); };

Node\* Min() { return Root->Min(); };

bool isLess(void\* x1, void\* x2) const

{

return Compare(x1, x2) == LESS;

};

bool isGreat(void\* x1, void\* x2) const

{

return Compare(x1, x2) == GREAT;

};

bool isEqual(void\* x1, void\* x2) const

{

return Compare(x1, x2) == EQUAL;

};

bool Insert(void\* data); // добавить элемент

Node\* Search(void\* d, Node\* n); // найти по ключу

Node\* Search(void\* d)

{

return Search(d, Root);

};

bool Delete(Node\* e); // удалить по адресу элемента

bool Delete(void\* data)

{

return Delete(Search(data));//удалить по ключу

};

void ScanDown(void(\*f)(void\* n))

{

Root->ScanDown(f);

}; // обход дерева

btree::Object BuildTree(char \*);

void SaveToFile(void \*);

void SaveTree(btree::Object tree, char \*);

//int ViewTreeN(Node \*p, int level);

};

Object Create(CMP(\*f)(void\*, void\*)); // Создать бинарное дерево

}; **TreeC.cpp**#include "stdafx.h"

#include "Tree.h"

#include <iostream>

#pragma once

using namespace std;

namespace btree // бинарное дерево, не допускается дублирование ключей

{

Object Create(CMP(\*f)(void\*, void\*))

{

return \*(new Object(f));

}

Node\* Node::Min()

{

Node\* rc = this; if (rc->Left != NULL)

rc = rc->Left->Min();

return rc;

}

Node\* Node::Next()

{

Node\* rc = this, \*x = this;

if (rc->Right != NULL) rc = rc->Right->Min();

else

{

rc = this->Parent;

while (rc != NULL && x == rc->Right)

{

x = rc;

rc = rc->Parent;

}

}

return rc;

}

void Node::ScanDown(void(\*f)(void\* n))//вывод

{

f(this->Data);

std::cout << std::endl;

if (this->Left != NULL) this->Left->ScanDown(f);

if (this->Right != NULL) this->Right->ScanDown(f);

}

Node\* Object::Search(void\* d, Node\* n)//найти по ключу

{

Node\* rc = n;

if (rc != NULL)

{

if (isLess(d, n->Data)) rc = Search(d, n->Left);

else if (isGreat(d, n->Data)) rc = Search(d, n->Right);

}

return rc;

}

bool Object::Insert(void\* d)//добавить

{

Node\* x = this->Root, \*n = NULL;

bool rc = true;

while (rc == true && x != NULL)

{

n = x;

if (isLess(d, x->Data)) x = x->Left; //выбор куда идти - влево или вправо

else if (isGreat(d, x->Data)) x = x->Right;

else rc = false;

}

if (rc == true && n == NULL) this->Root = new Node(NULL, NULL, NULL, d);

else if (rc == true && isLess(d, n->Data))

n->Left = new Node(n, NULL, NULL, d);

else if (rc == true && isGreat(d, n->Data))

n->Right = new Node(n, NULL, NULL, d);

return rc;

};

bool Object::Delete(Node\* n)

{

bool rc = true;

if (rc = (n != NULL))

{

if (n->Left == NULL && n->Right == NULL) //если потомков нет

{

if (n->Parent == NULL) this->Root = NULL; //обнуление корня

else if (n->Parent->Left == n) n->Parent->Left = NULL;

else n->Parent->Right = NULL;

delete n;

}

else if (n->Left == NULL && n->Right != NULL) //только правый потомок

{

if (n->Parent == NULL) this->Root = n->Right;

else if (n->Parent->Left == n) n->Parent->Left = n->Right;

else n->Parent->Right = n->Right;

n->Right->Parent = n->Parent;

delete n;

}

else if (n->Left != NULL && n->Right == NULL)//только левый пото-мок

{

if (n->Parent == NULL) this->Root = n->Left;

else if (n->Parent->Right == n) n->Parent->Left = n->Left;

else n->Parent->Right = n->Left;

n->Left->Parent = n->Parent;

delete n;

}

else if (n->Left != NULL && n->Right != NULL) //если есть оба потомка

{

Node\* x = n->Next(); n->Data = x->Data; rc = Delete(x);

}

}

return rc;

}

void Node::ScanLevel(void(\*f)(void\* n), int i) //Вывести вершины уровня

{

if (this->Left != NULL) this->Left->ScanLevel(f, i);

if (this->GetLevel() == i) f(this->Data);

if (this->Right != NULL) this->Right->ScanLevel(f, i);

