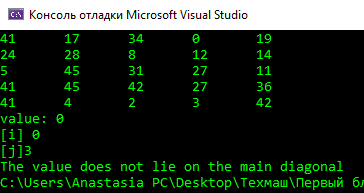
**//1 блок**

**//Лабораторная 1**

*2 задание*

/\*Дана целочисленная квадратная матрица.

Если минимальное значение находится на главной диагонали, то вывести его значение и индексы.\*/

**

#include <iostream>

using namespace std;

int elem(int, int);

int main()

{

int kol = 5;

int\*\* arr = new int\* [kol];

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

{

arr[i] = new int[kol];

}

for ( int i = 0; i < kol; i++) //ввод значений матрицы

{

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

{

arr[i][j] = rand() % 50;

}

}

for ( int i = 0; i < kol; i++) //вывод значений

{

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

{

cout << arr[i][j] << '\t';

}

cout << endl;

}

int min = arr[0][0];

int i, j,imin,jmin;

for ( i = 0; i < kol; i++) //minimum

{

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

{

if (arr[i][j] < min) {

min = arr[i][j];

imin = i;

jmin = j;

}

}

}

cout << "value: " << min << endl;

cout << "[i] " << imin << endl;

cout << "[j]" << jmin << endl;

elem(imin, jmin);

for (int i = 0; i < kol; i++) //удаление динамической памяти

{

delete[] arr[i];

}

delete[] arr;

}

int elem(int jmin, int imin) //собственная функция

{

if (imin == jmin)

{

cout << "result:" << imin << jmin << endl;

}

else {

cout << "The value does not lie on the main diagonal";

}

return imin, jmin;

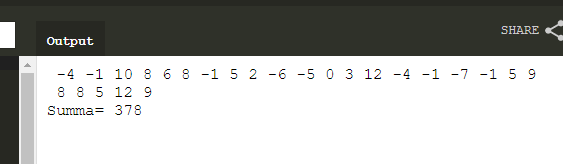
}

*1 задание*

/\*Дан массив A вещественного типа, содержащий 20 положительных и отрицательных элементов.

Сформировать массив B из положительных элементов массива A, имеющих четный индекс

Найти сумму квадратов элементов нового массива.\*/

**#include <iostream>

using namespace std;

double square(double\*, const int);

int main()

{

const int kol = 20;

double\* arrA = new double [kol];

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

{

arrA[i] = rand() % 20-7;

}

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

{

cout<<" "<<arrA[i];

}

cout << endl;

double\* arrB = new double[kol];

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

{

if ((arrA[i] > 0) &&((i+1)%2==0)) {

arrB[i] = arrA[i];

cout << " " <<arrB[i];

}

}

square(arrB,kol);

delete[] arrA;

delete[] arrB;

}

double square(double \*arrB,const int kol)

{

**double sum=0;**

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

sum += \*(arrB+i)\* \*(arrB+i);

}

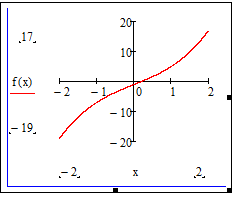
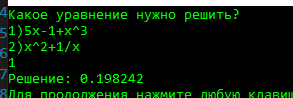
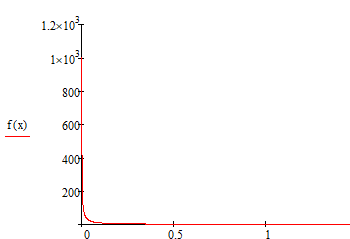
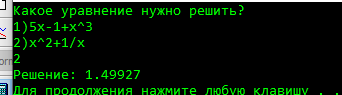
cout << endl;

cout << "Summa= " << sum;

return sum;

}

**//Лабораторная 2**

*Задание 1*

#include <iostream>

using namespace std;

double f1(double);

double f2(double);

double dichotomy(double (\*)(double), double, double);

int main() {

setlocale(0, "");

int number;

cout << "Какое уравнение нужно решить?\n1)5x-1+x^3\n2)x^2+1/x\n";

cin >> number;

switch (number) {

case 1: cout << "Решение: " << dichotomy(f1, -2, 2) << endl; break;

case 2: cout << "Решение: " << dichotomy(f2, 0, 1.5) << endl; break;

