Лабораторная работа №1

Функции пользователя

1. В одномерном массиве найти минимальный и максимальный элементы. Вычислить их разность.

#include <iostream>

#include <time.h>

using namespace std;

int\* fmax(int A[], int n)

{ int i, \* pm;

for (i = 1, pm = A; i < n; i++)

if (A[i] > \* pm)

pm = &A[i];

return pm;

}

int\* fmin(int A[], int n)

{

int i, \* pm;

for (i = 1, pm = A; i < n; i++)

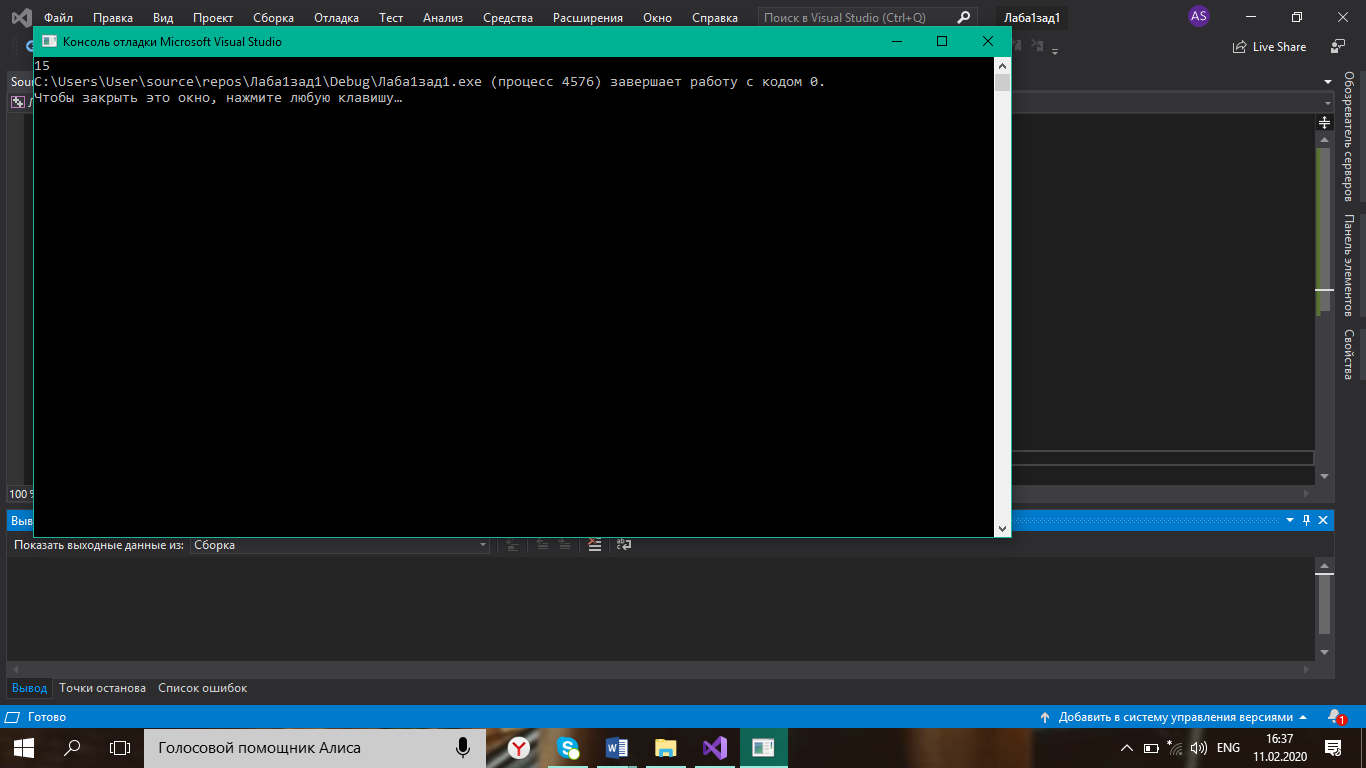
if (A[i] < \* pm)

pm = &A[i];

return pm;

}

void main()

 {

int A[] = { 8, 4, -3, -6, 9 };

int s = \*fmax(A, 5)-\*fmin(A,5);

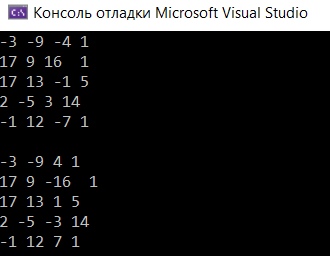
cout << s;

}

2. Если в матрице имеется столбец, все элементы которого положительны, то знаки элементов предыдущего столбца изменить на противоположные.

#include<iostream>

#include<ctime>

using namespace std;

int\*\* polel(int\*\* array, int n, int m);

int main()

{

int i, j, \*\* res;

int\*\* array = new int\* [5];

for (i = 0; i < 5; i++)

array[i] = new int[4];

srand((unsigned)time(NULL));

for (i = 0; i < 5; i++)

{

for (j = 0; j < 4; j++)

{

array[i][j] = rand() % 30 - 9;

cout << array[i][j] << ' '; }

cout << endl;

}

cout << endl;

res = polel(array, 5, 4);

for (i = 0; i < 5; i++)

{

for (j = 0; j < 4; j++)

cout << res[i][j] << ' ';

cout << endl;

}

for (int i = 0; i < 5; i++)

delete array[i];

delete[] array;

}

int\*\* polel(int\*\* array, int n, int m)

{

int i, j;

for (i = 0; i < n; i++)

{

for (j = 0; j < m; j++)

{

if (\*(\*(array + i) + j) > 0)

for (i = 0; i < n; i++)

{

\*(\*(array + i) + j - 1) = -\*(\*(array + i) + j - 1);

}

}

}

return array;

}

Лабораторная работа №2

Указатели на функции

1.

e x + x − 4,

x2 – 4

#include<iostream>

using namespace std;

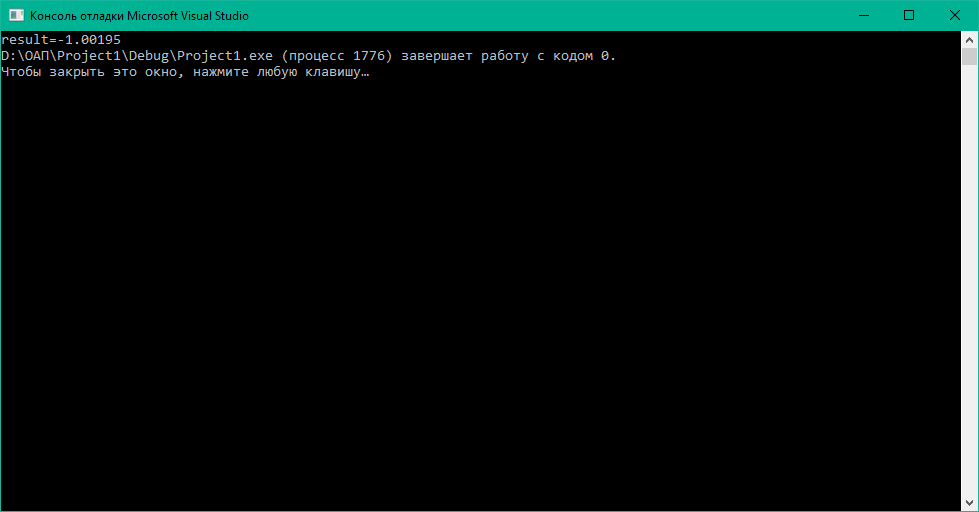
double ur(double(\*)(double), double, double, double);

double f(double);

double k(double);

int main()

{

 double z;

z = ur(f, -3, -1, 0.001);

cout << "result=" << z;

}

double ur(double(\*fun)(double), double a, double b, double e)

{

double x;

do

{

x = (a + b) / 2;

if (fun(x) \*fun(a)<=0)

b = x;

else a = x;

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

return x;

}

double f(double x)

{

return(exp(x) + x - 4);

}

double k(double x)

{

return(exp(x) + x - 4);

}

2. Написать функцию **compr**, которая «сжимает» строку, удаляя из нее все пробелы. С ее помощью сжать различные строки

#include <iostream>

using namespace std;

void removeElement(char\* str, int size, ...)

{

int\* index = &size; index += 2;

for (int i = \*index; i < size; i++)

str[i] = str[i + 1];

str[size] = '\0';

}

void compr(char\* str, int size)

{

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

if (str[i] == ' ') removeElement(str, --size, i);

else i++;

}

char str[255];

int main()

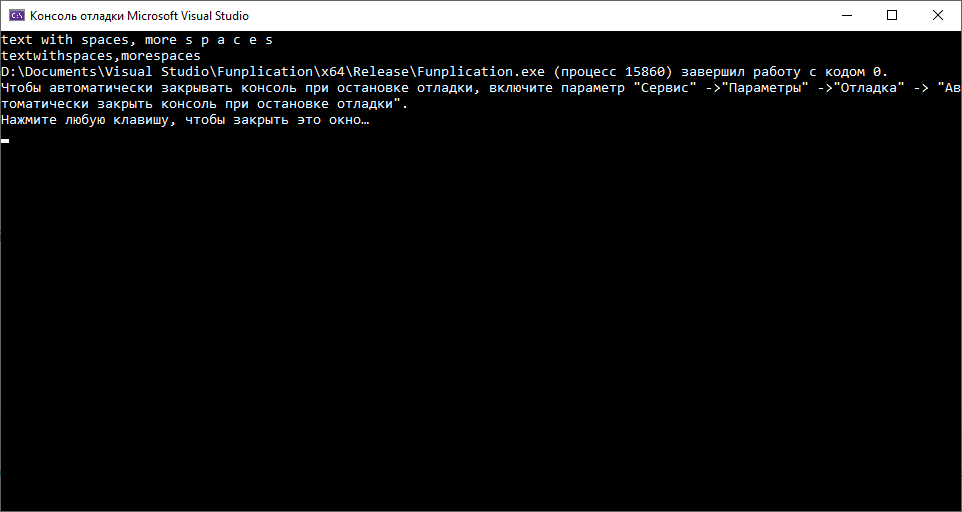
{

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

gets\_s(str);

compr(str, strlen(str));

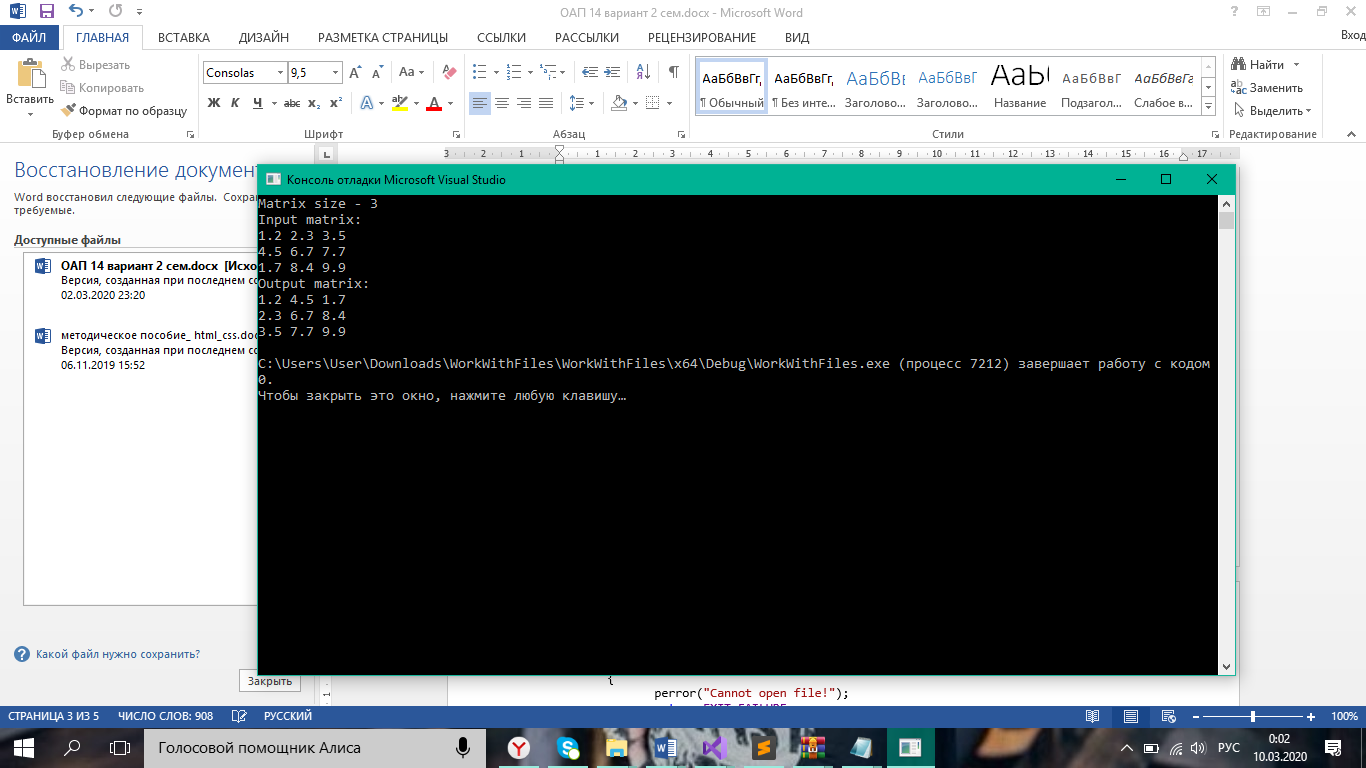
cout << str;

}

Лабораторная работа № 3.

Работа с файлами на языке С

1. Дан файл вещественных чисел, содержащий элементы квадратной матрицы по строкам, причем начальный элемент файла содержит значение количества столбцов матрицы. Создать новый файл той же структуры, содержащий **k**-ый столбец исходной матрицы

 #include <iostream>

using namespace std;

char str[255];

int main()

{

FILE\* open;

fopen\_s(&open, "input.txt", "r");

if (open == NULL)

{

perror("Cannot open file!");

return EXIT\_FAILURE;

}

fgets(str, sizeof(str), open);

int size = atoi(str), row\_size = 0;

cout << "Matrix size - " << size << endl;

double\*\* matrix = new double\* [size];

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

matrix[i] = new double[size];

while (fgets(str, sizeof(str), open))

{

int col\_size = 0;

for (int start = 0, end = 0; end <= strlen(str); end++)

{

if (str[end] == ' ' || str[end] == '\0')

{

char cifra[255]; for (int i = 0; i < 255; cifra[i++] = 0);

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

cifra[i - start] = str[i];

matrix[row\_size][col\_size++] = atof(cifra);

start = end + 1;

}

}

row\_size++;

}

cout << "Input matrix: " << endl;

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

{

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

cout << matrix[i][j] << ' ';

cout << endl;

}

FILE\* output;

fopen\_s(&output, "output.txt", "w");

if (output == NULL)

{

cout << "Cannot open file!";

return EXIT\_FAILURE;

}

cout << "Output matrix: " << endl;

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

{

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

{

cout << matrix[j][i] << ' ';

fprintf\_s(output, "%f ", matrix[j][i]);

}

cout << endl;

fputs("\n", output);

}

}

2. Компоненты файла **f** – целые числа. Получить файл **g**, образованный из файла **f** исключением повторных вхождений одного и того же числа.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include <stdlib.h>

int main()

{

system("chcp 1251");

system("cls");

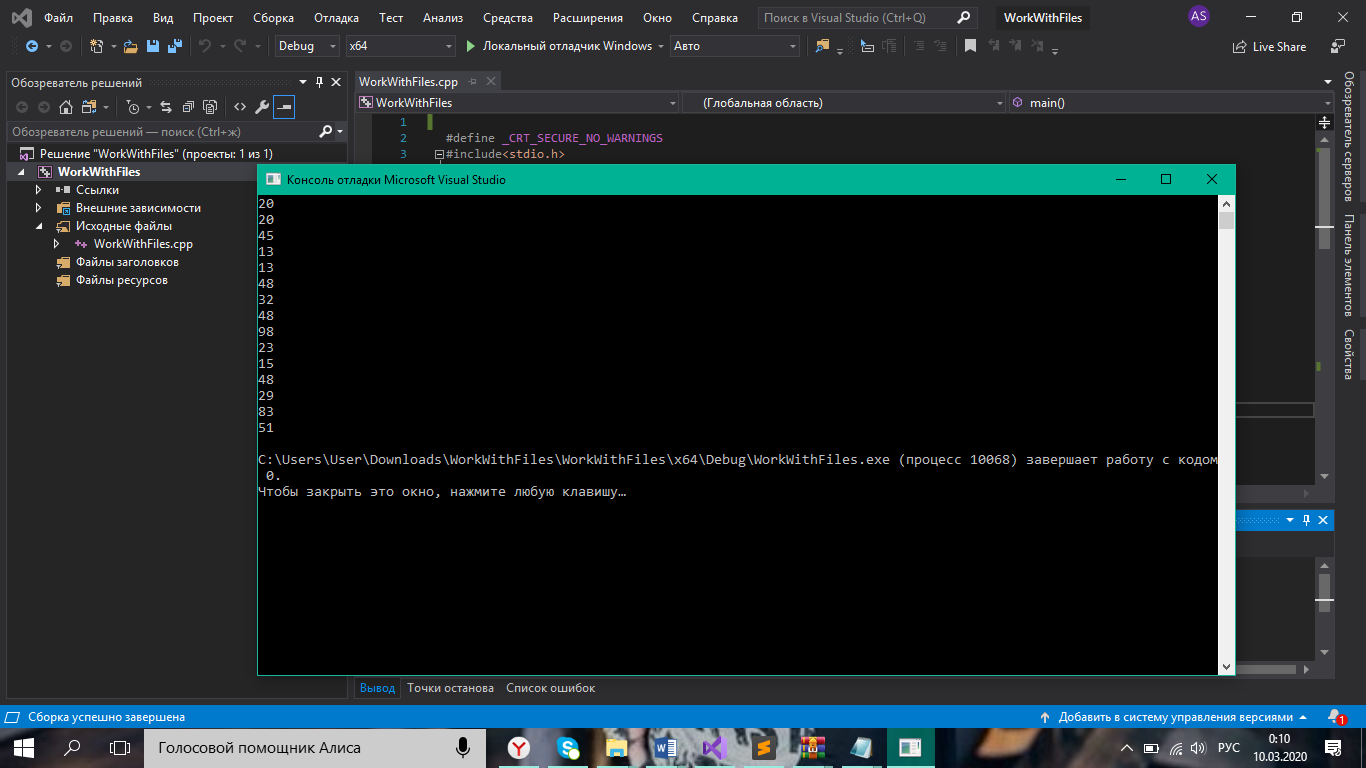
FILE\* f, \* g;

int array[15] = { 20,20,45,13,13,48,32,48,98,23,15,48,29,83,51 };

int\* new\_array = new int[15];

int f1, f2;

for (int i = 0; i < 15; i++)