}

int Node::GetLevel()

{

Node \*rc = this; int q = 0;

while (rc->Parent != NULL)

{

rc = rc->Parent; q++;

}

return q;

}

void Node::ScanByLevel(void(\*f)(void\* n))

{

for (int i = 0; i < 10; i++)

{

std::cout << '\t';

this->ScanLevel(f, i);

std::cout << '\n';

}

}

void Node::DescScan(void(\*f) (void\* n))

{

f(this->Data); //обработка узла дерева

if (this->Left != NULL)

this->Left->DescScan(f); //рекурс.вызов для левого поддерева

if (this->Right != NULL)

this->Right->DescScan(f); //рекурс.вызов для правого поддерева

}

void Node::MixedScan(void(\*f) (void\* n))

{

if (this->Left != NULL)

this->Left->MixedScan(f); //рекурс.вызов для левого поддерева

f(this->Data); //обработка узла дерева

if (this->Right != NULL)

this->Right->MixedScan(f); //рекурс.вызов для правого поддерева

}

int Node::ViewTreeN(int level)

{

if (this == NULL)

return 0;

else

if (level == 0)

{

cout << "(" << ((NodeTree\*)this->Data)->key << ")" << " ";

return 1;

}

else

{

return (this->Right->ViewTreeN(level - 1) + this->Left->ViewTreeN(level - 1));

}

}

} **Tree.cpp**// Tree.cpp : Defines the entry point for the console application.

//Добавить к проекту функции смешанного и нисходящего обхода дерева с выводом на консоль, проверки сбалансированности дерева

#define \_CRT\_SECURE\_NO\_DEPRECATE //для устаревших ф-ций, которые не проверяют переполнение буфера

#include "stdafx.h"

#include "Tree.h"

#include <fstream>

#pragma warning (disable : 4996)//для устаревших ф-ций, которые не проверяют переполнение буфера

using namespace std;

struct NodeTree

{

int key;

};

btree::CMP cmpfnc(void\* x, void\* y) // Сравнение |область видимости|бинарный оператор|имя|

{

btree::CMP rc = btree::EQUAL;

if (((NodeTree\*)x)->key < ((NodeTree\*)y)->key) rc = btree::LESS;

else if (((NodeTree\*)x)->key >((NodeTree\*)y)->key) rc = btree::GREAT;

return rc;

}

void Print(void\* x)// Вывод при обходе

{

cout << ((NodeTree\*)x)->key << ends;

}

bool BuildTree(char \*FileName, btree::Object& tree)//Построение дерева из файла

{

bool rc = true;

FILE \*inFile = fopen(FileName, "r");

if (inFile == NULL)

{

cout << "Ошибка открытия входного файла" << endl; rc = false;

}

cout << " Исходные данные" << endl; // заполнение дерева и вывод исходных данных

while (!feof(inFile))

{

int num;

fscanf(inFile, "%d", &num, 1);

NodeTree \*a = new NodeTree();

a->key = num; tree.Insert(a); cout << num << endl;

}

fclose(inFile); return rc;

}

FILE \* outFile;

int level;

void SaveToFile(void \*x)// Запись одного элемента в файл

{

NodeTree \*a = (NodeTree\*)x;

int q = a->key;

fprintf(outFile, "%d\n", q);

}

void SaveTree(btree::Object &tree, char \*FileName) //Сохранение дерева в файл

{

outFile = fopen(FileName, "w");

if (outFile == NULL)

{

cout << "Ошибка открытия выходного файла" << endl; return;

}

tree.Root->ScanDown(SaveToFile);

fclose(outFile);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale(LC\_CTYPE, "Russian");

char fl[10] = "G.txt";

btree::Object t1 = btree::Create(cmpfnc);

int k, choise;

for (;;)

{

NodeTree \*a = new NodeTree;

cout << endl;

cout << "1 - вывод дерева на экран" << endl;

cout << "2 - добавление элемента" << endl;

cout << "3 - удаление элемента" << endl;

cout << "4 - сохранить в файл" << endl;

cout << "5 - загрузить из файла" << endl;

cout << "6 - очистить дерево" << endl;

cout << "7 - нисходящий обход" << endl;

cout << "8 - смешанный обход (по возрастанию)" << endl;

cout << "0 - выход" << endl;

cout << "сделайте выбор" << endl; cin >> choise;

switch (choise)