}

system("pause");

}

double f1(double x) {

return 5 \* x - 1 + x\*x\*x;

}

double f2(double x) {

return x \* x + 1/x ;

}

double dichotomy(double(\*f)(double), double a, double b) {

double x;

while (abs(a - b) > 0.001) {

x = (b + a) / 2;

if (f(x) \* f(a) <= 0) {

b = x;

}

else {

a = x;

}

}

return x;

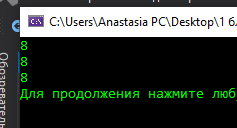
}

*Задание 2*

/\*Написать функцию, которая в предложении подсчитывает количество

символов в слове максимальной длины (слова разделяются пробелами).

Опробовать работу функции на нескольких предложениях.\*/

#include <iostream>

#include <stdarg.h>

#include <string>

using namespace std;

int maxs(const char\*, ...);

int main() {

setlocale(0, "");

cout << maxs("у", "Алисы", "Точкар", "сегодня", "праздник",

NULL) << endl;

cout << maxs("16", "марта", "день", "Рождение",

NULL) << endl;

cout << maxs("С", "днем", "Рождения", "Кроха", NULL) << endl;

system("pause");

}

int maxs(const char\* s1, ...) {

va\_list par;

char\* cp;

int len = strlen(s1);

va\_start(par, s1);

int max = strlen(s1);

while (cp = va\_arg(par, char\*)) {

len = strlen(cp);

if (max < len) {

max = len;

}

}

va\_end(par);

return max;

}

//Лабораторная 3

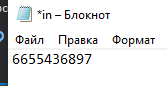
//Компоненты файла f – целые числа.Получить файл g, образованный из f исключением

//повторных вхождений одного и того же числа.

#include <iostream>

#include <stdio.h>

using namespace std;

void main() {

setlocale(0, "");

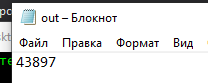
FILE\* A;

char s[10], s2[10];

errno\_t err = fopen\_s(&A, "in.txt", "r");

if (err != 0) {

perror("Ошибка открытия файла!");



return;

}

fgets(s, 51, A);

int len = strlen(s), repete = 0, in = 0;

fclose(A);

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

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

if (s[i] == s[j])

repete++;

}

if (repete <= 1) {

s2[in] = s[i];

in++;

}

repete = 0;

}

s2[in] = '\0';

err = fopen\_s(&A, "out.txt", "w");

if (err != 0) {

perror("Ошибка открытия файла!");

return;

}

fputs(s2, A);

fclose(A);

system("pause");

}

Компоненты файла fileA – целые числа, значения которых повторяются.Получить файл fileB,

из fileA числами, которые встречаются в fileA более двух раз.

#include <iostream>

#include <stdio.h>

using namespace std;

void main() {

setlocale(0, "");

FILE\* A;

char s[10], s2[10];

errno\_t err = fopen\_s(&A, "in.txt", "r");

if (err != 0) {

perror("Ошибка открытия файла!");

return;

}

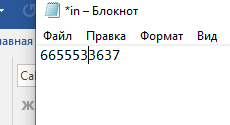
fgets(s, 51, A);

int len = strlen(s), repete = 0, in = 0;

fclose(A);

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

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

 if (s[i] == s[j])

repete++;

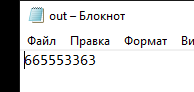
}

if (repete > 2) {

s2[in] = s[i];

in++;

}

 else s2[i] = '\0';

repete = 0;

}

err = fopen\_s(&A, "out.txt", "w");

if (err != 0) {

perror("Ошибка открытия файла!");

return;

}

fputs(s2, A);

fclose(A);

system("pause");

}

**//2 БЛОК**

*Лабораторная работа 4*

*/\*Скопировать из файла FILE1 в файл FILE2 все строки, в которых более 2 слов. Определить номер слова, в котором больше всего гласных букв.*