 {

printf("%d\n", array[i]);

}

f1 = fopen\_s(&f, "f.txt", "w");

for (int i = 0; i < 15; i++)

{

fprintf(f, "%d\n", array[i]);

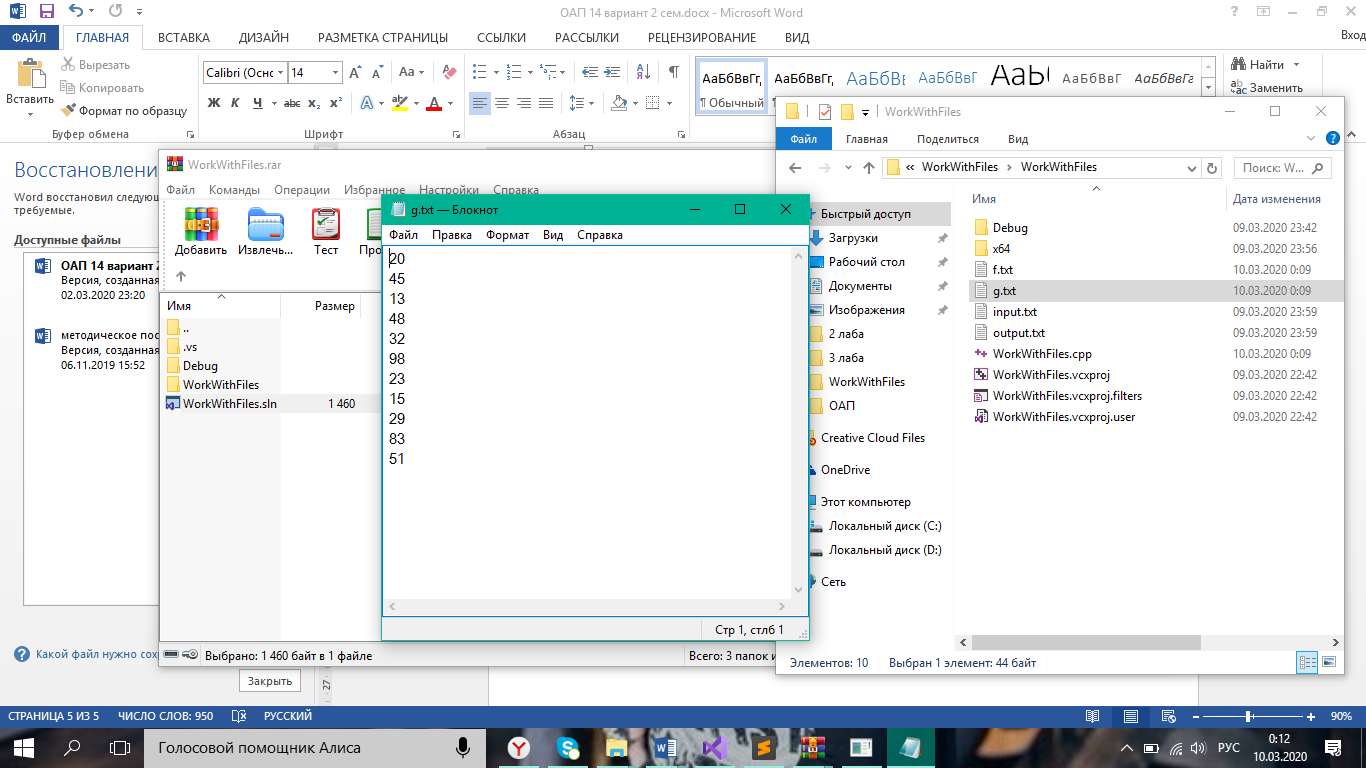
}

fclose(f);

f1 = fopen\_s(&f, "f.txt", "r");

f2 = fopen\_s(&g, "g.txt", "w");

int counter\_for\_new\_array = 0;

 for (int i = 0; i < 15; i++)

{

bool no\_insert = false;

for (int k = 0; k < counter\_for\_new\_array; k++)

{

if (new\_array[k] == array[i])

{

no\_insert = true;

}

}

if (!no\_insert)

{

new\_array[counter\_for\_new\_array++] = array[i];

}

}

for (int i = 0; i < counter\_for\_new\_array; i++)

{

if (new\_array[i] != '\0')

{

fprintf(g, "%d\n", new\_array[i]);

}

}

fclose(g);

fclose(f);

}

Лабораторная работа № 4

Работа с файлами С++

1. Скопировать из файла **FILE1** в файл **FILE2** все строки, которые не содержат цифры. Подсчитать количество строк, которые начинаются на букву «**А**» в файле **FILE2**.

#include <iostream>

#include <fstream>

#include<string>

#include<Windows.h>

using namespace std;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string str;

ifstream in("FILE1.txt");

ofstream out("FILE2.txt");

if (!in.is\_open())

{

cout<< "";

}

int p = 0;

while (in >> str)

{

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

{

p = 0;

if (str[i] == '0' || str[i] == '1' || str[i] == '2' || str[i] == '3' || str[i] == '4' || str[i] == '5' || str[i] == '6' || str[i] == '7' ||

str[i] == '8' || str[i] == '9')

{

p = 1;

if (p == 1)break;

}

}

if (p != 1)

{

out << str << " ";

}

}

in.close();

out.close();

ifstream fin("FILE2.txt"); int colich = 0;

while (fin >> str)

{

if (str[0] == 'А')

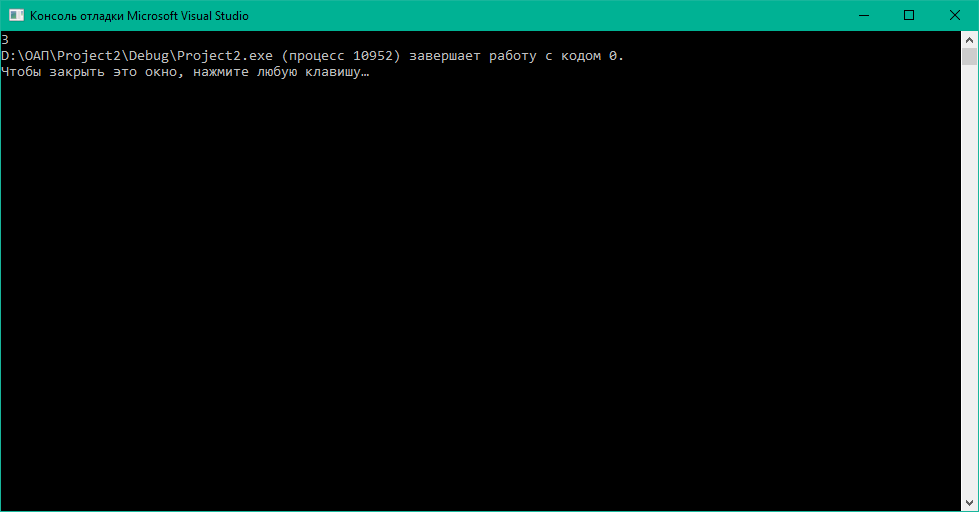
colich++;

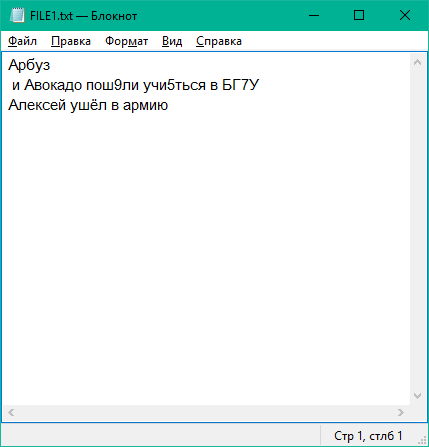
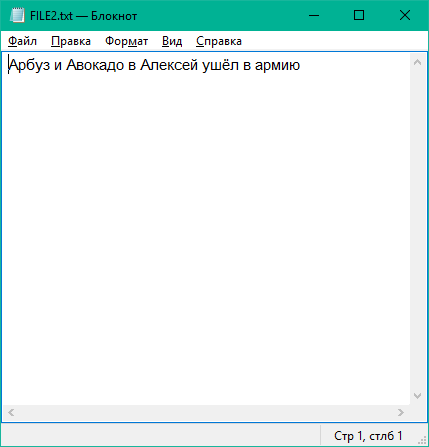
}

cout << colich;

fin.close();

}



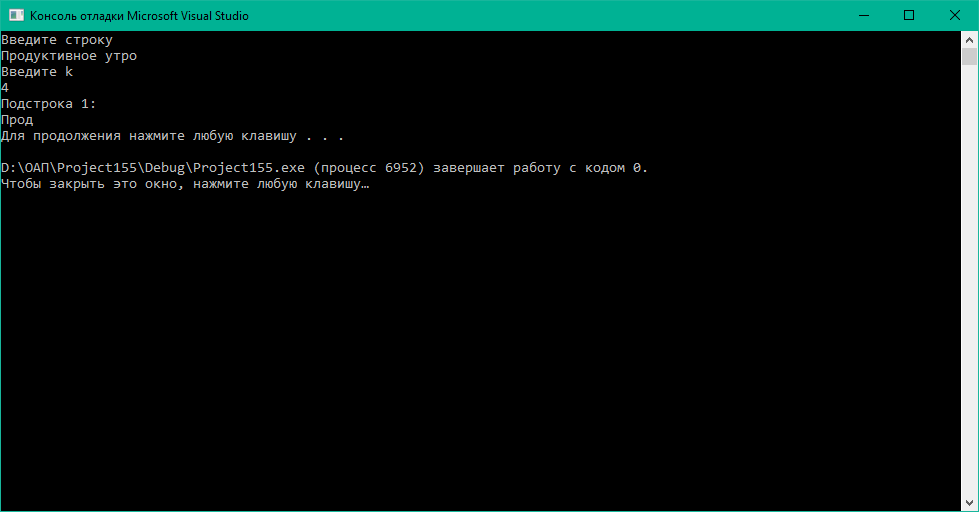


2. Ввести с клавиатуры строку, состоящую из слов, и записать ее в файл. Прочитать из файла данные. Разбить строку на две подстроки, первая из которых имеет длину **k** символов.

#include <iostream>

#include <fstream>

#include<windows.h>

using namespace std;

int main() {

setlocale(0, "");

SetConsoleCP(1251);

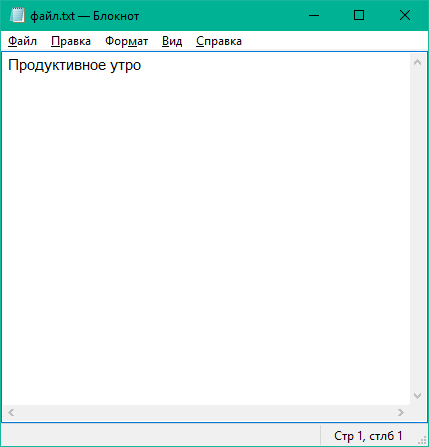
SetConsoleOutputCP(1251);

char str[30], str1[30], str2[30];

int k;

ofstream out("файл.txt");

cout << "Введите строку\n";

 gets\_s(str);

out << str << endl;

out.close();

cout << "Введите k\n";

cin >> k;

ifstream fin("файл.txt");

fin.getline(str1, k + 1);

cout << "Подстрока 1:\n" << str1 << endl;

system("pause");

return 0;

}

Лабораторная работа № 5

Представление информации в виде структуры

**Справочник абитуриента.** Наименование вуза, адрес, перечень специальностей, конкурс прошлого года по каждой специальности. Выбор по специальности.

#include<iostream>

#include<windows.h>

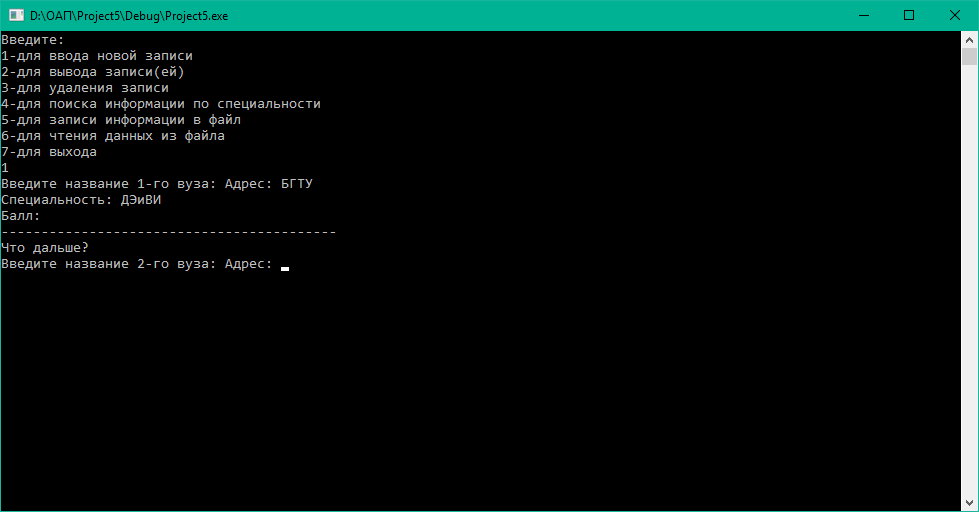
#include <fstream>

#define size 10

using namespace std;

void enter\_new();

void del();

void poisk();

void file\_out();

void out();

void file\_if();

struct Abit

{

char FIO[50];

char type[30];

int number;

int sym;

char date[12];

};

int op, size\_end = 0;

struct Abit a[size];

int main()

{

setlocale(0, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Введите:" << endl;

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

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

cout << "3-для удаления записи" << endl;

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

cout << "5-для записи информации в файл" << endl;

cout << "6-для чтения данных из файла" << endl;

cout << "7-для выхода" << endl;

cin >> op;

do

{

switch (op)

{

case 1: enter\_new(); break;

case 2: out(); break;

case 3: del(); break;

case 4: poisk(); break;

case 5: file\_out(); break;

case 6: file\_if(); break;

}

} while (op != 7);

system("pause");

}

void poisk()

{

int num;

cout << "Введите специальность: ";

cin >> num;

cout << endl;

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

{

if (num == a[i].number)

{

cout << "Название вуза: " << a[i].FIO << endl;

cout << "Адрес: " << a[i].type << endl;

cout << "Специальность: " << a[i].number << endl;

cout << "Балл: " << a[i].sym << endl;

cout << endl << "------------------------------------------ " << endl;

cout << "Что дальше?" << endl;

cin >> op;

break;

}

}

}

void enter\_new()

{

if (size\_end < size)

{

cout << "Введите название " << size\_end + 1 << "-го вуза: ";

gets\_s(a[size\_end].FIO);

cout << "Адрес: ";

gets\_s(a[size\_end].type);

cout << "Специальность: ";

cin >> a[size\_end].number;

cout << "Балл: ";

cin >> a[size\_end].sym;

size\_end++;

cout << endl << "------------------------------------------ " << endl;

cout << "Что дальше?" << endl;

cin >> op;

}

}

void del()

{

int d;

cout << "Номер вуза , который нужно удалить: "; cin >> d;

for (int de1 = (d - 1); de1 < size\_end; de1++)

a[de1] = a[de1 + 1];

size\_end = size\_end - 1;

cout << "Что дальше?" << endl;

cin >> op;

}

void out()

{

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

{

cout << "Название вуза: " << a[i].FIO << endl;

cout << "Адрес: " << a[i].type << endl;

cout << "Специальность: " << a[i].number << endl;

cout << "Балл: " << a[i].sym << endl;

cout << endl << "------------------------------------------ " << endl;

}

cout << "Что дальше?" << endl;

cin >> op;

}

void file\_out()

{

int s = 0;

ofstream fout("FILE.txt");

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

{

fout << "Название вуза: " << a[i].FIO << endl;

fout << "Адрес: " << a[i].type << endl;

fout << "Специальность: " << a[i].number << endl;

fout << "Балл: " << a[i].sym << endl;

fout << endl << "------------------------------------------ " << endl;

s = 1;

}

if (s = 1)

{

cout << "Инофрмация записана в файл" << endl;

}

else

{

cout << "Информация НЕ ЗАПИСАНА в файл !!!" << endl;

}

cout << "Что дальше?" << endl;

cin >> op;

fout.close();

}

void file\_if()

{

ifstream fin("FILE.txt");

char str[50];

for (int i = 0; i < (7 \* size\_end); i++)

{

fin.getline(str, 50);

cout << str << endl;

}

fin.close();

cout << "Что дальше?" << endl;

cin >> op;

}

Лабораторная работа № 6

Объединения, перечисления, битовые поля

**Список клиентов гостиницы.** Паспортные данные, даты приезда и отъезда, номер, тип размещения (люкс, одноместный, двухместный, трехместный, апартаменты). Поиск гостя по дате приезда. Даты приезда и отъезда реализовать с помощью битового поля, тип размещения − с помощью перечисления.