{

case 0: exit(0);

case 2:

cout << "введите ключ" << endl; cin >> k;

a->key = k;

t1.Insert(a); break;

case 1:

if (t1.Root) t1.Root->ScanByLevel(Print);

else cout << "Дерево пустое" << endl; break;

case 3:

cout << "введите ключ" << endl; cin >> k;

a->key = k;

t1.Delete(a); break;

case 4:

SaveTree(t1, fl); break;

case 5:

BuildTree(fl, t1); break;

case 6:

{

while (t1.Root) t1.Delete(t1.Root);

}

break;

case 7:

if (t1.Root) t1.Root->DescScan(Print);

else cout << "Дерево пустое" << endl; break;

case 8:

if (t1.Root) t1.Root->MixedScan(Print);

else cout << "Дерево пустое" << endl; break;

}

}

return 0;

}  
**Лабораторная работа № 13. Бинарные кучи**

**Бинарная куча** (binary heap) представляет собой бинарное дерево, для которого выполняется основное свойство кучи: приоритет каждой вершины **больше** приоритетов её потомков. В простейшем случае приоритет можно считать равным значению.

Реализация бинарной кучи возможна на основе *массива* и на основе *списка*.

5.В проект добавить следующие функции: удаление минимального **extractMin**; удаление i-ого элемента **extractI**; объединение **unionHeap** двух куч в одну.

**Heap.h**#pragma once

struct virus

{

int x;//приоритет

void Print();//прототип ф-ции

int GetPriority() const;//прототип

};

namespace heap//куча, пространство имен

{

enum CMP//объединение

{

LESS = -1, EQUAL = 0, GREAT = 1

};

struct Heap

{

int Size; int MaxSize;//есть размер

void\*\* Storage;

CMP(\*Compare)(void\*, void\*);//хранилище для данных

Heap(int maxsize, CMP(\*f)(void\*, void\*))//

{

Size = 0; Storage = new void \*[MaxSize = maxsize]; Compare = f;

};

Heap(int maxsize, CMP(\*f)(void\*, void\*), void\* x[])

{

Size = 0; Storage = x; MaxSize = maxsize; Compare = f;

};

int Left(int ix); int Right(int ix);

int Parent(int ix);

bool isFull() const { return (Size >= MaxSize); };//переполнение

bool isEmpty() const { return (Size <= 0); };//пустой

bool isLess(void\* x1, void \* x2) const

{

return Compare(x1, x2) == LESS;

};

bool isGreat(void\* x1, void\* x2) const

{

return Compare(x1, x2) == GREAT;

};

bool isEqual(void\* x1, void\* x2) const

{

return Compare(x1, x2) == EQUAL;

};

void Swap(int i, int j);

void Heapify(int ix);

void Insert(void\* x);

void\* ExtractMax();

void\* ExtractMin();

void DeleteHeap();

void\* ExtractI(int i);

void Union(Heap H);

void Scan(int i) const;

};

Heap Create(int maxsize, CMP(\*f)(void\*, void\*));//создает кучу

};

**HeapC.cpp**#include "stdafx.h"

#include "Heap.h"

#include <iostream>

#include <iomanip>

void virus::Print()//вывод х

{

std::cout << x;

}

int virus::GetPriority() const { return x; }//возвращаем х, х -это приоритет

namespace heap

{

Heap Create(int maxsize, CMP(\*f)(void\*, void\*))//выделяет память под кучу

{

return \*(new Heap(maxsize, f));

}

int Heap::Left(int ix)//решается куда размещается эта ф-ция

{

return (2 \* ix + 1 >= Size) ? -1 : (2 \* ix + 1);//?:\это if

}

int Heap::Right(int ix)

{

return (2 \* ix + 2 >= Size) ? -1 : (2 \* ix + 2);

}

int Heap::Parent(int ix) { return (ix + 1) / 2 - 1; }//смотрит на родителя

void Heap::Swap(int i, int j)//поменять местами

{

void\* buf = Storage[i]; Storage[i] = Storage[j]; Storage[j] = buf;

}

void Heap::Heapify(int ix)//сортировка

{

int l = Left(ix), r = Right(ix), irl = ix;

if (l > 0)

{

if (isGreat(Storage[l], Storage[ix])) irl = l;

if (r > 0 && isGreat(Storage[r], Storage[irl])) irl = r;

if (irl != ix) { Swap(ix, irl); Heapify(irl); }

}

}

void Heap::Insert(void\* x)//вставить

{

int i;

if (!isFull())//переполнение

{

Storage[i = ++Size - 1] = x;

while (i > 0 && isLess(Storage[Parent(i)], Storage[i]))