*Ввести с клавиатуры строку символов, состоящую из цифр и скобок, и записать ее в файл. Прочитать из файла данные, посчитать и вывести количество скобок различного вида.\*/*

#include <iostream>

#include <fstream>

#include <windows.h>

#include <stdio.h>

#include <cstdio>

using namespace std;

void v11\_1();

void v11\_2();

void outFile(ofstream& f, char\* buffer, char\* str);

void inFile(ifstream& f, char\* buffer, char\* str);

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int choice;

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

cin >> choice;

switch (choice)

{

case 1:

{

v11\_1();

break;

}

case 2:

{

v11\_2();

break;

}

}

}

void v11\_1()

{

char buffer[50];

int minVowel = 0, numberOfSent = 0, countVowel = 0, necessSent = 0;

ifstream fin("ff1.txt");

ofstream fout("FILE2.txt");

if (!fin.is\_open())

cout << "Файл не может быть открыт!\n";

else

{

while (!fin.eof())

{

fin.getline(buffer, 50);

int sizeOfStr = strlen(buffer);

bool isSpace = false;

bool strCopied = false;

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

{

if (buffer[i] == ' ' && isSpace)

{

fout << buffer;

fout << endl;

strCopied = !strCopied;

break;

}

if (buffer[i] == ' ' && !isSpace)

{

isSpace = !isSpace;

}

}

if (strCopied)

{

for (int k = 0; k < sizeOfStr + 1; k++)

{

if (buffer[k] == 'A' || buffer[k] == 'a' || buffer[k] == 'E' || buffer[k] == 'e' || buffer[k] == 'I' || buffer[k] == 'i' || buffer[k] == 'O' || buffer[k] == 'o' || buffer[k] == 'U' || buffer[k] == 'u' || buffer[k] == 'Y' || buffer[k] == 'y')

{

countVowel++;

}

if (buffer[k] == ' ' || buffer[k] == '\0')

{

numberOfSent++;

if (countVowel > minVowel)

{

minVowel = countVowel;

necessSent = numberOfSent;

}

countVowel = 0;

}

}

}

}

}

cout << "Большее количество гласных (" << minVowel << ") в (во) " << necessSent << " слове." << endl;

if (fin.eof())

{

cout << "~~Достигнут конец файла~~" << endl;

}

fin.close();

fout.close();

}

void v11\_2()

{

char buffer[50];

char str[200];

int i = 0;

int roundBrackets = 0, fiqureBrakets = 0, squareBrakets = 0, angledBrackets = 0;

ifstream fin;

ofstream fout;

cout << "Введите путь к файлу для записи: " << endl;

cin >> str;

cout << "Введите строку для записи: " << endl;

cin >> buffer;

outFile(fout, buffer, str);

inFile(fin, buffer, str);

while (buffer[i] != EOF)

{

if (buffer[i] == '(' || buffer[i] == ')')

{

roundBrackets++;

}

if (buffer[i] == '[' || buffer[i] == ']')

{

squareBrakets++;

}

if (buffer[i] == '{' || buffer[i] == '}')

{

fiqureBrakets++;

}

if (buffer[i] == '<' || buffer[i] == '>')

{

angledBrackets++;

}

i++;

}

cout << "Круглых скобок: " << roundBrackets << endl;

cout << "Квадратных скобок: " << squareBrakets << endl;

cout << "Фигурных скобок: " << fiqureBrakets << endl;

cout << "Угловых скобок: " << angledBrackets << endl;

}

void outFile(ofstream& f, char\* buff, char\* str)

{

f.open(str);

if (f.fail())

{

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

}

else {f << buff;

f.close();}}

void inFile(ifstream& f, char\* buff, char\* str)

{

f.open(str);

if (f.fail())

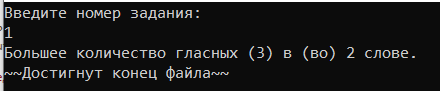
{

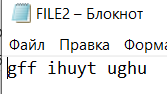
cout << "Ошибка открытия файл" << endl;