#include <iostream>

#define size 100

#define large 30

enum type {

lux = 1,

one,

two,

three,

apartments

} aparts;

struct hotel {

char passport[size];

int place = aparts;

int number;

unsigned int yearp : 12;

unsigned int monthp : 4;

unsigned int dayp : 5;

unsigned int yearo : 12;

unsigned int montho : 4;

unsigned int dayo : 5;

};

hotel arr[large];

hotel order66;

int i = 0;

int input();

int removal();

int output();

int search();

using namespace std;

int main() {

setlocale(LC\_ALL, "ru");

int k = 0;

int yon = 0;

cout << "Что вы хотите сделать?\n";

do {

system("cls");

cout << "1 - Ввод данных в структуру\n";

cout << "2 - Удаление данных из структуры\n";

cout << "3 - Вывод данных на экран\n";

cout << "4 - Поиск по дате прибытия\n";

cout << "5 - Для завершения работы\n";

cin >> k;

switch (k)

{

case 1:

yon = input();

break;

case 2:

yon = removal();

break;

case 3:

yon = output();

break;

case 4:

yon = search();

break;

default:

yon = 0;

cout << "Рад помочь вам!\n";

break;

}

} while (yon == 1);

return 0;

}

int input() {

int yon = 0;

cout << "Ввод данных в структуру\n";

if (i < large) {

int yearp, yearo, monthp, montho, dayp, dayo;

cout << "Введите пасспортные данные: ФИО\n";

gets\_s(arr[i].passport);

gets\_s(arr[i].passport);

cout << "Введите дату приезда(год, месяц, день)\n";

cin >> yearp >> monthp >> dayp;

arr[i].yearp = yearp;

arr[i].monthp = monthp;

arr[i].dayp = dayp;

cout << "Введите дату отъезда(год, месяц, день)\n";

cin >> yearo >> montho >> dayo;

arr[i].yearo = yearo;

arr[i].montho = montho;

arr[i].dayo = dayo;

cout << "Введите номер в гостинице\n";

cin >> arr[i].number;

cout << "Введите тип размещения(1 - люкс, 2 - одноместный, 3 - двухместный, 4 - трёхместный, 5 - апартаменты)\n";

cin >> arr[i].place;

i++;

}

else {

cout << "Число строк уже максимально\n";

}

cout << "Что дальше?(1-продолжить,0-выйти)\n";

cin >> yon;

while (yon != 0 && yon != 1) {

cout << "Вы ввели неверное значение. Повторите\n";

cin >> yon;

}

return yon;

}

int removal() {

int yon = 0;

if (i > 0) {

int numb, str;

cout << "Введите номер строки, которую вы хотите удалить<---->66 для удаления всех строк\n";

cin >> numb;

if (numb != 66) {

for (str = numb - 1; str < i; str++) {

arr[str] = arr[str + 1];

}

i--;

std::cout << "Строка 'удалена'\n";

}

if (numb == 66) {

std::cout << "Выполнить приказ 66!\n";

for (int j = 0; j < i; j++) {

arr[j] = order66;

}

i = 0;

std::cout << "Все строки были уничтожены!\n";

}

}

else {

cout << "Нечего удалять\n";

}

cout << "Что дальше?(1-продолжить,0-выйти)\n";

cin >> yon;

while (yon != 0 && yon != 1) {

cout << "Вы ввели неверное значение. Повторите\n";

cin >> yon;

}

return yon;

}

int output() {

int yon = 0;

if (i > 0) {

int numb;

cout << "Введите номер строки, которую вы хотите вывести\n";

cin >> numb;

numb--;

cout << "Элемент " << numb + 1 << endl;

cout << "Пасспортные данные:\n" << arr[numb].passport;

cout << "\nДата приезда:\n" << arr[numb].yearp << ":" << arr[numb].monthp << ":" << arr[numb].dayp;

cout << "\nДата отъезда:\n" << arr[numb].yearo << ":" << arr[numb].montho << ":" << arr[numb].dayo;

cout << "\nНомер в гостинице:\n" << arr[numb].number;

cout << "\nТип размещения:\n";

switch (arr[numb].place) {

case lux:

cout << "Люкс\n";

break;

case one:

cout << "Одноместный\n";

break;

case two:

cout << "Двухместный\n";

break;

case three:

cout << "Трёхместный\n";

break;

case apartments:

cout << "Апартаменты\n";

break;

default:

cout << "Были введены неверные данные\n";

break;

}

}

else {

cout << "Нечего выводить\n";

}

cout << "Что дальше?(1-продолжить,0-выйти)\n";

cin >> yon;

while (yon != 0 && yon != 1) {

cout << "Вы ввели неверное значение. Повторите\n";

cin >> yon;

}

return yon;

}

int search() {

int yon = 0, temp = 0;

if (i > 0) {

unsigned int a, b, c;

cout << "Введите дату прибытия(Год, Месяц, День)\n";

cin >> a >> b >> c;

for (int j = 0; j < i; j++) {

if (a == arr[j].yearp && b == arr[j].monthp && c == arr[j].dayp) {

cout << "Есть совпадения!\n";

cout << "Элемент " << j + 1 << endl;

cout << "Пасспортные данные:\n" << arr[j].passport;

cout << "\nДата приезда:\n" << arr[j].yearp << ":" << arr[j].monthp << ":" << arr[j].dayp;

cout << "\nДата отъезда:\n" << arr[j].yearo << ":" << arr[j].montho << ":" << arr[j].dayo;

cout << "\nНомер в гостинице:\n" << arr[j].number;

cout << "\nТип размещения:\n";

switch (arr[j].place) {

case lux:

cout << "Люкс\n";

break;

case one:

cout << "Одноместный\n";

break;

case two:

cout << "Двухместный\n";

break;

case three:

cout << "Трёхместный\n";

break;

case apartments:

cout << "Апартаменты\n";

break;

default:

cout << "Были введены неверные данные\n";

break;

}

temp++;

}

}

if (temp == 0) {

cout << "Совпадений нет!\n";

}

}

else {

cout << "Чтобы искать введите данные\n";

}

cout << "Что дальше?(1-продолжить,0-выйти)\n";

cin >> yon;

while (yon != 0 && yon != 1) {

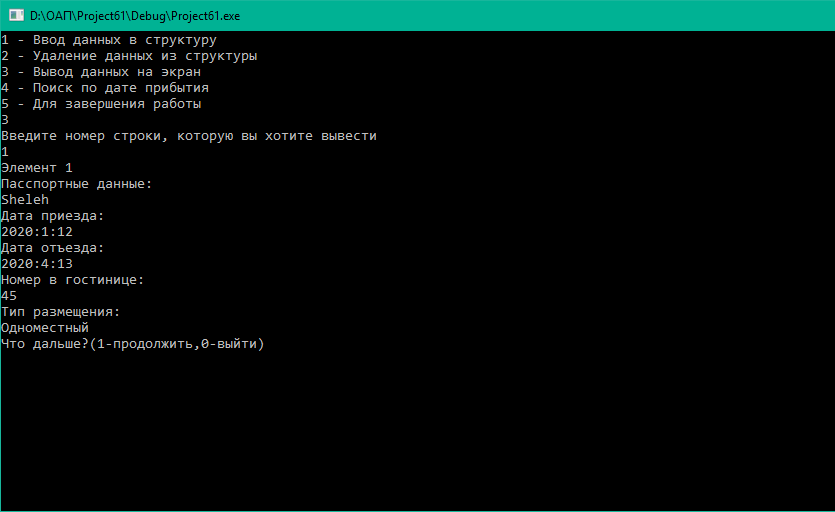
cout << "Вы ввели неверное значение. Повторите\n";

cin >> yon;

}

return yon;

}



**Список клиентов гостиницы.** Паспортные данные, даты приезда и отъезда, номер, тип размещения (люкс, одноместный, двухместный, трехместный, апартаменты). Поиск гостя по фамилии.

#include <iostream>

#define size 100

#define large 30

enum type {

lux = 1,

one,

two,

three,

apartments

} aparts;

union hot {

char passport[size];

int place = aparts;

int number;

char datap[size];

char datao[size];

} uni;

struct hotel {

char passport[size];

int place = aparts;

int number;

char datap[size];

char datao[size];

};

hotel arr[large];

int i = 0;

int input();

int output();

int search();

using namespace std;

bool func(const char fam[size], const char t[size]) {

int str1 = 0, str2 = 0;

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

str1 += (int)fam[i];

}

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

str2 += (int)t[i];

}

if (str1 == str2) {

return true;

}

else {

return false;

}

}

int main() {

setlocale(LC\_ALL, "ru");

int k = 0;

int yon = 0;

cout << "Что вы хотите сделать?\n";

do {

system("cls");

cout << "1 - Ввод данных в структуру\n";

cout << "2 - Вывод данных на экран\n";

cout << "3 - Поиск по фамилии\n";

cout << "4 - Для завершения работы\n";

cin >> k;

switch (k)

{

case 1:

yon = input();

break;

case 2:

yon = output();

break;

case 3:

yon = search();

break;

default:

yon = 0;

cout << "Рад помочь вам!\n";

break;

}

} while (yon == 1);

return 0;

}

int input() {

int yon = 0;

cout << "Ввод данных в структуру\n";

if (i < large) {

cout << "Введите пасспортные данные: Фамилия\n";

cin >> uni.passport;

strcpy\_s(arr[i].passport, uni.passport);

cout << "Введите дату приезда(год.месяц.день)\n";

cin >> uni.datap;

strcpy\_s(arr[i].datap, uni.datap);

cout << "Введите дату отъезда(год, месяц, день)\n";

cin >> uni.datao;

strcpy\_s(arr[i].datao, uni.datao);

cout << "Введите номер в гостинице\n";

cin >> uni.number;

arr[i].number = uni.number;

cout << "Введите тип размещения(1 - люкс, 2 - одноместный, 3 - двухместный, 4 - трёхместный, 5 - апартаменты)\n";

cin >> uni.place;

arr[i].place = uni.place;

i++;

}

else {

cout << "Число строк уже максимально\n";

}

cout << "Что дальше?(1-продолжить,0-выйти)\n";

cin >> yon;

while (yon != 0 && yon != 1) {

cout << "Вы ввели неверное значение. Повторите\n";

cin >> yon;

}

return yon;

}

int output() {

int yon = 0;

if (i > 0) {

int numb;

cout << "Введите номер строки, которую вы хотите вывести\n";

cin >> numb;

numb--;

cout << "Элeмент " << numb + 1 << endl;

cout << "Пасспортные данные:\n" << arr[numb].passport;

cout << "\nДата приезда:\n" << arr[numb].datap;

cout << "\nДата отъезда:\n" << arr[numb].datao;

cout << "\nНомер в гостинице:\n" << arr[numb].number;

cout << "\nТип размещения:\n";

switch (arr[numb].place) {

case lux:

cout << "Люкс\n";

break;

case one:

cout << "Одноместный\n";

break;

case two:

cout << "Двухместный\n";

break;

case three:

cout << "Трёхместный\n";

break;

case apartments:

cout << "Апартаменты\n";

break;

default:

cout << "Были введены неверные данные\n";

break;

}

}

else {

cout << "Нечего выводить\n";

}

cout << "Что дальше?(1-продолжить,0-выйти)\n";

cin >> yon;

while (yon != 0 && yon != 1) {

cout << "Вы ввели неверное значение. Повторите\n";

cin >> yon;

}

return yon;

}

int search() {

int yon = 0, temp = 0;

if (i > 0) {

char fam[size] = "", t[size] = "";

cout << "Введите фамилию\n";

cin >> fam;

for (int j = 0; j < i; j++) {

strcpy\_s(t, arr[j].passport);

if (func(fam, t)) {

cout << "Есть совпадения!\n";

cout << "Элемент " << j + 1 << endl;

cout << "Пасспортные данные:\n" << arr[j].passport;

cout << "\nДата приезда:\n" << arr[j].datap;

cout << "\nДата отъезда:\n" << arr[j].datao;

cout << "\nНомер в гостинице:\n" << arr[j].number;

cout << "\nТип размещения:\n";

switch (arr[j].place) {

case lux:

cout << "Люкс\n";

break;

case one:

cout << "Одноместный\n";

break;

case two:

cout << "Двухместный\n";

break;

case three:

cout << "Трёхместный\n";

break;

case apartments:

cout << "Апартаменты\n";

break;

default:

cout << "Были введены неверные данные\n";

break;

}

temp++;

}

}

if (temp == 0) {

cout << "Совпадений нет!\n";

}

}

else {

cout << "Чтобы искать введите данные\n";

}

cout << "Что дальше?(1-продолжить,0-выйти)\n";

cin >> yon;

while (yon != 0 && yon != 1) {

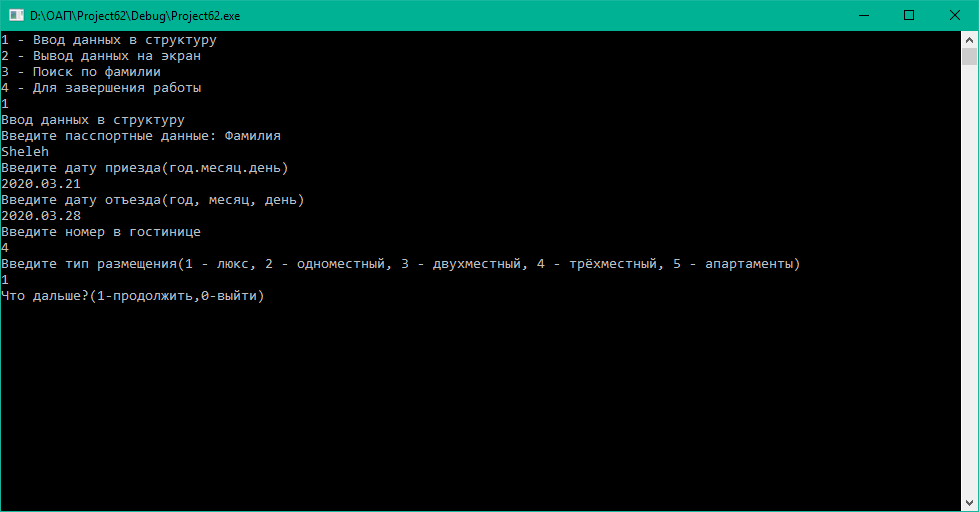
cout << "Вы ввели неверное значение. Повторите\n";

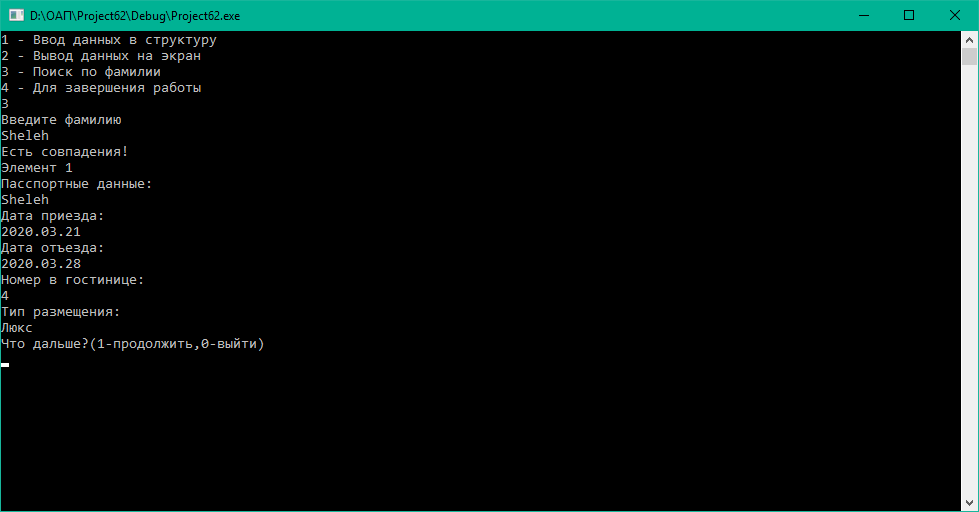
cin >> yon;

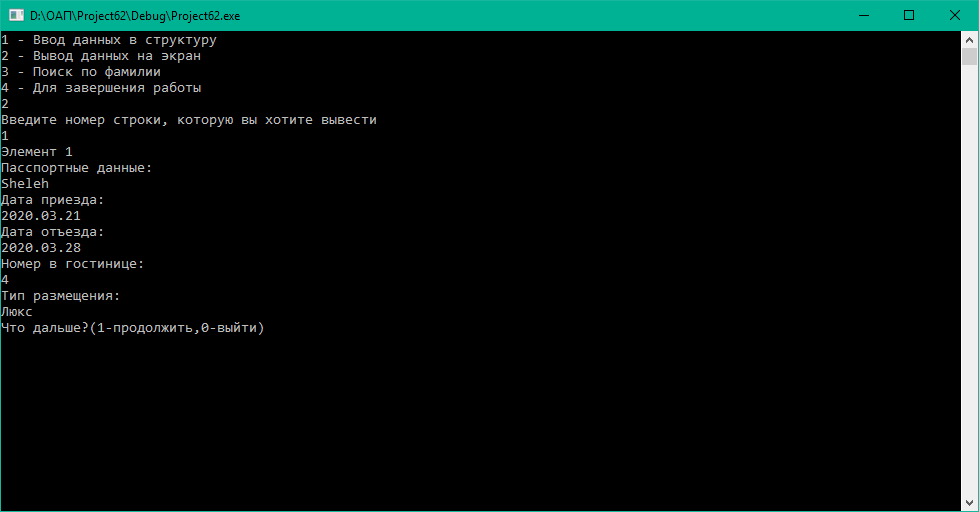
}

return yon;

}



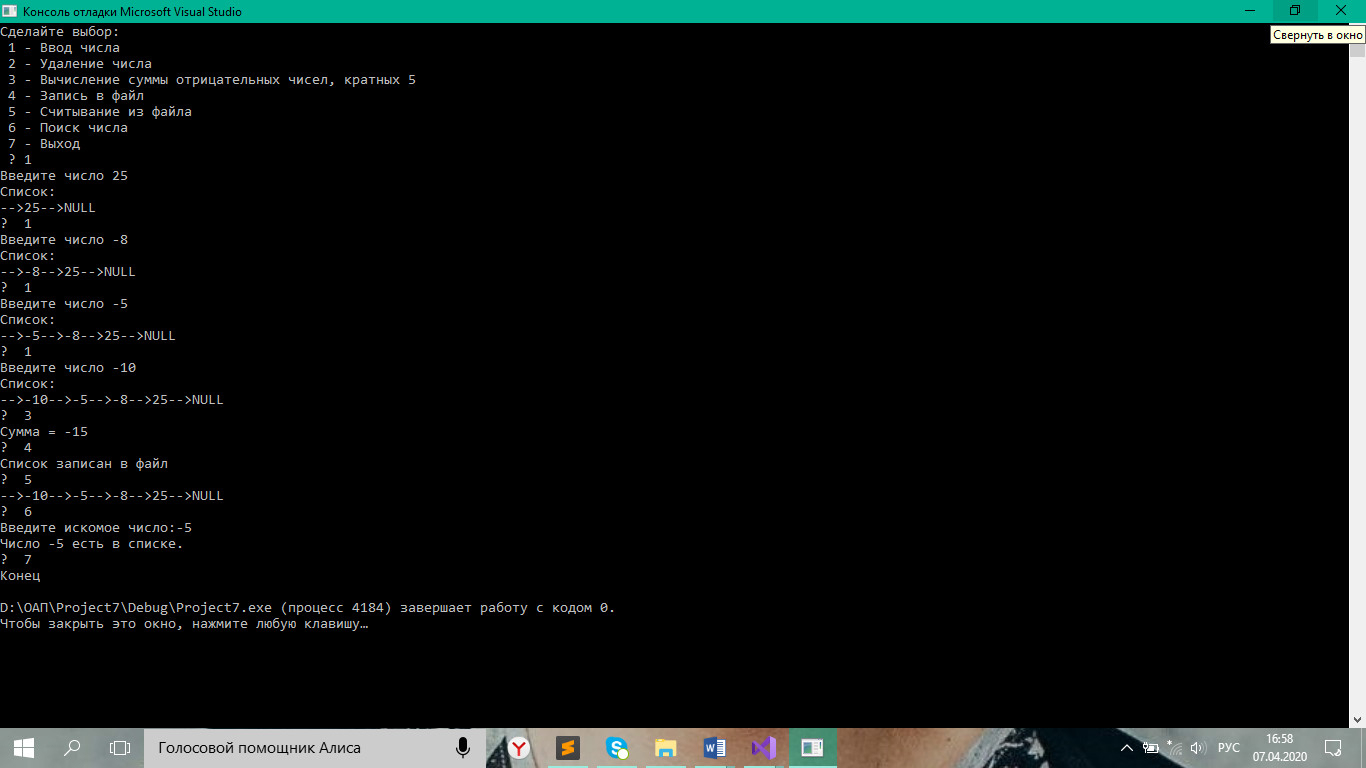




Лабораторная работа № 7

Динамические структуры данных. Односвязные списки

Создать список, содержащий элементы целого типа. Найти сумму отрицательных элементов, кратных 5, или выдать сообщение, что таких элементов нет.