{

Swap(Parent(i), i); i = Parent(i);

}

}

}

void\* Heap::ExtractMax()//удаление макс

{

void\* rc; rc = NULL;

if (!isEmpty())

{

rc = Storage[0]; Storage[0] = Storage[Size - 1];

Size--; Heapify(0);

} return rc;

}

void Heap::Scan(int i) const //Вывод значений элементов на экран

{

int probel = 20; std::cout << '\n';

if (Size == 0) std::cout << "Куча пуста";

for (int u = 0, y = 0; u < Size; u++)

{

std::cout << std::setw(probel + 10) << std::setfill(' ');

((virus\*)Storage[u])->Print();

if (u == y) { std::cout << '\n'; if (y == 0) y = 2; else y += y \* 2; }

probel /= 2;

}

std::cout << '\n';

}

void Heap::DeleteHeap()

{

if (!isEmpty())

{

Size = 0; this->~Heap();

}

}

void\* Heap::ExtractMin()

{

int min = 0;

void\* rc; rc = NULL;

if (!isEmpty())//не пустой

{

for (int i = 1; i < Size; i++)

if (((virus\*)Storage[min])->x>((virus\*)Storage[i])->x) min = i;//запоминаем индекс мин эл-та

rc = Storage[min];

Storage[min] = Storage[Size - 1];

Size--;

Heapify(0);

} return rc;

}

void\* Heap::ExtractI(int i)

{

void\*rc; rc = NULL;

if (!isEmpty())

{

if (i < Size)

{

rc = Storage[i];

Storage[i] = Storage[Size - 1];

Size--;

Heapify(0);

}

else

std::cout << "Нет такого элемента\n";

}

return rc;

}

void Heap::Union(Heap H)

{

for (int i = 0; i < H.Size; i++)

Insert(H.Storage[i]);

}

}

**Heap.cpp**#include "stdafx.h"

#include <iostream>

#include <fstream>

#include "Heap.h"

using namespace std;

heap::CMP cmpvirus(void\* a1, void\* a2) //Функция сравненияиз пространства heap (Сompare)

{

#define A1 ((virus\*)a1)

#define A2 ((virus\*)a2)

heap::CMP rc = heap::EQUAL;

if (A1->x > A2->x) rc = heap::GREAT;

else if (A2->x > A1->x) rc = heap::LESS;

return rc;

#undef A2

#undef A1

}

bool BuildHeap(char \*FileName, heap::Heap& h) // Построение кучи из файла

{

bool rc = true; int n;

ifstream inFile;

inFile.open(FileName, std::ios::out);

if (!inFile)

{

cout << "Невозможно открыть файл" << endl; exit(1);

}

while (inFile >> n)

{

int \*a = new int;

\*a = n; h.Insert((void\*)a);

}

inFile.close(); return rc;

}

void SaveHeap(heap::Heap &h, char \*FileName) // Функция записи в файл

{

ofstream outFile(FileName, ios\_base::out | ios\_base::trunc);

if (!outFile)

{

cout << "Ошибка открытия выходного файла" << std::endl;

return;

}

int \*a = new int;

for (int u = 0, y = 0; u < h.Size; u++)

{

a = (int\*)h.Storage[u];

outFile << \*a;

outFile << endl;

}

outFile.close();

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale(LC\_ALL, "rus");

char fl[10] = "G.txt";

int k, i;

virus \*a; a = NULL;

int choise;

heap::Heap h1 = heap::Create(60, cmpvirus);

heap::Heap h2 = heap::Create(30, cmpvirus);

for (;;)

{

cout << "1 - вывод кучи на экран" << endl;

cout << "2 - добавить элемент" << endl;

cout << "3 - удалить максимальный элемент" << endl;

cout << "4 - очистить кучу" << endl;

cout << "5 - сохранить в файл" << endl;

cout << "6 - загрузить из файла" << endl;

cout << "7 - удалить минимальный элемент" << endl;

cout << "8 - удалить i-ый элемент" << endl;

cout << "9 - Объединение 2 куч" << endl;

cout << "0 - выход" << endl;

cout << "сделайте выбор" << endl; cin >> choise;

switch (choise)