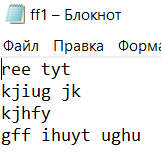
}

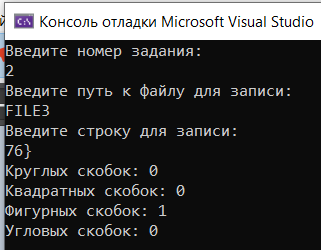
else {f >> buff;

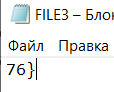
f.close();}}











**Начало**

i = 0; roundBrackets = 0, fiqureBrakets = 0, squareBrakets = 0, angledBrackets = 0;

Ifsteram fin; ofstream fout

Ввод str и buffer

нет

f.fail()

Вывод "Ошибка открытия файла"

f << buff;

f.close();

f.open(str);

1

нет

f.fail()

1

Вывод "Ошибка открытия файла"

f >> buff;

f.close();

buffer[i]!=EOF

нет

нет

buffer[i] == '(' || buffer[i] == ')'

roundBrackets++

buffer[i] == '[' || buffer[i] == ']'

squareBrakets++

нет

2

нет

buffer[i] == '{' || buffer[i] == '}'

fiqureBrakets++

2

нет

buffer[i] == '<' || buffer[i] == '>'

angledBrackets++

**Конец**

Вывод roundBrackets, squareBrakets, fiqureBrakets, angledBrackets

i++

*Лабораторная работа 5*

*//Вокзал.Номер поезда, пункт назначения, дни следования, время прибытия, время //отправления.Выбор по пункту назначения.*

#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

#define size 3

void enterNew();

void del();

void find(char\* findname);

void out();

void record();

void read();

void menu();

struct Trains

{

char number[20];

char destination[20];

char days[40];

char arrTime[10];

char depTime[10];

};

struct Trains list[size];

struct Trains bad;

int choice, currentSize = 0;

ifstream fin("trains.txt");

ofstream fout("trains.txt");

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

char findDest[20];

cout << "РАСПИСАНИЕ ПОЕЗДОВ ИЗ МИНСКА!" << endl;

menu();

cout << "Выберите пункт: ";

cin >> choice;

do

{

switch (choice)

{

case 1: enterNew(); break;

case 2: record(); break;

case 3: out(); break;

case 4: read(); break;

case 5: del(); break;

case 6: cout << "Введите пункт назначения" << endl; cin >> findDest; find(findDest); break;

}

} while (choice != 7);

cout << "~~Выход~~"; return;

}

void menu()

{

cout << "\n1.Ввод новой записи\n";

cout << "2.Запись информации в файл\n";

cout << "3.Вывод данных структуры\n";

cout << "4.Чтение данных из файла\n";

cout << "5.Удаление структуры данных\n";

cout << "6.Поиск информации\n";

cout << "7.Выход\n";

}

void enterNew()

{

cout << endl << "~~Ввод информации~~" << endl << endl;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

if (currentSize < size)

{

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

cout << "Номер поезда: "; cin >> list[currentSize].number; cout << endl;

cout << "Пункт назначения: "; cin >> list[currentSize].destination; cout << endl;

cout << "Дни следования: "; cin >> list[currentSize].days; cout << endl;

cout << "Время прибытия: "; cin >> list[currentSize].arrTime; cout << endl;

cout << "Время отправления: "; cin >> list[currentSize].depTime; cout << endl;

currentSize++;

}

else

{

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

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

}

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

cout << "~~Запись добавлена~~" << endl << endl;

menu();

cout << "Что дальше? (выберите пункт)" << endl;

cin >> choice;

}

void record()

{

cout << endl << "~~Запись данных в файл~~" << endl << endl;

int i = 0;

if (!fout.is\_open())

cout << "Файл не может быть открыт!\n";

else

{

while (i < currentSize)

{

fout << "Строка номер: " << i + 1 << endl;

fout << "Номер поезда: " << list[i].number << endl;

fout << "Пункт назначения: " << list[i].destination << endl;

fout << "Дни следования: " << list[i].days << endl;

fout << "Время прибытия: " << list[i].arrTime << endl;

fout << "Время отправления: " << list[i].depTime << endl;

i++;

fout << endl;