#include <iostream>

#include <fstream>

#include<windows.h>

using namespace std;

struct list

{

int number;

list\* next;

};

void insert(list\*&,int);

float del(list\*&, int);

int IsEmpty(list\*);

void printList(list\*);

void menu(void);

void sum(list\*);

void file\_inf(list\*& pfirst);

void file\_outf(list\*& pfirst);

int poisk(list\*& pfirst, int num);

int main()

{

setlocale(LC\_CTYPE, "Russian");

list\* first = NULL;

int choice;

float value;

menu();

cout << " ? ";

cin >> choice;

while (choice != 7)

{

switch (choice)

{

case 1: cout << "Введите число ";

cin >> value;

insert(first, value);

printList(first);

break;

case 2: if (!IsEmpty(first))

{

cout << "Введите удаляемое число ";

cin >> value;

if (del(first, value))

{

cout << "Удалено число " << value << endl;

printList(first);

}

else

cout << "Число не найдено" << endl;

}

else

cout << "Список пуст" << endl;

break;

case 3: sum(first);

break;

case 4: file\_inf(first);

break;

case 5: file\_outf(first);

break;

case 6: cout << "Введите искомое число:";

cin >> value;

if (poisk(first, value))

{

cout << "Число " << value << " есть в списке." << endl;

}

else

cout << "Число не найдено" << endl;

break;

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

menu();

break;

}

cout << "? ";

cin >> choice;

}

cout << "Конец" << endl;

return 0;

}

void menu(void)

{

cout << "Сделайте выбор:" << endl;

cout << " 1 - Ввод числа" << endl;

cout << " 2 - Удаление числа" << endl;

cout << " 3 - Вычисление суммы отрицательных чисел, кратных 5" << endl;

cout << " 4 - Запись в файл" << endl;

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

cout << " 6 - Поиск числа" << endl;

cout << " 7 - Выход" << endl;

}

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;

}

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

{

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

if (value == p->number)

{

temp = p;

p = p->next;

delete temp;

return value;

}

else

{

previous = p;

current = p->next;

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

{

previous = current;

current = current->next;

}

if (current != NULL)

{

temp = current;

previous->next = current->next;

delete(temp);

return value;

}

}

return 0;

}

int IsEmpty(list\* p)

{

return p == NULL;

}

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 file\_inf(list\*& pfirst)

{

list\* temp = pfirst;

list buf;

ofstream outf("FILE.txt");

if (outf.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

while (temp)

{

buf = \*temp;

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

temp = temp->next;

}

outf.close();

cout << "Список записан в файл\n";

}

void file\_outf(list\*& pfirst)

{

list buf, \* first = nullptr;

ifstream fin("FILE.txt");

if (fin.fail())

{

cout << "\n Ошибка открытия файла";

exit(1);

}

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

while (!fin.eof())

{

insert(first, buf.number);

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

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

}

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

fin.close();

pfirst = first;

}

int poisk(list\*& pfirst, int value)

{

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

if (value == pfirst->number)

{

return value;

}

else

{

previous = pfirst;

current = pfirst->next;

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

{

previous = current;

current = current->next;

}

if (current != NULL)

{

temp = current;

previous->next = current->next;

delete(temp);

return value;

}

}

return 0;

}

void sum(list\* p)

{

int sm = 0;

if (p == NULL)

cout << "Список пуст" << endl;

else

{

while (p != NULL)

{

if (p->number%5==0)

if(p->number < 0)

sm = sm + (p->number);

p = p->next;

}

cout << "Сумма = " << sm << endl;

}

}

Лабораторная работа № 8

Полустатические структуры данных: стеки

Разработать функцию, которая удаляет первый положительный элемент, если такой есть.

**Главная функция:**

#include <iostream>

#include "myStack.h"

using namespace std;

int main()

{

setlocale(0, "");

int choice;

Stack \*myStk = new Stack; //выделение памяти для стека

myStk->head = NULL; //инициализация первого элемента

for (;;)

{

cout << "Выберите команду:" << endl;

cout << "1 - Добавление элемента в стек" << endl;

cout << "2 - Извлечение элемента из стека" << endl;

cout << "3 - Вывод стека" << endl;

cout << "4 - Очистка стека" << endl;

cout << "5 - Моя функция" << endl;

cout << "6 - Выход" << endl;

cin >> choice;

switch (choice)

{

case 1: cout << "Введите элемент: " << endl;

cin >> choice;

push(choice, myStk);

break;

case 2: choice = pop(myStk);

if (choice != -1)

cout << "Извлеченный элемент: " << choice << endl;

break;

case 3: cout << "Весь стек: " << endl;

show(myStk);

break;

case 4:clear(myStk);

break;

case 5: MyFunction(myStk);

break;

case 6: return 0;

break;

}

}

return 0;

}

**Функции для манипуляции:**

#include <iostream>

#include "myStack.h"

using namespace std;

void push(int x, Stack \*myStk) //Добавление элемента х в стек

{

Stack\* e = new Stack; //выделение памяти для нового элемента

e->data = x; //запись элемента x в поле v

e->next = myStk->head; //перенос вершины на следующий элемент

myStk->head = e; //сдвиг вершины на позицию вперед

}

int pop(Stack \*myStk) //Извлечение (удаление) элемента из стека

{

if (myStk->head == NULL)

{

cout << "Стек пуст!" << endl;

return -1; //если стек пуст - возврат -1

}

else

{

Stack \*e = myStk->head; //е - переменная для хранения адреса элемента

int a = myStk->head->data; //запись числа из поля data в переменную a

myStk->head = myStk->head->next; //перенос вершины

delete e; //удаление временной переменной

return a; //возврат значения удаляемого элемента

}

}

void show(Stack \*myStk) //Вывод стека

{

Stack\* e = myStk->head; //объявляется указатель на вершину стека

int a;

if (e == NULL)

cout << "Стек пуст!" << endl;

while (e != NULL)

{

a = e->data; //запись значения в переменную a

cout << a << " ";

e = e->next;

}

cout << endl;

}

int MyFunction(Stack \*myStk) //Моя функция

{

if (myStk->head == NULL)

{

cout << "Стек пуст!";

return -1; //если стек пуст - возврат -1

}

else

{

const int N = 50;

Stack\*e = myStk->head;

int arr[N], i = 0, f=1;

while (e != NULL)

{

int a = e->data;

arr[i] = a;

e = e->next;

i++;

}

while (myStk->head != NULL)

{

Stack \*e = myStk->head; //е-переменная для хранения адреса элемента

int a = myStk->head->data; //запись числа из поля data в переменную a

myStk->head = myStk->head->next; //перенос вершины

delete e; //удаление временной переменной

}

for (int j = i-1; j>=0;)

{

if (arr[j] > 0 && f==1)

{

j--;

f = 0;

}

else

{

Stack\* e = new Stack;

e->data = arr[j]; //запись элемента arr[j] в поле v

e->next = myStk->head; //перенос вершины на следующий элемент

myStk->head = e; //сдвиг вершины на позицию вперед

j--;

}

}

}

}

int clear(Stack\*myStk) {

if (myStk->head == NULL)

{

cout << "Стек пуст!";

return -1; //если стек пуст - возврат -1

}

else

{

while(myStk->head != NULL)

{

Stack \*e = myStk->head; //е - переменная для хранения адреса элемента

int a = myStk->head->data; //запись числа из поля data в переменную a

myStk->head = myStk->head->next; //перенос вершины

delete e; //удаление временной переменной

}

cout << "Стек полностью очищен!" << endl;

}

}

**Заголовочный файл:**

struct Stack

{

int data; //информационный элемент

Stack \*head; //вершина стека

Stack \*next; //указатель на следующий элемент

};

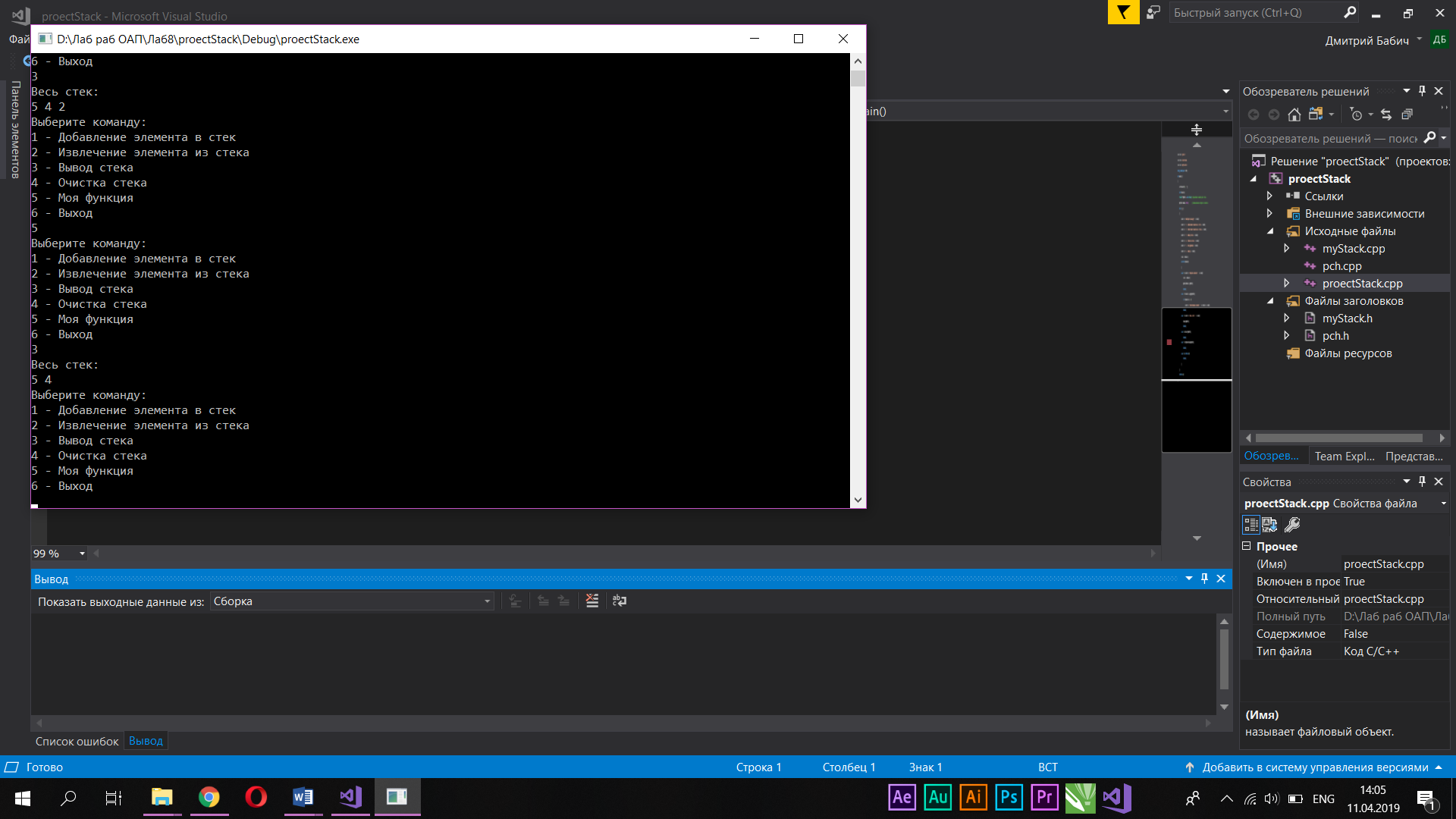
void show(Stack \*myStk); //прототип

int pop(Stack \*myStk); //прототип

void push(int x, Stack \*myStk); //прототип

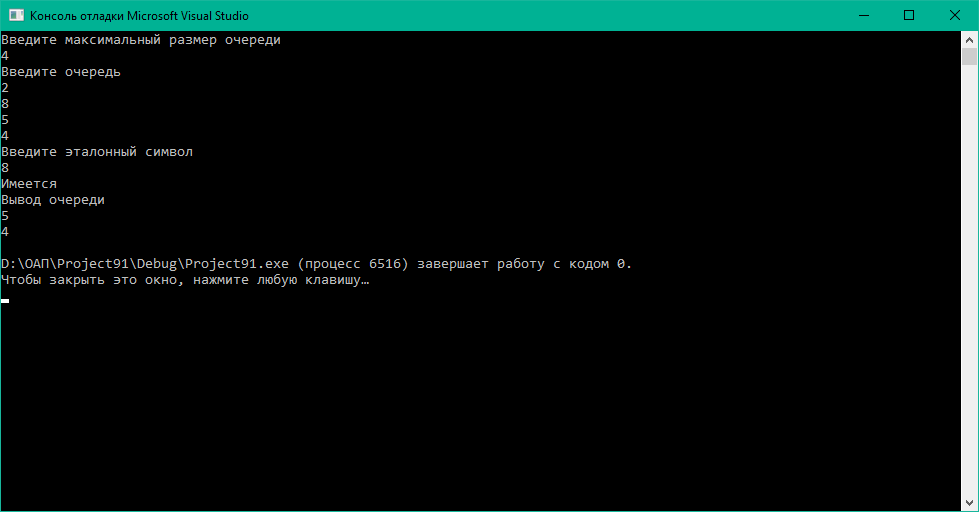
int clear(Stack \*myStk); //прототип очистки

int MyFunction(Stack \*myStk); //прототип



Лабораторная работа № 9

Полустатические структуры данных: очереди

Создать очередь для символов и функции для ввода, вывода и удаления элементов очереди. Ввести символы с клавиатуры. Ввести эталонный символ и проверить, имеется ли он в очереди. При его наличии в очереди, удалить все элементы до него.

#include <iostream>

using namespace std;

struct queue {

char data;

struct queue\* next;

} \*p, \* head, \* tail;

char datainside;

queue\* tmp;

void vvod() {

p = new queue;

p->data = datainside;

p->next = NULL;

if (head == NULL) {

head = p;

}

else {

tail->next = p;

}

tail = p;

}

void vivod() {

while (head != NULL) {

cout << head->data;

cout << "\n";

head = head->next;

}

}

void ydalenie(queue\* a) {

while (head->next != NULL) {

a = head;

head = head->next;

delete a;

}

}

int main(int argc, char\* argv[]) {

setlocale(0, "");

head = NULL;

tail = NULL;

cout << "Введите максимальный размер очереди\n";

int max;

cin >> max;

cout << "Введите очередь\n";

for (int i = 0; i < max; i++) {

cin >> datainside;

vvod();

}

//Ввести эталонный символ и проверить, имеется ли он в очереди.