{

case 0: exit(0);

case 1: h1.Scan(0); break;

case 2: a = new virus;//выделяем память

cout << "введите ключ" << endl; cin >> k;

a->x = k;//в поле х вставляем значение к

h1.Insert(a); break;

case 3: h1.ExtractMax(); break;//вызываем фцию

case 4: h1.DeleteHeap(); break;

case 5: SaveHeap(h1, fl); break;

case 6: h1.DeleteHeap();

BuildHeap(fl, h1); break;

case 7: h1.ExtractMin(); break;

case 8: cout << "Введите номер удаляемого элемента ";

cin >> i;

virus\*p;

p = (virus\*)h1.ExtractI(i);//присваиваем значение фции

if (p != NULL)

cout << "Удаленный элемент " << p->x << endl; break;

case 9: cout << "Первая куча\n";

h1.Scan(0);//вывод первой

BuildHeap(fl, h2);//из файла вторая куча

cout << "Вторая куча записана из файла\nВторая куча\n";

h2.Scan(0);

h1.Union(h2);

h2.DeleteHeap();

cout << "Результирующая куча\n";

h1.Scan(0); break;

case 10:

BuildHeap(fl, h1);

break;

default: cout << endl << "Введена неверная команда!" << endl;

}

} return 0;

}

**Лабораторная работа № 14. Хеш-таблицы c открытой адресацией**

**Хеш-табли́ца** (перемешанная таблица) − это структура данных, которая позволяет хранить пары (ключ, значение) и выполнять три операции: добавления новой пары, поиска и удаления пары по ключу. Основное отличие  таблиц от других динамических множеств – **вычисление адреса** элемента по значению ключа с помощью хеш-функции. Ситуация, когда для различных ключей получается одно и то же значение хеш-функции, называется *коллизией*.

Существуют два основных варианта хеш-таблиц: *с цепочками* (хеш-таблица содержит списки пар) и *открытой адресацией* (хеш-таблица представляет собой некоторый [массив](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), элементы которого есть пары).  
Использовать в проекте функции мультипликативного и модульного хеширования. Сравнить время поиска информации.  
#include "LR14\_h.h"

#include <conio.h>

#include <iostream>

//Использовать в проекте функции мультипликативного и

//модульного хеширования.Сравнить время поиска информации.

using namespace std;

struct AAA

{

int key;

char \*mas;

AAA(int k, char\*z)

{

key = k; mas = z;

} AAA() {}

};

//-------------------------------

int key(void\* d)

{

AAA\* f = (AAA\*)d; return f->key;

}

//-------------------------------

void AAA\_print(void\* d) //функция вывода хэш-таблицы

{

cout << "Ключ " << ((AAA\*)d)->key << " | " << ((AAA\*)d)->mas << "\n";

}

//-------------------------------

int main()

{

setlocale(LC\_ALL, "rus");

AAA a1(1, "one"), a2(2, "two"), a3(4, "three"), a4(2, "four");

int siz = 10;

cout << "Введите размер: ";

cin >> siz;

Object H = create(siz, key); //создать хэш-таблицу

int choice; int k;

for (;;)

{

cout << "1 - Вывод Хэш-таблицы;\n";

cout << "2 - Добавить элементt;\n";

cout << "3 - Удалить элемент;\n";

cout << "4 - Найти элемент;\n";

cout << "0 - Выход;\n\n";

cout << "Твой выбор: ";

cin >> choice;

switch (choice)

{

case 0:

exit(0);

case 1: //вывод

H.scan(AAA\_print);

break;

case 2: //добавление элемента

{

AAA \*a = new AAA;

char \*str = new char[20];

cout << "Введите элемент (ключ): ";

cin >> k;

a->key = k;

cout << "Введите строку: ";

cin >> str;

a->mas = str;

if (H.N == H.size)

cout << "Таблица заполнена.\n";

else

H.insert(a);

break;

}

case 3: //удаление элемента

{

cout << "Введите ключ для удаления: ";

cin >> k;

H.deleteByKey(k);

break;

}

case 4:

{

cout << "Введите ключ для поиска: ";

cin >> k;

if (H.search(k) == NULL)

cout << "Не найдено\n";

else

AAA\_print(H.search(k));

break;

}

default:

{

cout << "Не найдено\n";

break;

}

}

}

return 0;

}

**Лабораторная работа № 15. Хеш-таблицы c цепочками**

В хеш-таблицах с цепочками (с прямой адресацией) каждая ячейка массива  является указателем на связный *список* (цепочку) пар ключ-значение, соответствующих одному и тому же хеш-значению ключа. Коллизии здесь приводят к тому, что появляются цепочки длиной более одного элемента.