}

cout << "~~Запись прошла успешно~~" << endl;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

}

menu();

cout << "Что дальше? (выберите пункт)" << endl;

cin >> choice;

}

void out()

{

int strout, number = 1;

cout << endl << "~~Вывод структуры~~" << endl << endl;

cout << "1.Вывод одной строки" << endl;

cout << "2.Вывод всех строк" << endl;

cin >> strout;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

if (strout == 1)

{

cout << "Введите номер строки" << endl;

cin >> number;

cout << "Строка номер: " << number << endl;

cout << "Номер поезда: " << list[number - 1].number << endl;

cout << "Пункт назначения: " << list[number - 1].destination << endl;

cout << "Дни следования: " << list[number - 1].days << endl;

cout << "Время прибытия: " << list[number - 1].arrTime << endl;

cout << "Время отправления: " << list[number - 1].depTime << endl;

}

if (strout == 2)

{

while (number < currentSize + 1)

{

cout << "Строка номер: " << number << endl;

cout << "Номер поезда: " << list[number - 1].number << endl;

cout << "Пункт назначения: " << list[number - 1].destination << endl;

cout << "Дни следования: " << list[number - 1].days << endl;

cout << "Время прибытия: " << list[number - 1].arrTime << endl;

cout << "Время отправления: " << list[number - 1].depTime << endl;

number++;

cout << endl;

}

}

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

menu();

cout << "Что дальше? (выберите вариант)" << endl;

cin >> choice;

}

void read()

{

cout << endl << "~~Чтение из файла~~" << endl << endl;

char bufFile[50];

while (!fin.eof())

{

fin.getline(bufFile, 50);

cout << bufFile << endl;

}

fin.seekg(0);

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

menu();

cout << "Что дальше? (выберите вариант)" << endl;

cin >> choice;

}

void del()

{

int d;

cout << endl << "~~Введите номер строки, которую хотите удалить (для удаления всех строк введите -1)~~" << endl << endl;

cin >> d;

if (d != -1)

{

for (int dstr = (d - 1); dstr < currentSize; dstr++)

{

list[dstr] = list[dstr + 1];

}

currentSize--;

}

if (d == -1)

{

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

{

list[i] = bad;

}

}

cout << "~~Удаление выполнено~~" << endl;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

menu();

cout << "Что дальше? (выберите вариант)" << endl;

cin >> choice;

}

void find(char\* findname)

{

bool noFind = false;

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

{

if (strcmp(findname, list[i].destination) == 0)

{

cout << "\nНомер поезда\t" << list[i].number << endl;

cout << "Пункт назначения\t" << list[i].destination << endl;

cout << "Дни следования:\t" << list[i].days << endl;

cout << "Время прибытия:\t " << list[i].arrTime << endl;

cout << "Время отправления:\t " << list[i].depTime << endl;

noFind = !noFind;

break;

}

}

if (!noFind)

{

cout << "~~Совпадений не найдено~~" << endl;

}

cout << "~~Поиск завершён~~" << endl;

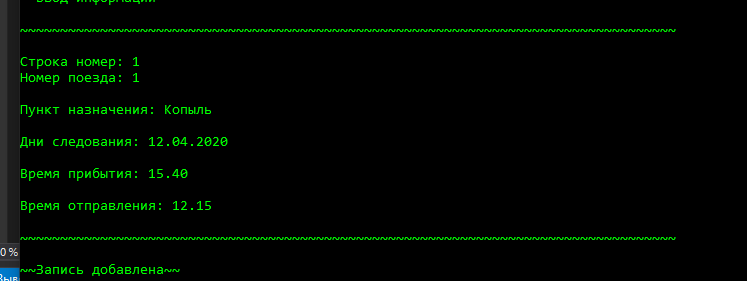
cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

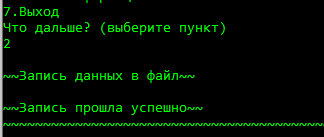
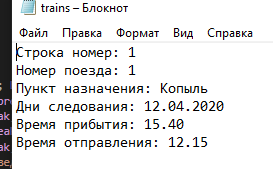
menu();