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cout << "Введите эталонный символ\n";

char etalon;

cin >> etalon;

tmp = head;

while (tmp != NULL) {

if (tmp->data == etalon) {

cout << "Имеется\n";

head = tmp->next;

break;

}

tmp = tmp->next;

}

cout << "Вывод очереди\n";

vivod();

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

Лабораторная работа №10

Двусвязные списки

**returnN** – функция возвращения истинного значения, если список пуст и возвращение ложного в противном случае.

#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;

};

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

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

{

Address\* p = \*plast;

if (\*plast == NULL)

{

e->next = NULL;

e->prev = NULL;

\*plast = e;

\*phead = e;

return;

}

else

{

p->next = e;

e->next = NULL;

e->prev = p;

\*plast = e;

}

}

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

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

{

Address\* temp = new Address();

if (!temp)

{

cout << "Ошибка выделения памяти";

return NULL;

}

cout << "Введите имя: ";

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

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 = NULL;

temp->prev = NULL;

return temp;

}

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

void outputList(Address\*\* phead, Address\*\* plast) //Вывод списка на экран

{

Address\* t = \*phead;

while (t)

{

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

t = t->next;

}

cout << "" << endl;

}

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

void find(char name[NAME\_SIZE], Address\*\* phead) // Поиск имени в списке

{

Address\* t = \*phead;

while (t)

{

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

t = t->next;

}

if (!t)

cout << "Имя не найдено" << endl;

else

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

}

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

void delet(char name[NAME\_SIZE], Address\*\* phead, Address\*\* plast) { //Удаление элемента из списка

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 = NULL;

}

else {

\*plast = NULL;

}

}

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) //Запись в файл

{

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) //Считывание из файла

{

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 = NULL;

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 returnN(Address\*\* phead, Address\*\* plast)

{

if (phead == NULL) cout << "Список пуст" << endl; else cout << "Список не пуст" << endl;

}

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

int main()

{

Address\* head = NULL;

Address\* last = NULL;

setlocale(LC\_CTYPE, "Rus");

int choose;

do {

cout << "Введите: " << endl;

cout << "1 - Добавление в конец списка" << endl;

cout << "2 - Удаление элемента из списка" << endl;

cout << "3 - Вывод списка" << endl;

cout << "4 - Поиск элемента" << endl;

cout << "5 - Запись в файл" << endl;

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

cout << "7 - Моя функция" << endl;

cout << "8 - Выход" << endl;

cin >> choose;

switch (choose)

{

case 1: {

insert(setElement(), &head, &last);

break;

}

case 2: {

char dname[NAME\_SIZE];

cout << "Введите имя: ";

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

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

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

cin.sync();

delet(dname, &head, &last);

break;

}

case 3: {

outputList(&head, &last);

break;

}

case 4: {

char fname[NAME\_SIZE];

cout << "Введите имя: ";

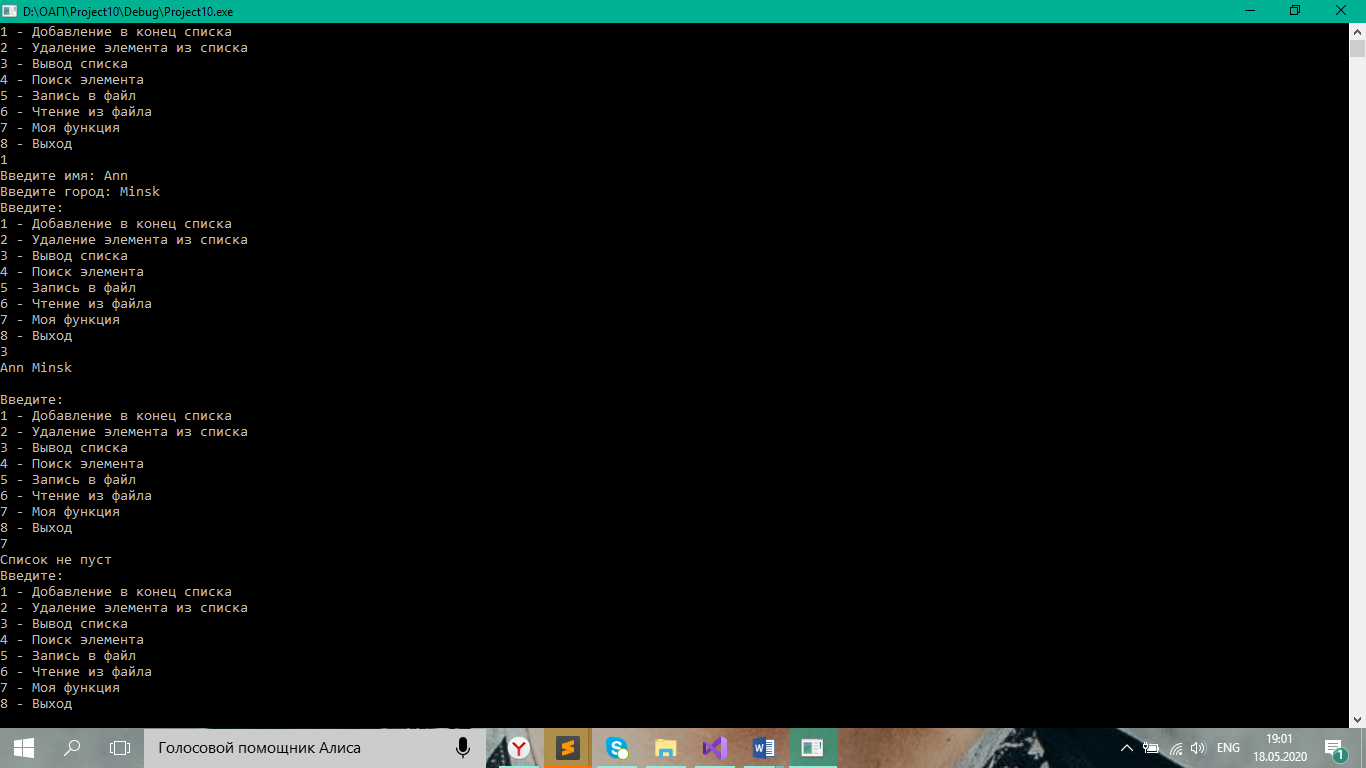
cin.getline(fname, NAME\_SIZE - 1, '\n');

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

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

cin.sync();

find(fname, &head);

 break;

}

case 5: {

writeToFile(&head);

break;

}

case 6: {

readFromFile(&head, &last);

break;

}

case 7: {

returnN(&head, &last);

break;

}

case 8: {

break;

}

default:

cout << "Некорректный ввод" << endl;

break;

}

} while (choose != 8);

}

Лабораторная работа №11

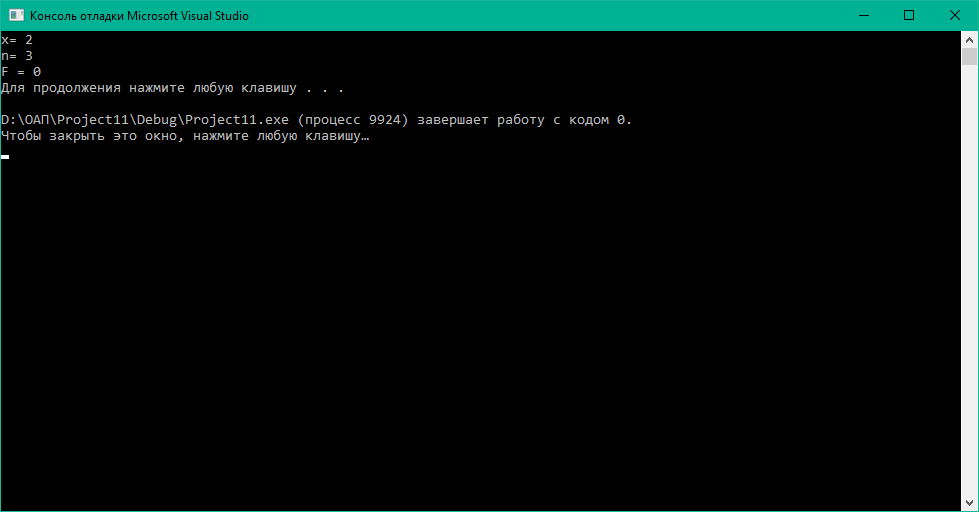
Рекурсивные алгоритмы

Разработать программу, реализующую рекурсивную функцию **f(x, n)**, вычисляющую величину **xn/n!** при любом вещественном **x** и любом неотрицательном целом **n**.

Для вычисления **xn-1/(n-1)!** надо рекурсивно обратиться к **f(x, n-1),** а затем полученную величину умножить на **x/n**, чтобы получить значение **f(x, n)**.

Функция **f(x, n)** принимает следующий вид:



#include <iostream>

using namespace std;

int i = 0;

float Func(int x, int n)

{

int F;

if (n == 0)

return 1;

else

if (n == 1)

return x;

else

{

F=(x \* x / (n \* (n - 1))) \* Func(x, (n - 2));

return F;

}

}

void main()

{

setlocale(0, "");

int x, n;

cout << "x= "; cin >> x;

cout << "n= "; cin >> n;

cout << "F = " << Func(x,n) << endl;

system("pause");

}

Вариант 16

#include <iostream>

using namespace std;

float Func(int n, int m)

{

int F;

if (n == 0)

return 1;

else

if (n == m)

return -1;

else

{

F = 2\*Func(n-1, m);

return F;

}

}

void main()

{

setlocale(0, "");

int n, m;

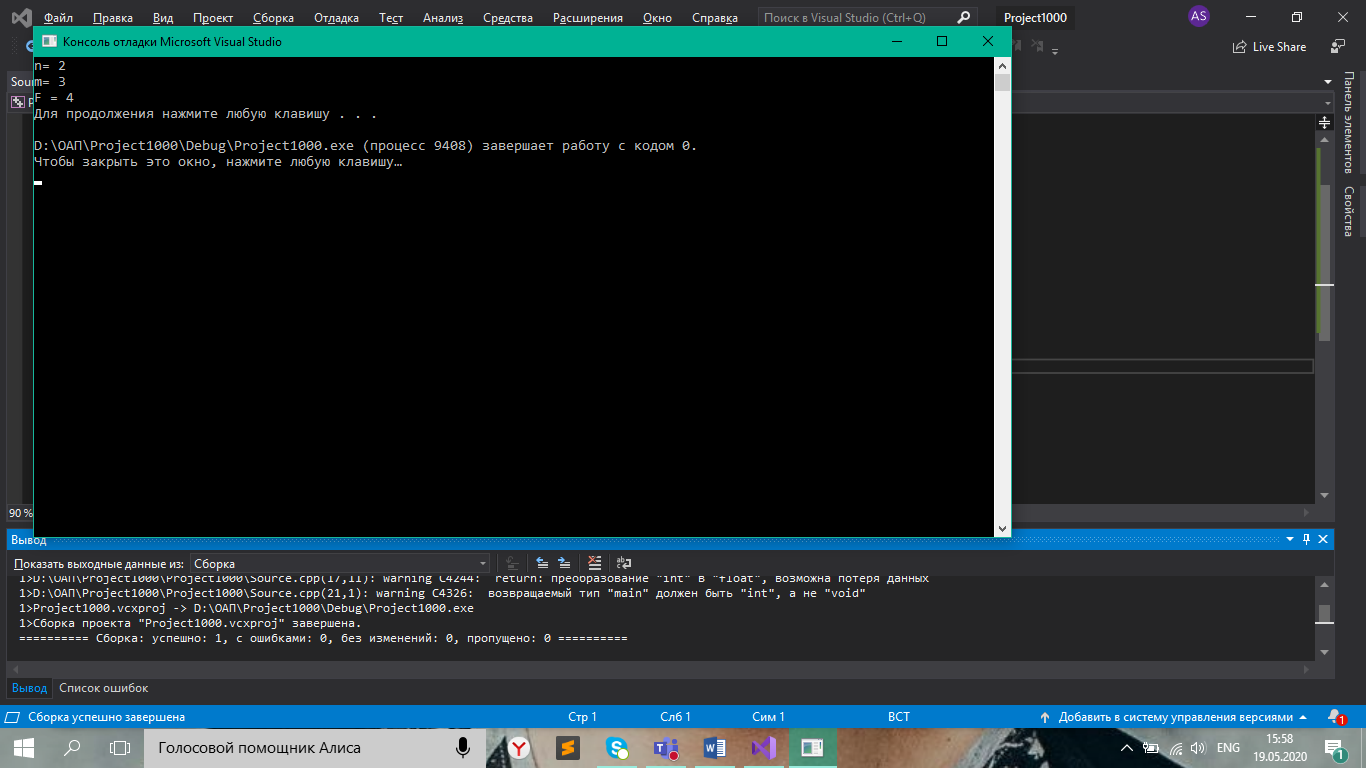
cout << "n= "; cin >> n;

cout << "m= "; cin >> m;

cout << "F = " << Func(n, m) << endl;

system("pause");

}



Лабораторная работа №12

Бинарные деревья

Вершина бинарного дерева содержит ключ и два указателя на потомков. Написать функцию вычисления среднего арифметического всех элементов дерева.

#include <iostream>

#include <cstring>

#include <algorithm>

#include <string>

using namespace std;

struct Tree

{

int key;

string str;

struct Tree\* left;

struct Tree\* right;

};

Tree\* root = NULL;

Tree\* CreateTree(Tree\* root);

Tree\* CreateEl(int k, string str);

Tree\* AddEl(Tree\* root);

Tree\* DeleteEl(Tree\* root, int key);

void PrintTree(Tree\* root, int depth);

void PrintTree2(Tree\* root, int depth);

void DeleteTree(Tree\* root);

int arsum(Tree\* root, int depth, int key);

int isBalanced(Tree\* root);

int height(Tree\* node);

int maxInt(int a, int b);

int main()

{

setlocale(LC\_ALL, "rus");

int choice = 0;

do

{

cout << "1 - добавление нового элемента\n";

cout << "2 - удаление элемента\n";

cout << "3 - вывод дерева в нисходящем порядке\n";

cout << "4 - вывод дерева в смешанном порядке\n";

cout << "5 - среднее арифметическое\n";

cout << "6 - проверка на сбалансированность\n";

cout << "7 - очистка дерева\n";

cout << "0 - выход\n";

cout << "\nНомер операции > "; cin >> choice;

switch (choice)

{

case 1: {

if (root == NULL)

root = CreateTree(root);

else AddEl(root);

break;

}

case 2: {

int key;

cout << "Ключ > "; cin >> key;

DeleteEl(root, key);

break;

}

case 3: {

PrintTree(root, 0);

cout << endl << endl;

break;

}

case 4: {

PrintTree2(root, 0);

cout << endl << endl;

break;

}

case 6: {

int r = isBalanced(root->right), l = isBalanced(root->left);

if (isBalanced(root->right) == 1 || isBalanced(root->left) == 1)

cout << "\nДерево сбалансировано!\n";

else

cout << "\nДерево не сбалансировано!\n";

}

case 7: {

DeleteTree(root);

cout << endl << endl;

root = NULL;

break;

}

case 5: {

cout << "Среднее арифметическое > " << arsum(root, 0, root->key) / 5 << endl;

break;

}

case 0: {

break;

}

}

} while (choice != 0);

}

Tree\* CreateTree(Tree\* root)

{

int key;

string str;

if (root == NULL)

{

cout << "Ключ корня > "; cin >> key;

cout << "Строка корня > "; cin >> str;

root = CreateEl(key, str);

cout << "\nКорень дерева создан!\n\n";

}

return root;

}

Tree\* CreateEl(int k, string str)

{

Tree\* interm = new Tree[sizeof(Tree)];

interm->key = k;

interm->str = str;

interm->left = interm->right = NULL;

return interm;

}

Tree\* AddEl(Tree\* root)

{

int key; string str;

cout << "Ключ элемента > "; cin >> key;

if (key == root->key)

{

cout << "Ключ должнен быть уникальным, измените значение!\n";

cout << "Ключ элемента > "; cin >> key;

}

cout << "Строка элемента > "; cin >> str; cout << endl << "Элемент добавлен!\n";

Tree\* prev = root; // из-за ошибки "Потенциально неинициализируемая переменная"

while (root)

{

prev = root;

if (key < root->key)

root = root->left;

else root = root->right;

}

root = CreateEl(key, str);

if (key < prev->key)

prev->left = root;

else

prev->right = root;

return root;

}

Tree\* DeleteEl(Tree\* root, int key)

{

Tree\* d, \* prevd, \* r, \* prevr;

d = root; prevd = NULL;

while (d != NULL && d->key != key)

{

prevd = d;

if (d->key > key)

d = d->left;

else

d = d->right;

}

if (d == NULL)

{

cout << "Элемент не найден!";

return root;

}

if (d->right == NULL)

r = d->left;

else if (d->left == NULL)

r = d->right;

else {

prevr = d;

r = d->left;

while (r->right != NULL)

{

prevr = r;

r = r->right;

}

if (prevr == d)

r->right = d->right;

else {

r->right = d->right;

prevr->right = r->left;

r->left = prevr;

}

}

if (d == root)

root = r;

else {

if (d->key < prevd->key)

prevd->left = r;

else

prevd->right = r;

}

int Dkey = d->key;

cout << "Удален элемент с ключом " << Dkey << "\n";

d = NULL;

return root;

}

void PrintTree(Tree\* root, int depth)

{

if (root)

{

// смешанный обход - слева направи

PrintTree(root->right, depth);

cout << root->key << ' ';

PrintTree(root->left, depth);

}

}

void PrintTree2(Tree\* root, int depth)

{

if (root)

{

// нисходящий обход - сверху вниз

PrintTree2(root->left, depth);

PrintTree2(root->right, depth);

cout << root->key << ' ';

}

}

void DeleteTree(Tree\* root)

{

if (root != NULL)

{

DeleteTree(root->left);

DeleteTree(root->right);

root = NULL;

}

}

int arsum(Tree\* root, int depth, int key)

{

int sum = 0;

if (root != NULL) {

sum += root->key + arsum(root->left, depth, key) + arsum(root->right, depth, key);

++depth;

}

return sum;

}

int maxInt(int a, int b)

{

return (a >= b) ? a : b;

}

int height(Tree\* node)

{

if (node == NULL)

return 0;

return 1 + maxInt(height(node->left), height(node->right));

}

int isBalanced(Tree\* root)

{

int lh;

int rh;

if (root == NULL)

return 1;

lh = height(root->left);