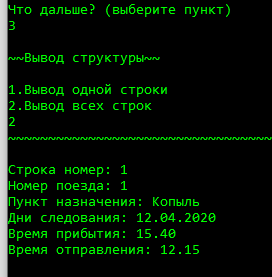
cout << "Что дальше? (выберите вариант)" << endl;

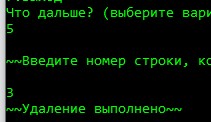
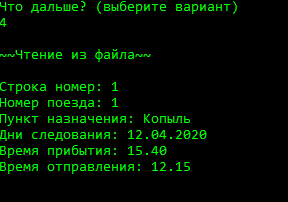
cin >> choice;

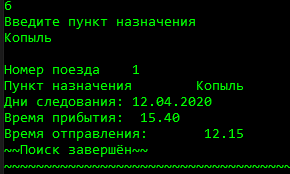
}

**

**

**

**

**

*Лабораторная работа 6*

**//Студенты**. Ф.И.О., дата поступления, специальность, группа, факультет, средний балл. Выбор по среднему баллу. Дату поступления реализовать с помощью битового поля.

#include <iostream>

using namespace std;

void enter(int\*);

void out(int, int\*);

void del(int, int\*);

void find(int\*);

struct byte {

int months : 12;

int days : 31;

};

struct student{

char name[20];

char spec[30];

char facult[30];

int grup;

int sred;

byte date;

};

struct student stud[10];

struct student stud0;

int main()

{

setlocale(LC\_ALL, "rus");

int choise;

do {

cout << "\n1.Ввод информации о студенте." << endl;

cout << "2.Вывод информации о студенте." << endl;

cout << "3.Хорошист/отличник/троечник" << endl;

cout << "4.Удаление записи." << endl;

cout << "5.Закончить программу." << endl;

cout << "Выберите номер операции: ";

cin >> choise;

switch (choise) {

case 1:

cout << "Сколько записей хотите сделать?";

int q;

cin >> q;

enter(&q);

break;

case 2:

cout << "О какой записи вывести информацию? Если о всех, введите 444: ";

int k;

cin >> k;

out(k, &q);

break;

case 3:

cout << "Определим по среднему баллу, является ли студент хорошистом, отличником или средним студентом";

find(&q);

break;

case 4:

cout << "Какую запись удалить? если все, введите 444: ";

int s;

cin >> s;

del(s, &q);

break;

case 5:

exit(0); break;

default:

cout << "Некорректный ввод!" << endl;

}

} while (choise != 6);

}

void enter(int\* q) {

for (int i = 0; i < \*q; i++) {

cout << "Фамилия студента: ";

cin >> stud[i].name;

cout << "Название специальности: ";

cin >> stud[i].spec;

cout << "Название факультета: ";

cin >> stud[i].facult;

cout << "Номер группы ";

cin >> stud[i].grup;

cout << "Введите срений балл студента: ";

cin >> stud[i].sred;

cout << "Введите месяц поступления: ";

int a;

cin >> a;

stud[i].date.months = a;

cout << "Введите день поступления: ";

int b;

cin >> b;

stud[i].date.days = b;

}

}

void out(int k, int\* q) {

cout << "Выведем информацию: ";

if (k == 444) {

int p = 0;

while (p < \*q) {

cout << "Фамилия: " << stud[p].name << endl;

cout << "Название специальности: " << stud[p].spec << endl;

cout << "Название факультета: " << stud[p].facult << endl;

cout << "Номер группы: " << stud[p].grup << endl;

cout << "Средний балл студента: " << stud[p].sred << endl;

cout << "День поступления: " << stud[p].date.days << endl;

cout << "Месяц поступления: " << stud[p].date.months << endl;

p++;

}

}

else {

cout << "Фамилия: " << stud[k-1].name << endl;

cout << "Название специальности: " << stud[k-1].spec << endl;

cout << "Название факультета: " << stud[k-1].facult << endl;

cout << "Номер группы: " << stud[k-1].grup << endl;

cout << "Средний балл студента: " << stud[k-1].sred << endl;

cout << "День поступления: " << stud[k-1].date.days << endl;

cout << "Месяц поступления: " << stud[k-1].date.months << endl;

}

}

void find(int\* q) {

for (int i = 0; i < \*q; i++) {

if ((stud[i].sred) < 4)cout << "Студенту " << stud[i].name << " стоит повысить свой сред. балл(троечник)";

if ((stud[i].sred) > 4 && (stud[i].sred) < 7) cout << "Студент " << stud[i].name << " хорошист!";

else cout << "Студент " << stud[i].name << " отличник!";

}

}

void del(int s, int\* q) {

if (s == 444) {

int del = 0;

while (del < \*q) {

stud[del] = stud0;

}

\*q = 0;

cout << "Все записи удалены!" << endl;

}

else {

for (int i = s - 1; i < \*q; i++)

{

stud[i] = stud[i + 1];

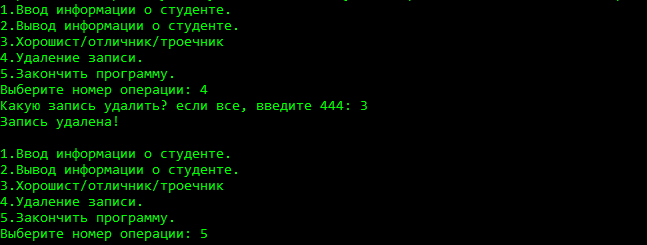
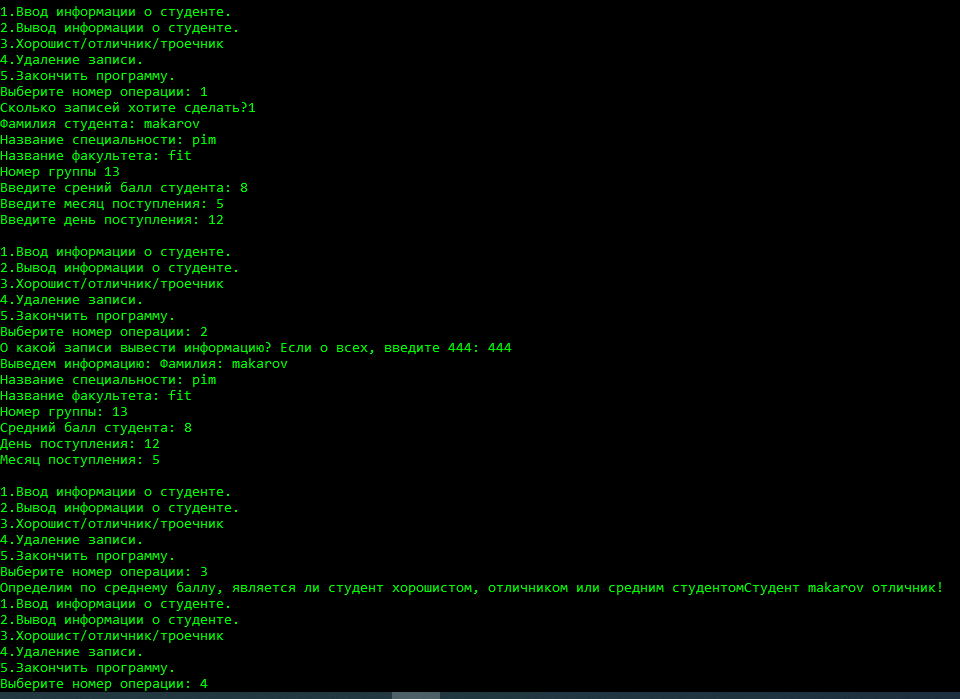
}

\*q = \*q - 1;

cout << "Запись удалена!" << endl;

}

}

**

**Государство**. Наименование, столица, численность населения, площадь, фамилия президента. Выбор государства по названию.