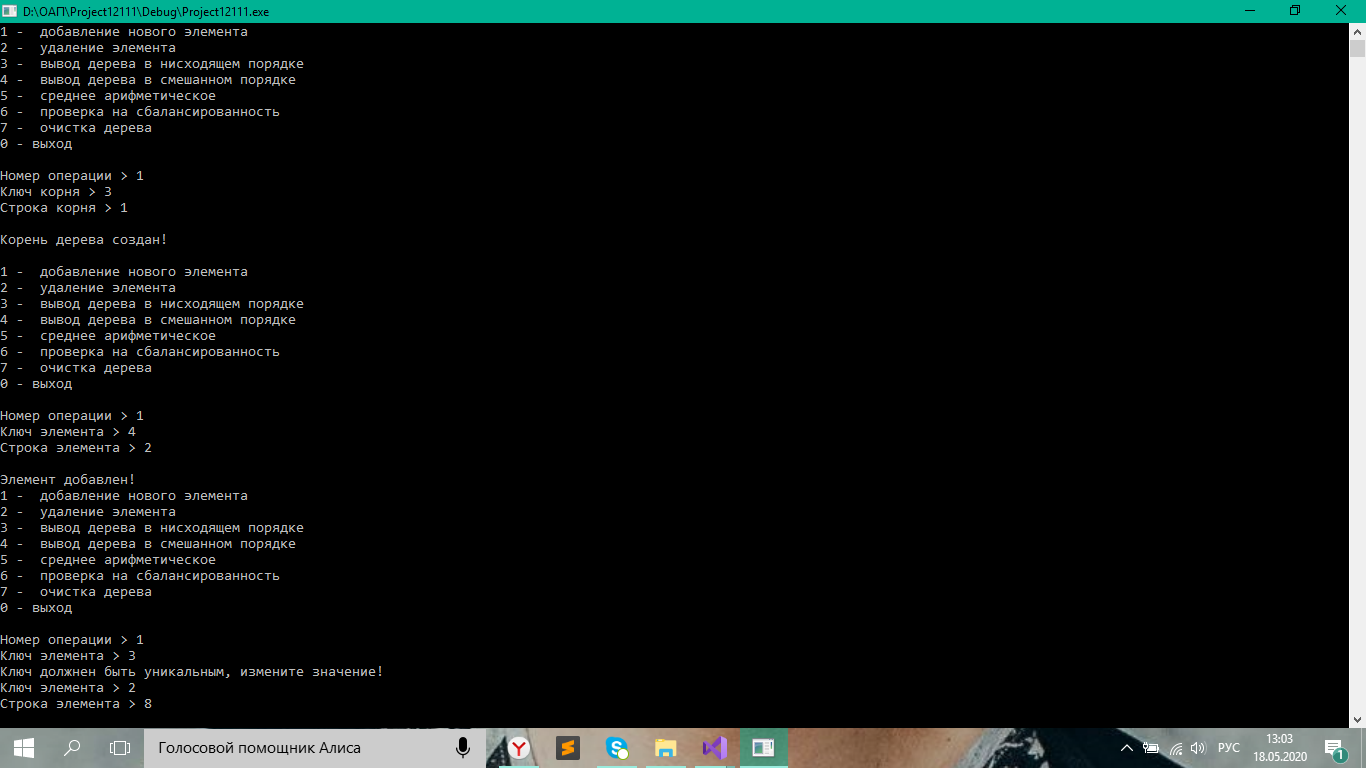
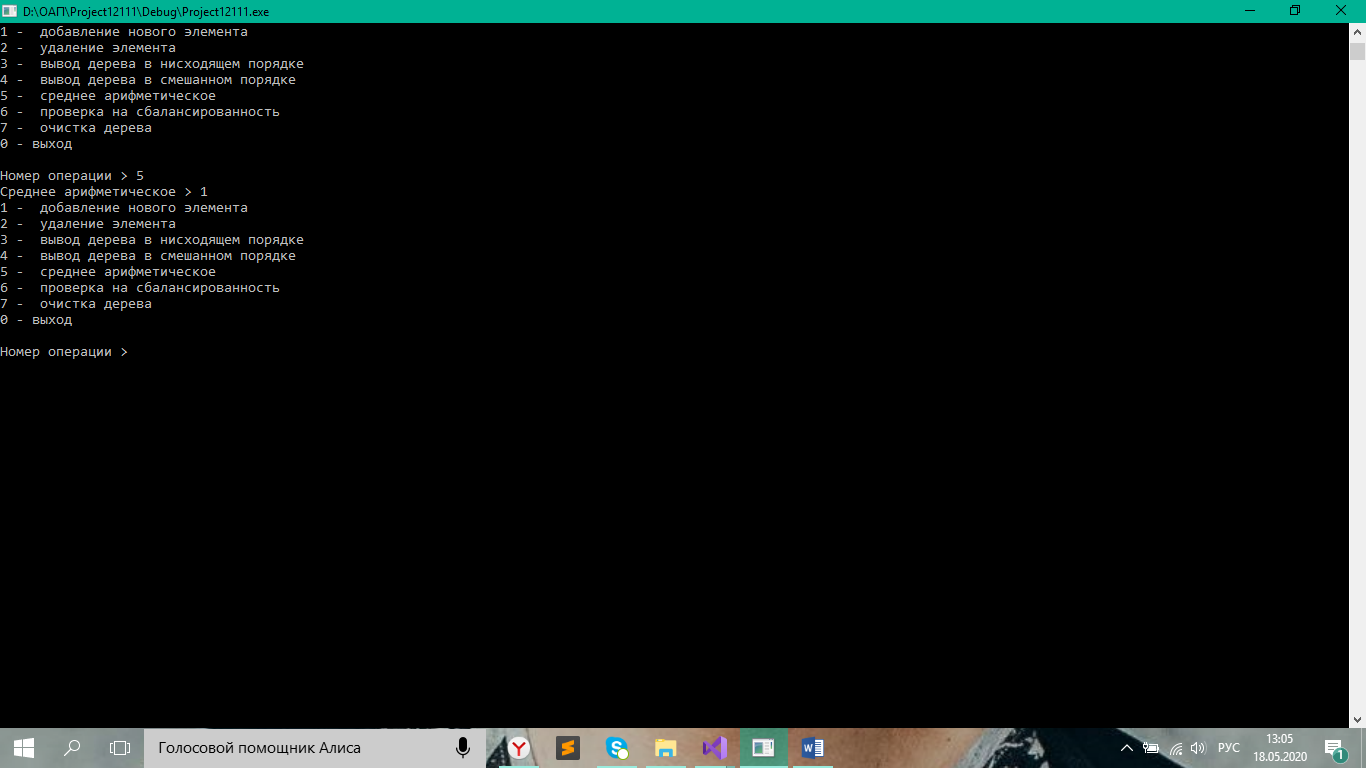
rh = height(root->right);

if (abs(lh - rh) == 0 && isBalanced(root->left) && isBalanced(root->right))

return 1;

return 0;

}



**Лабораторная работа №13**

**Бинарные кучи**

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

**Файл заголовка Heap.h**

#pragma once

struct AAA

{

int x; //ключ

void print();

int getPriority() const;

};

namespace heap //создание пространство имен

{

enum CMP //перечисление

{

LESS = -1, EQUAL = 0, GREAT = 1 //обозначение для сравнения

};

struct Heap

{

int size; //размер кучи

int maxSize; //максимальный размер кучи

void\*\* storage; // данные

CMP(\*compare)(void\*, void\*); //сравнение функций

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

{

size = 0;

storage = new void\* [maxSize = maxsize]; //выделение динамической памяти

compare = f;

};

int left(int ix); //левый дочерний элемент

int right(int ix); //правый дочерний элемент

int parent(int ix); //родитель

bool isFull() const //проверка на переполнение

{

return (size >= maxSize); //возвращаем результат сравнения текущего размера с максимальным

};

bool isEmpty() const //проверка кучи на пустоту

{

return (size <= 0); //возвращаем результат сравнения текущего размера с нулем

};

bool isLess(void\* x1, void\* x2) const //функция сравнения является ли элемент х1 меньше элемента х2

{

return compare(x1, x2) == LESS; //вызов функции для элементов х1 и х2 и возврат сравнения результат функции с элементом перечисления LESS

};

bool isGreat(void\* x1, void\* x2) const //функция сравнения является ли элемент х1 больше элемента х2

{

return compare(x1, x2) == GREAT; //вызов функции для элементов х1 и х2 и возврат сравнения результат функции с элементом перечисления GREAT

};

bool isEqual(void\* x1, void\* x2) const //функция проверки равнества элементов х1 и х2

{

return compare(x1, x2) == EQUAL; //вызов функции для элементов х1 и х2 и возврат сравнения результат функции с элементом перечисления EQUAL

};

void swap(int i, int j); //обмен элементов

void heapify(int ix); //проверка элементов

void insert(void\* x); //добавление элементов

void\* extractMax(); //извлечение максимального

void extractMin(); //извлечение минимального

void extractI(); //удаление i-го элемента

void unionHeap(); //объединение куч

void scan(int i) const; //вывод на экран

};

Heap create(int maxsize, CMP(\*f)(void\*, void\*)); //создание кучи

};

**Файл для манипуляции Heap.cpp**

#include "Heap.h"

#include <iostream>

#include <iomanip>

void AAA::print()

{

std::cout << x;

}

int AAA::getPriority() const

{

return x;

}

namespace heap

{

Heap create(int maxsize, CMP(\*f)(void\*, void\*)) //создание кучи с заданным размером и функцией сравнения двух элементов

{

return \*(new Heap(maxsize, f)); //создаем новую кучу и возвращаем её(по указателю на выдел память получаем кучу и возвращаем ее)

}

int Heap::left(int ix) //определение номера левого потомка элемента ix

{

return (2 \* ix + 1 >= size) ? -1 : (2 \* ix + 1); //если номер левого потомка выходит за пределы текущего кол-ва элементов в куче, то возвращаем -1, иначе возвращаем номер девого элемента

}

int Heap::right(int ix) //определение номера правого потомка элемента ix

{

return (2 \* ix + 2 >= size) ? -1 : (2 \* ix + 2); //если номер правого потомка выходит за пределы текущего кол-ва элементов в куче, то возвращаем -1, иначе возвращаем номер правого элемента

}

int Heap::parent(int ix) //определение родительской вершины

{

return (ix + 1) / 2 - 1; //возврат номера родительского элемента

}

void Heap::swap(int i, int j) //перестановка двух элементов

{

void\* buf = storage[i]; //запоминаем i-тый элемент массива в буфере (storage-массив элементов произвольного типа)

storage[i] = storage[j]; //на место i-то элемента записываем j-тый

storage[j] = buf; //на место j-го записываем то, что сохранили в буфер

}

void Heap::heapify(int ix) //функция формирования кучи в массиве

{

int l = left(ix), r = right(ix), irl = ix; //переменная для хранения номера наибольшего из трех элементов

if (l > 0) //если есть левый потомок

{

if (isGreat(storage[l], storage[ix])) irl = l; //если левый больше вершины, вершина=левый

if (r > 0 && isGreat(storage[r], storage[irl])) irl = r; //если правый сущ и больше вершины, вершина=правый

if (irl != ix)

{

swap(ix, irl); //меняем местами

heapify(irl); //снова проверяем

}

}

}

void Heap::insert(void\* x) //добавление элемента

{

int i;

if (!isFull()) //наличие места в куче

{

storage[i = ++size - 1] = x;

while (i > 0 && isLess(storage[parent(i)], storage[i])) //пока не дойдем до вершины и пока родительский элемент меньше вставленного

{

swap(parent(i), i); //меняем элементы

i = parent(i); //новой позицией вставленного элемента становится позиция родительского элемента

}

}

}

void\* Heap::extractMax() //удаление максималього

{

void\* rc = nullptr; //устанавливаем указатель в начало

if (!isEmpty()) //если куча не пуста

{

rc = storage[0]; //присваиваем указатель первого элемента

storage[0] = storage[size - 1]; //верхним элементом становится последний

size--; //уменьшаем размер

heapify(0); //проверяем основное свойство кучи

} return rc;

}

void Heap::extractMin()

{

int min = 99, a;

int\*\* A = (int\*\*)storage;

void\* rc = nullptr;

if (!isEmpty())

{

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

{

if (\*A[i] < min)

{

min = \*A[i];

a = i; //запоминаем номер найденного минимального элемента

}

}

int i = parent(a);

for (int i = a; i < size - 1; i++)

{

storage[i] = storage[i + 1];

}

size--; //уменьшаем размер кода

heapify(0); //проверяем основное свойство кучи

}

}

void Heap::extractI() //функция удаления i-того элемента

{

int i;

std::cout << "Введите номер эллемента i: ";

std::cin >> i;

if (!isEmpty())

{

for (int j = i; j < size - 1; j++)

{

storage[j] = storage[j + 1];

}

size--; //уменьшаем размер кода

heapify(0); //проверяем основное свойство кучи

}

}

void Heap::unionHeap()

{

}

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(' ');

((AAA\*)storage[u])->print();

if (u == y) //если мы дошли до конца текущего уровня(у- номер последнего элемента в текущем уровне)

{

std::cout << '\n';

if (y == 0) //если текущий элемент нулевой

y = 2; //то номер последнего элемента в следующем первом

else

y += y \* 2; //иначе рассчитываем номер последнего элемента для следующего уровня

}

probel /= 2;

}

std::cout << '\n';

}

}**Главная функция** Source.cpp

#include "Heap.h"

#include <iostream>

using namespace std;

heap::CMP cmpAAA(void\* a1, void\* a2) //Функция сравнения 2х переменных типа ААА

{

#define A1 ((AAA\*)a1) //директива #define указывает компилятору, что нужно подставить вместо идентификатора замену

#define A2 ((AAA\*)a2)

heap::CMP //в зависимости от результата сравнения присваиваем значение -1/0/1

rc = heap::EQUAL;

if (A1->x > A2->x) //если значение первого элемента больше значения второго

rc = heap::GREAT; //записываем в переменную, что первый элемент большего второго

else

if (A2->x > A1->x) //если второй элемент больше первого

rc = heap::LESS; //записываем в переменную, что первый элемент меньше второго

return rc; //возвращаем значение переменной

#undef A2 //отменяем переменную А2

#undef A1

}

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

int main()

{

setlocale(LC\_ALL, "rus");

int k, choice;

heap::Heap h1 = heap::create(30, cmpAAA);

heap::Heap h2 = heap::create(30, cmpAAA);

heap::Heap h3 = heap::create(60, cmpAAA);

for (;;)

{

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

cout << "2 - добавить элемент в 1-ю кучу" << endl;

cout << "3 - вывод 2-й кучи на экран" << endl;

cout << "4 - добавить элемент в 2-ю кучу" << endl;

cout << "5 - удалить максимальный элемент 1-й кучи" << endl;

cout << "6 - удалить минимальный элемент 1-й кучи" << endl;

cout << "7 - удалить 1-й элемент 1-й кучи" << endl;

cout << "8 - объеденение 1-й и 2-й кучи" << endl;

cout << "0 - выход" << endl;

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

switch (choice)

{

case 0: exit(0);

case 1: h1.scan(0);

break;

case 2: { AAA\* a = new AAA;

cout << "введите ключ 1-й кучи" << endl; cin >> k;

a->x = k;

h1.insert(a);

}

break;

case 3: h2.scan(0);

break;

case 4: { AAA\* a = new AAA;

cout << "введите ключ 2-й кучи" << endl; cin >> k;

a->x = k;

h2.insert(a);

}

break;

case 5: h1.extractMax();

break;

case 6: h1.extractMin();

break;

case 7: h1.extractI();

break;

case 8:

{int g = 0, l = 0;

h3.size = h2.size + h1.size;

for (int i = 0; i <= h3.size; i++)

{

if (g >= 0 && g <= h1.size)

{

h3.storage[i] = h1.storage[g];

g++;

}

else {

g++;

}

if (l >= 0 && l <= h2.size && g > h1.size)

{

h3.storage[i] = h2.storage[l];

l++;

}

}

h3.heapify(0);

h3.scan(0);

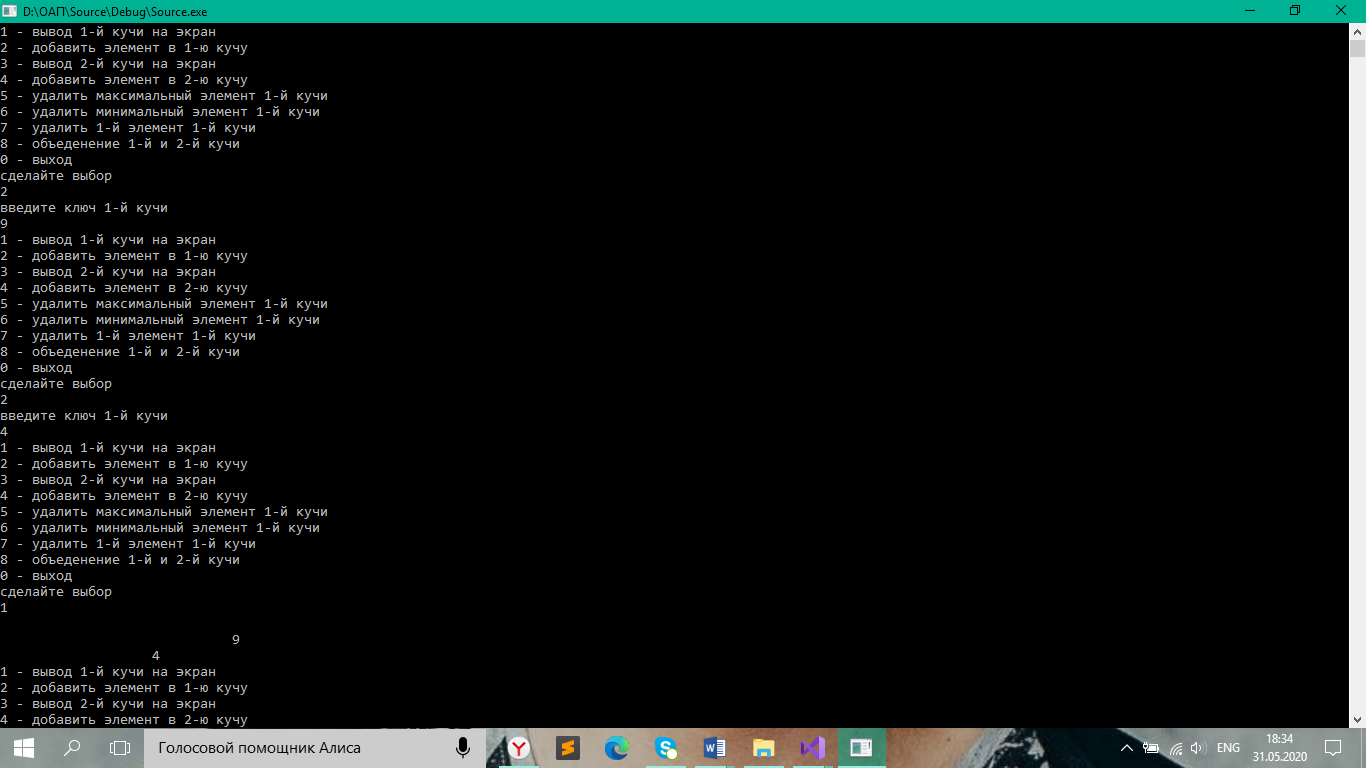
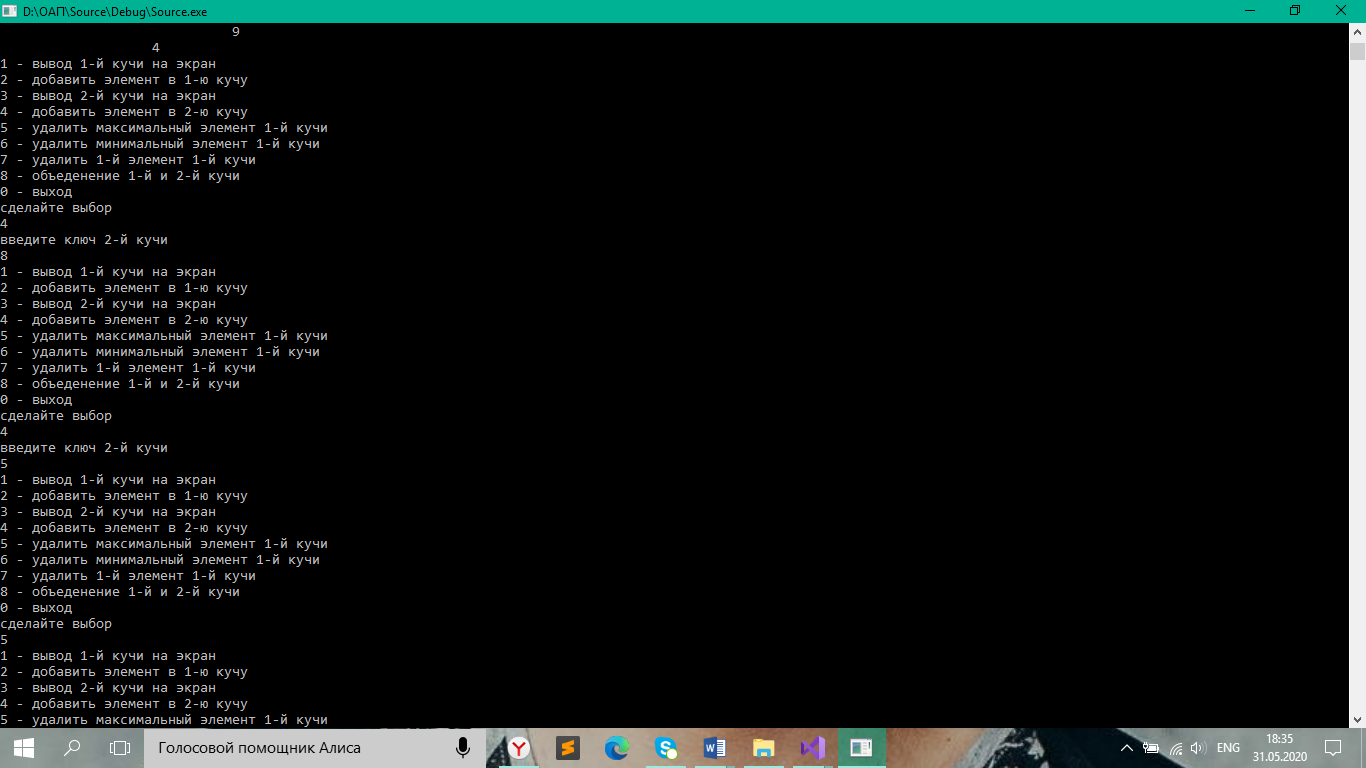
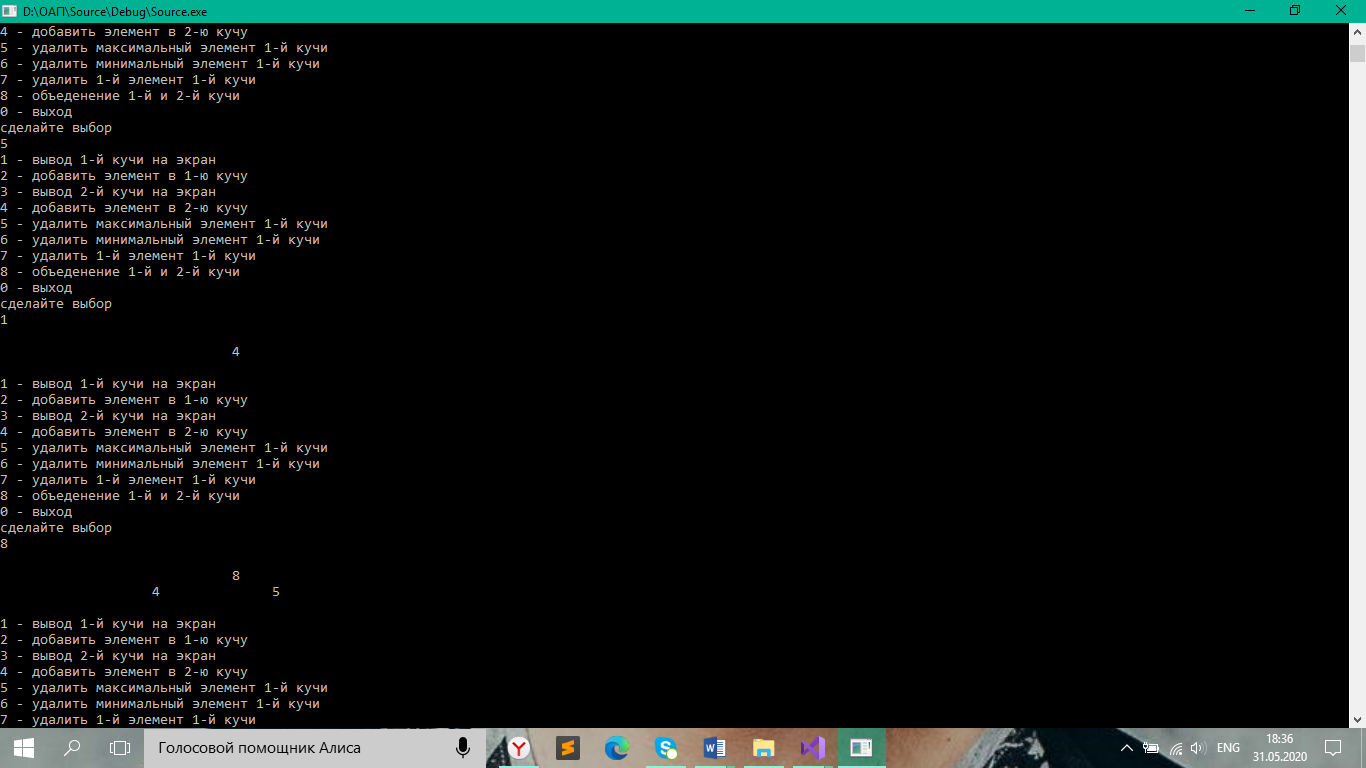
}

break;

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

}

} return 0;

}

**Лабораторная работа №14**

**Хеш-таблицы c открытой адресацией**

Использовать в проекте функции универсального хеширования и модульного. Сравнить время поиска информации.

**Главная функция:**

#include "Hash.h"

#include <iostream>

using namespace std;

//5 1 2

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;

}

char\* mas(void\* d)

{

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

}

char\* Object::getmas(void\* d)

{

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

}

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

void AAA\_print(void\* d)

{

cout << " ключ " << ((AAA\*)d)->key << " - " << ((AAA\*)d)->mas << endl;

}

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

int main()

{

setlocale(LC\_ALL, "rus");

AAA a1(1, (char\*)"one"), a2(2, (char\*)"two"), a3(4, (char\*)"three"), a4(2, (char\*)"fo");

int siz = 10, choice, k;

cout << "Введите размер хеш-таблицы" << endl; cin >> siz;

Object H = create(siz, key);

for (;;)

{

cout << "1 - вывод хеш-таблицы" << endl;

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

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

cout << "4 - поиск элемента" << endl;

cout << "0 - выход" << endl;

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

switch (choice)

{

case 0: exit(0);

case 1: H.scan(AAA\_print); break;

case 2: { AAA\* a = new AAA;

char\* str = new char[20];

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

a->key = k;

cout << "введите строку" << endl; cin >> str;

a->mas = str;

if (H.N == H.size)

cout << "Таблица заполнена" << endl;

else

H.insert(a);

} break;

case 3: { cout << "введите ключ для удаления" << endl;

cin >> k;

H.deleteByKey(k, H.data[k]);

} break;

case 4: { cout << "введите ключ для поиска" << endl;

cin >> k;

if (H.search(k, H.data[k]) == NULL)

cout << "Элемент не найден" << endl;

else

AAA\_print(H.search(k, H.data[k]));

} break;

}

}

return 0;

}

**Функции для манипуляции Hash.cpp:**

#include "Hash.h"

#include <iostream>

#include <stdlib.h>

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

int Next\_hash(int hash, int size, int p)

{

return (hash + 5 \* p + 3 \* p \* p) % size;

}

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

Object create(int size, int(\*getkey)(void\*))

{

return \*(new Object(size, getkey));

}

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

Object::Object(int size, int(\*getkey)(void\*))

{

N = 0;

this->size = size;

this->getKey = getkey;

this->data = new void\* [size];

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

data[i] = NULL;

}

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

bool Object::insert(void\* d)

{

bool b = false;

char\* k = 0;

if (N != size)

for (int i = 0, t = getKey(d), j = HashFunction(t, size, 0, k);

i != size && !b; j = Next\_hash(j, size, ++i))

if (data[j] == NULL || data[j] == DEL)

{

data[j] = d;

N++;

b = true;

}

return b;

}

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

int Object::searchInd(int key, void\* d)

{

int t = -1;

char\* k = 0;

bool b = false;

if (N != 0)

for (int i = 0, j = HashFunction(key, size, 0, k); data[j] != NULL && i != size && !b; j = HashFunction(key, size, ++i, k))

if (data[j] != DEL)

if (getKey(data[j]) == key)

{

t = j; b = true;

}

return t;

}

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

void\* Object::search(int key, void\* d) //функция поиска

{

int t = searchInd(key, d); //по индексу

return(t >= 0) ? (data[t]) : (NULL);

//если индекс больше 0, возвращаются данные, иначе ничего

}

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

void\* Object::deleteByKey(int key, void\* d) //функция удаления

{

int i = searchInd(key, d); //поиск по индексу

void\* t = data[i];//присвоение переменной данных элемента

if (t != NULL)

{

data[i] = DEL; //удаление элемента

N--; //уменьшение раз

}

return t;

}

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

bool Object::deleteByValue(void\* d) //функция удаления

{

return(deleteByKey(getKey(d), d) != NULL);

}

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

void Object::scan(void(\*f)(void\*))

{

for (int i = 0; i < this->size; i++)

{

std::cout << " Элемент" << i;

if ((this->data)[i] == NULL)

std::cout << " пусто" << std::endl;

else

if ((this->data)[i] == DEL)

std::cout << " удален" << std::endl;

else

f((this->data)[i]);

}

}

//модульная функция.Размер таблицы выбирается в виде простого числа m и вычисляется хэш-функция как остаток от деления

int Object::Module(int key, int size, int p, char\* mas)

{

int m = size;

unsigned int ans = key % m;

return ans;

}

//универсальная функция.Подразумевает случайный выбор хэш-функции из некоторого множества во время выполнения программы.

int Object::HashFunction(int key, int size, int p, char\* mas)

{

int h = 0, a = rand() % 10, b = a - 5;

h = (a \* key) % size;

return (h < 0) ? (h + size) : Module(key, size, p, mas); //выбор

}

**Заголовочный Hash.h**

#pragma once

#define HASHDEL (void\*) -1

struct Object

{

void\*\* data; //массив

Object(int, int(\*)(void\*)); //конструктор

int size; //размер таблицы

int N; //тек кол-во элементов

int(\*getKey)(void\*); //вычисление ключа

char\* getmas(void\* d);

bool insert(void\*);

int searchInd(int key, void\* d); //поиск индекса по ключу

void\* search(int key, void\* d); //поиск элемента по ключу

void\* deleteByKey(int key, void\* d); //удаление по ключу

bool deleteByValue(void\*); //удаление по значению

void scan(void(\*f)(void\*)); //вывод на экран

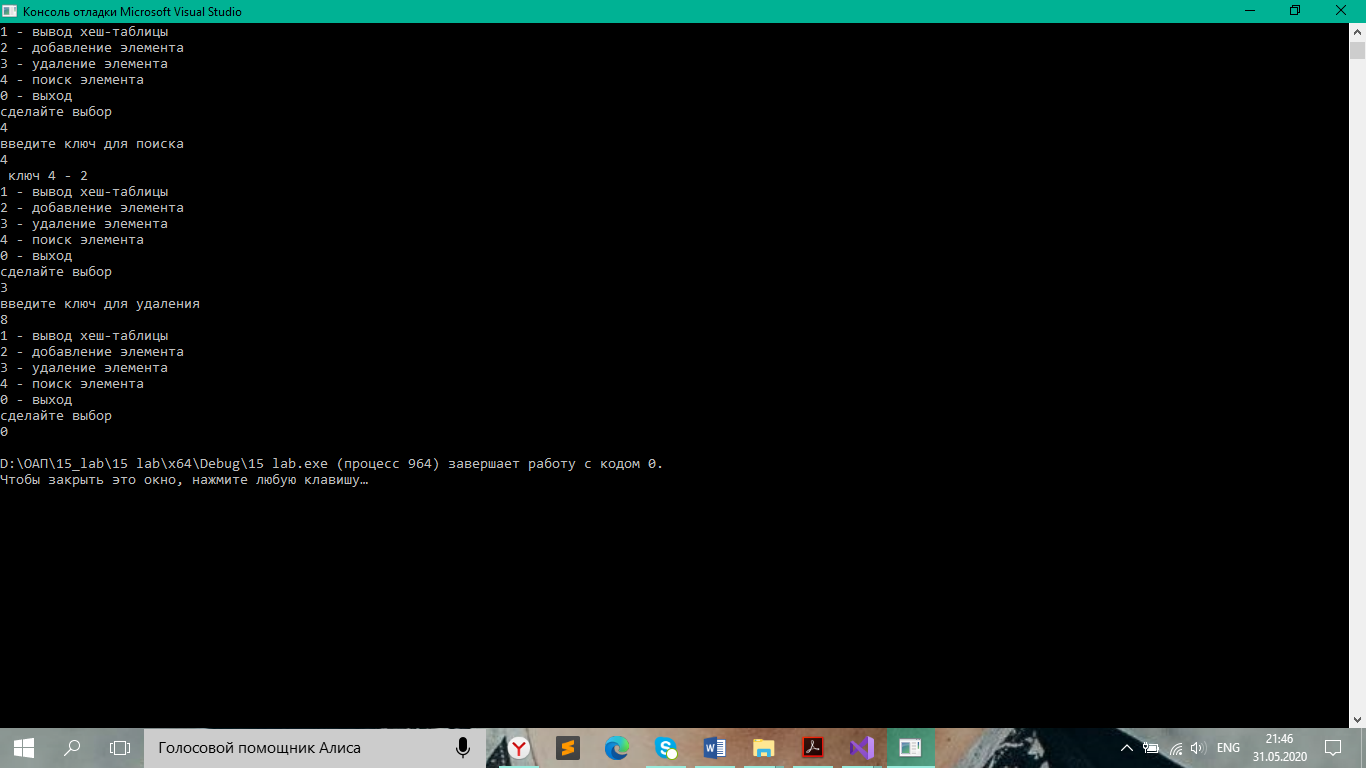
int Module(int key, int size, int p, char\* mas);

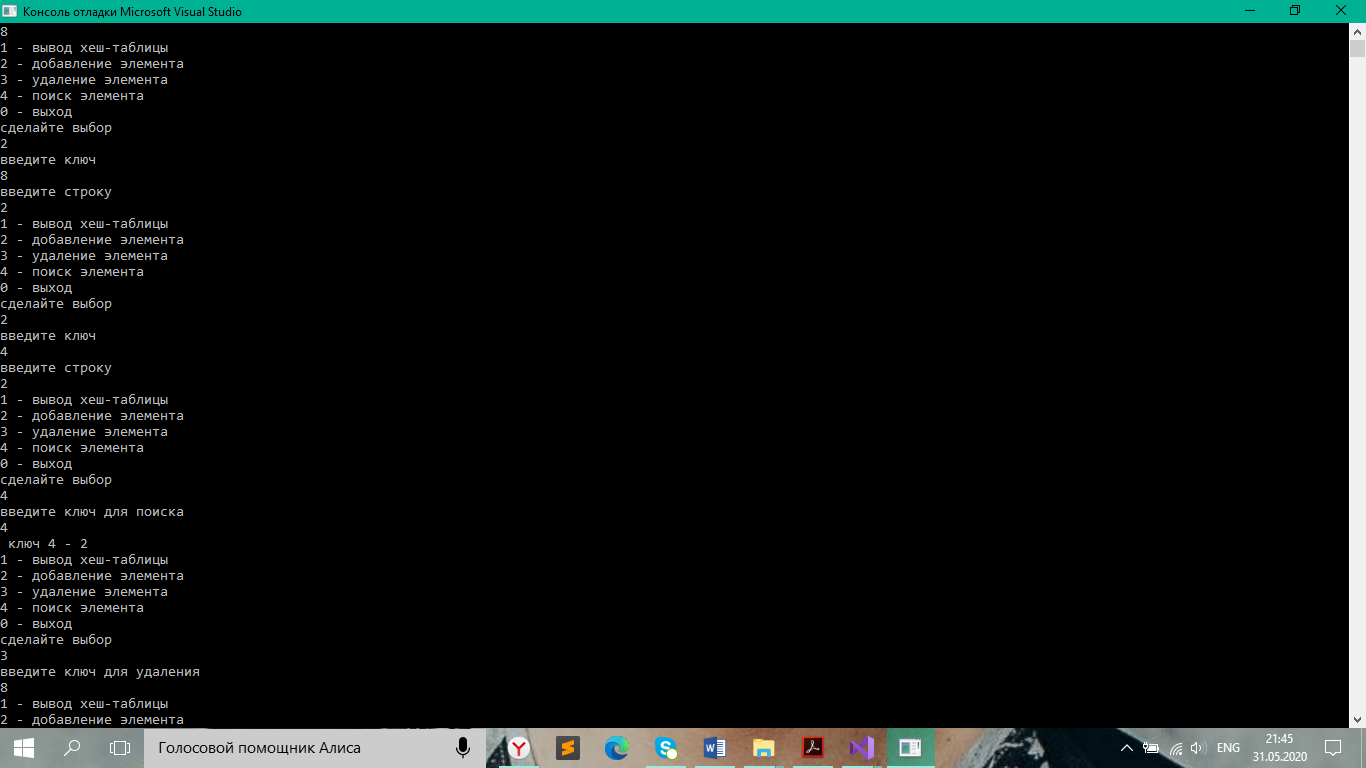
int HashFunction(int key, int size, int p, char\* mas);

};

static void\* DEL = (void\*)HASHDEL;

Object create(int size, int(\*getkey)(void\*));

#undef HASHDEL



**Лабораторная работа №15**

**Хеш-таблицы c цепочками**

**Университеты**.Реализовать хеш-таблицу со следующими полями: год создания, название университета. Ключ – год создания.

**Главная функция:**

#include "Hash\_ Twin\_Chain.h"

#include <iostream>

#include <windows.h>

using namespace std;

char c[] = "";

struct AAA

{

int key;

char\* mas;

AAA(int k, char\* z)

{

key = k;

mas = z;

}

AAA()

{

key = 0;

mas = c;

}

};

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

int hf(void\* d)

{

AAA\* f = (AAA\*)d;

return f->key;

}

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

void AAA\_print(listx::Element\* e)

{

std::cout << ((AAA\*)e->Data)->key << '-' << ((AAA\*)e->Data)->mas << " / ";

}

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

int main()

{

setlocale(LC\_ALL, "rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int current\_size = 7;

cout << "Введите размер хеш-таблицы" << endl;

cin >> current\_size;

hashTC::Object H = hashTC::create(current\_size, hf);

int choice;

int k;

for (;;)

{

cout << "1 - вывод хеш-таблицы" << endl;

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

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

cout << "4 - поиск элемента" << endl;

cout << "0 - выход" << endl;

cout << "сделайте выбор" << endl;

cin >> choice;

switch (choice)

{

case 0: exit(0);

case 2: { AAA\* a = new AAA;

char\* str = new char[20];

cout << "Введите ключ - год создания: ";

cin >> k;

a->key = k;

cout << "Название университета: ";

cin >> str;

a->mas = str;

H.insert(a);

}

break;

case 1: H.Scan();

break;

case 3: { AAA\* b = new AAA;

cout << "Введите ключ - год создания: ";

cin >> k;

b->key = k;

H.deleteByData(b);

}

break;

case 4: AAA \* c = new AAA;

cout << "Введите ключ - год создания: ";

cin >> k;

c->key = k;

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

cout << "Элемент не найден" << endl;

else

{

cout << "Первый элемент с данным ключом" << endl;

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

cout << endl;

}

break;

}

}

return 0;

}

**Функции для манипуляции Hash\_ Twin\_Chain.cpp:**

#include "Hash\_ Twin\_Chain.h"

#include "Lists.h"

#include <iostream>

int k = 3;

struct AAA

{

int key;

char \*mas;

AAA(int k, char\*z)

{

key = k;

mas = z;

}

};

namespace hashTC

{

Object create(int size, int(\*f)(void\*))

{

return \*(new Object(size, f));

}

int Object::hashFunction(void\* data)

{

int h, a = 31415, b = 27183, d = 1;

for (h = 0; d <= k; a = a \* b % (Size - 1))

{

h = (a\*h + FunKey(data)) % Size;

d++;

}

return (h < 0) ? (h + Size) : h;

};

bool Object::insert(void\* data)

{

return (Hash[hashFunction(data)].insert(data));

};

bool Object::deleteByData(void\* data)

{

return (Hash[hashFunction(data)].deleteByData(data));

};

listx::Element\* Object::search(void\* data)

{

return Hash[hashFunction(data)].search(data);

};

void Object::Scan()

{

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

{

Hash[i].scan();

std::cout << '\n';

}

};

}

**Функции для манипуляции Lists.cpp:**

#include "Lists.h"

#include <stdio.h>

#include <iostream>

using namespace std;

struct AAA //элемент таблицы

{

int key;

char \*mas;

};

namespace listx

{

bool Object::insert(void\* data)

{

bool rc = NULL;

if (Head == NULL)

Head = new Element(NULL, data, Head);

else

Head = (Head->Prev = new Element(NULL, data, Head));

return rc;

}

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

Element\* Object::search(void\* data)

{

Element\* rc = Head;

while ((rc != NULL) && ((((AAA\*)rc->Data)->key) != ((AAA\*)data)->key))

rc = rc->Next;

return rc;

}

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

bool Object::deleteByElement(Element\* e)

{

bool rc = NULL;

if (rc = (e != NULL))

{

if (e->Next != NULL)

e->Next->Prev = e->Prev;

if (e->Prev != NULL)

e->Prev->Next = e->Next;

else

Head = e->Next;

delete e;

}

std::cout << "Элемент удален" << std::endl;

return rc;

}

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

bool Object::deleteByData(void\* data)

{

return deleteByElement(search(data));

}

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

Element\* Object::getLast()

{

listx::Element\* e = this->getFirst(), \*rc = this->getFirst();

while (e != NULL)

{

rc = e;

e = e->getNext();

}

return rc;

}

Object create()

{

return \*(new Object());

};

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

void Object::scan()

{

listx::Element\* e = this->getFirst();

while (e != NULL)

{

std::cout << ((AAA\*)e->Data)->key << '-' << ((AAA\*)e->Data)->mas << " / ";

e = e->getNext();

};

}

}

**Заголовочная файл Hash\_ Twin\_Chain.h:**

#pragma once //Заголовочный файл Hash\_ Twin\_Chain.h

#include "Lists.h"

namespace hashTC

{

struct Object

{

int Size;

int(\*FunKey)(void\*);

listx::Object\* Hash;

Object(int size, int(\*f)(void\*))

{

Size = size;

FunKey = f;

Hash = new listx::Object[Size];

};

int hashFunction(void\* data);

bool insert(void\* data);

listx::Element\* search(void\* data);

bool deleteByData(void\* data);

void Scan();

};

Object create(int Size, int(\*f)(void\*));

}

**Заголовочная файл Lists.h:**

#pragma once

#include <stdio.h>

#define LISTNIL (Element\*)-1

namespace listx

{ struct Element

{ Element\* Prev;

Element\* Next;

void\* Data;

Element(Element\* prev, void\* data, Element\* next)

{ Prev = prev;

Data = data;

Next = next;

}

Element\* getNext()

{ return Next; };

Element\* getPrev()

{ return Prev; };

};

static Element\* NIL = NULL;

struct Object

{ Element\* Head;

Object()

{ Head = NIL; };

Element\* getFirst()

{ return Head; };

Element\* getLast();

Element\* search(void\* data);

bool insert(void\* data);

bool deleteByElement(Element\* e);

bool deleteByData(void\* data);

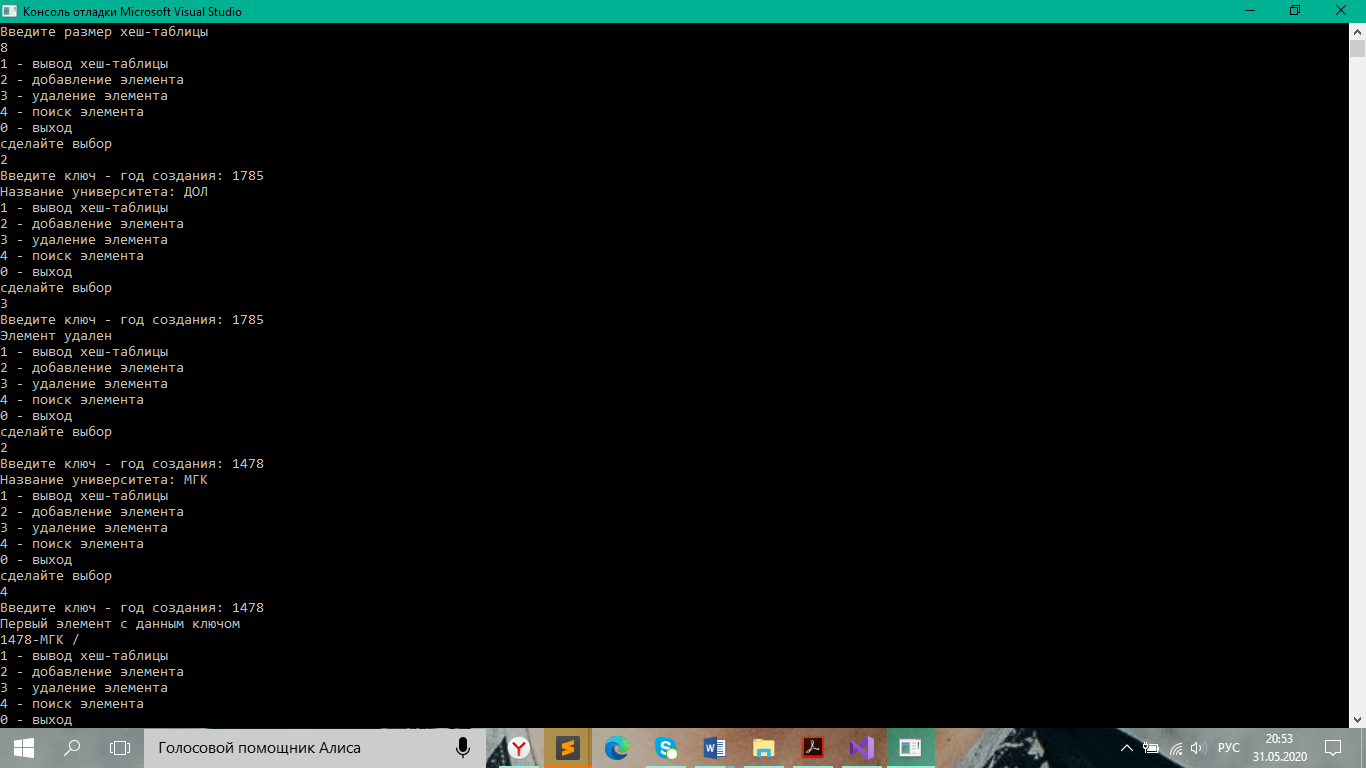
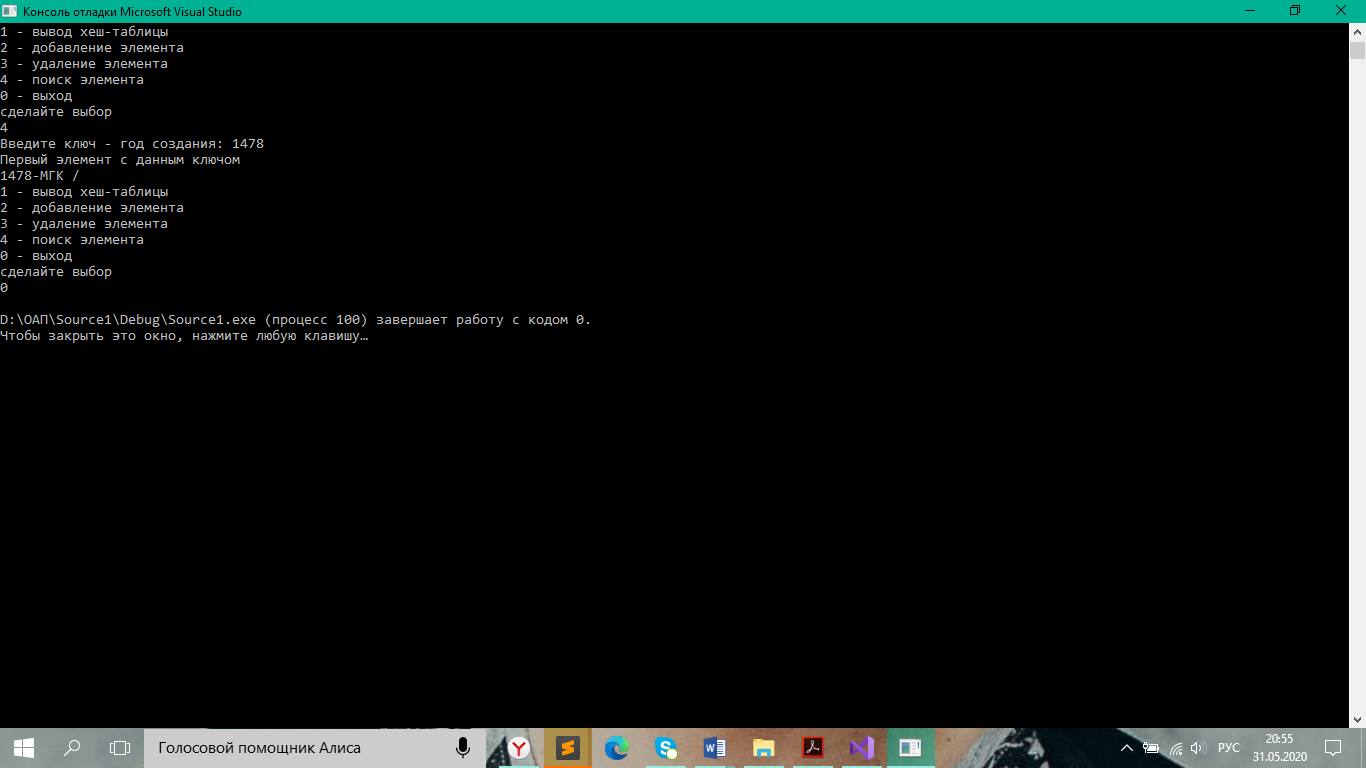
void scan();

};

Object create();

}

#undef LISTNIL



**Лабораторная работа №16**

**Анализ алгоритмов сортировок**

Ввести массивы **А** и **В**. В массив **С** перенести нечетные элементы массива **А** и четные элемента массива **В**. Массив **С** отсортировать по убыванию, используя алгоритмы сортировок: пирамидальная сортировка, «пузырек».

#include <iostream>

#include <ctime>

using namespace std;

void bubbleSort(int C[], int n) //сортировка пузырьком

{

int i, j, t;

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

for(j=n-1; j>=i; j--)

if (C[j - 1] < C[j])

{

t = C[j - 1];

C[j - 1] = C[j];

C[j] = t;

}

}

void heapify(int C[], int pos, int n) //сортировка пирамидой

{

int t, tm;

while (2 \* pos + 1 < n)

{

t = 2 \* pos + 1;

if (2 \* pos + 2 < n && C[2 \* pos + 2] < C[t])

t = 2 \* pos + 2;

if (C[pos] > C[t])

{

tm = C[pos];

C[pos] = C[t];

C[t] = tm;

pos = t;

}

else break;

}

}

void piramSort(int C[], int n)

{

for (int i = n - 1; i >= 0; i--)

heapify(C, i, n);

while (n>0)

{

int tm = C[0];

C[0] = C[n - 1];

C[n - 1] = tm;

n--;

heapify(C, 0, n);

}

}

int main()

{

setlocale(0, "rus");

int size1, size2, size3 = 0, i = 0, j = 0;

int A[10000], B[10000], C[20000];

int schet[10000];

int sm = 0, em;

cout << "Количество элементов массива А = ";

cin >> size1;

cout << "Количество элементов массива B = ";

cin >> size2;

unsigned int start\_time = clock(); //начальное время

cout << "\nМассив А: \n";

for (i = 0; i < size1; i++)

{

A[i] = rand() % 1000;

cout << A[i] << " ";

}

cout << endl;

cout << "\nМассив B: \n";

for (i = 0; i < size2; i++)

{

B[i] = rand() % 1000;

cout << B[i] << " ";

}

cout << endl;

for (i = 0; i < size1; i++)

{

if (A[i] % 2 == 1)

{

C[j] = A[i];

j++;

}

}

for (i = 0; i < size2; i++)

{

if (B[i] % 2 == 0)

{

C[j] = B[i];

j++;

}

size3 = j - 1;

}

//сортировка пузырьком

unsigned int start\_time\_bubbleSort = clock();

bubbleSort(C, size3);

unsigned int end\_time\_bubbleSort = clock(); //конечное время

unsigned int search\_time\_bubbleSort = end\_time\_bubbleSort - start\_time\_bubbleSort;

//искомое время

cout << "\n Сортировка пузырьком: \n";

for (i = 0; i < size3; i++)

cout << C[i] << " ";

cout << endl;

cout << "Время выполнения сортировки пирамидой: " << search\_time\_bubbleSort << " млс" << endl;

//пирамидальная сортировка

unsigned int start\_time\_piramSort = clock();

piramSort(C, size3);

unsigned int end\_time\_piramSort = clock(); //конечное время

unsigned int search\_time\_piramSort = end\_time\_piramSort - start\_time\_piramSort;

//искомое время

cout << "\n Сортировка пирамидой: \n";

for (i = 0; i < size3; i++)

cout << C[i] << " ";

cout << endl;

cout << "Время выполнения сортировки пирамидой: " << search\_time\_piramSort << " млс" << endl;

unsigned int end\_time = clock(); //конечное время

unsigned int search\_time = end\_time - start\_time; //искомое время

cout << "Время выполнения программы: " << search\_time << " млс";

}