#include <cstring>

#include <iostream>

using namespace std;

void enter(int\*);

void out(int, int\*);

void find(int\*);

struct Country {

char country[15];

union {

char capital[20];

int number;

int square;

char name[20];

};

};

struct Country stol[15];

int main()

{

setlocale(LC\_ALL, "rus");

int choice;

do {

cout << "\n1.ввод информации.";

cout << "\n2.вывод информации.";

cout << "\n3.поиск информации (по названию).";

cout << "\n4.выход из программы.";

cout << "\nвведите номер операции: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "Сколько записей хотите сделать?";

int q;

cin >> q;

enter(&q);

break;

case 2:

cout << "О какой стране хотите вывести информацию? (если о всех, введите 444): ";

int k;

cin >> k;

out(k, &q);

break;

case 3:

find(&q);

break;

case 4:

exit(0); break;

default:

cout << "Некорректный ввод!" << endl;

}

} while (choice != 5);

}

void enter(int\* q) {

for (int i = 0; i < \*q; i++) {

cout << "Страна " << (i + 1);

cin >> stol[i].country;

cout << "Столица " << (i + 1) << " этой страны ";

cin >> stol[i].capital;

cout << " Численность состовляет" << (i + 1) << " человек ";

cin >> stol[i].number;

cout << "Площадь состовляет " << (i + 1) << " км^2 ";

cin >> stol[i].square;

cout << "Президент " << (i + 1);

cin >> stol[i].name;

}

}

void out(int k, int\* q) {

if (k == 444) {

for (int i = 0; i < \*q; i++) {

cout << (i + 1) << ".Страна: " << stol[i].country << endl;

cout << (i + 1) << ".Столица: " << stol[i].capital << endl;

cout << (i + 1) << ".Численность: " << stol[i].number << endl;

cout << (i + 1) << ".Площадь: " << stol[i].square << endl;

cout << (i + 1) << ".Президент: " << stol[i].name << endl;

}

}

else {

cout << (k) << ".Страна: " << stol[k - 1].country << endl;

cout << (k) << ".Столица: " << stol[k - 1].capital << endl;

cout << (k) << ".Численность: " << stol[k - 1].number << endl;

cout << (k) << ".Площадь: " << stol[k - 1].square << endl;

cout << (k) << ".Президент: " << stol[k - 1].name << endl;

}

}

void find(int\* q) {

char n[20];

cout << "\nВведите страну: ";

cin >> n;

for (int i = 0; i < \*q; i++) {

int count = 0;

for (int j = 0; j < strlen(n); j++) {

if (stol[i].country[j] == n[j]) {

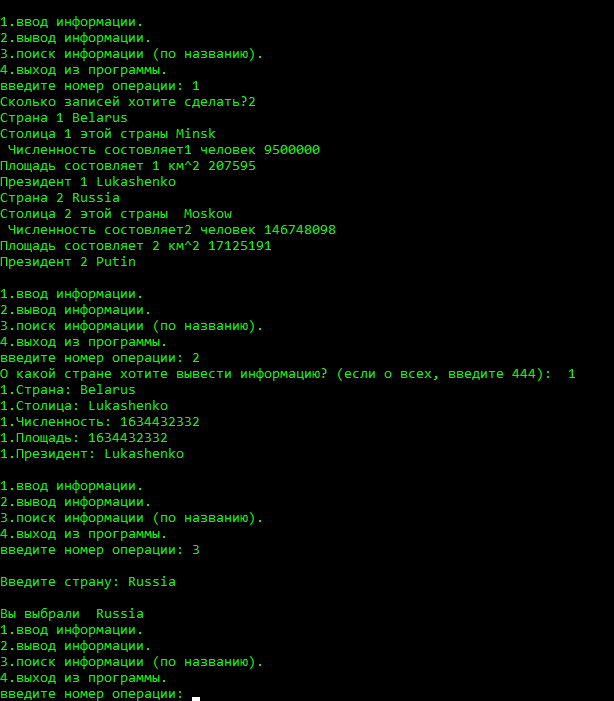
count++;

if (count == strlen(n)) cout << "\nВы выбрали " << stol[i].country;

}

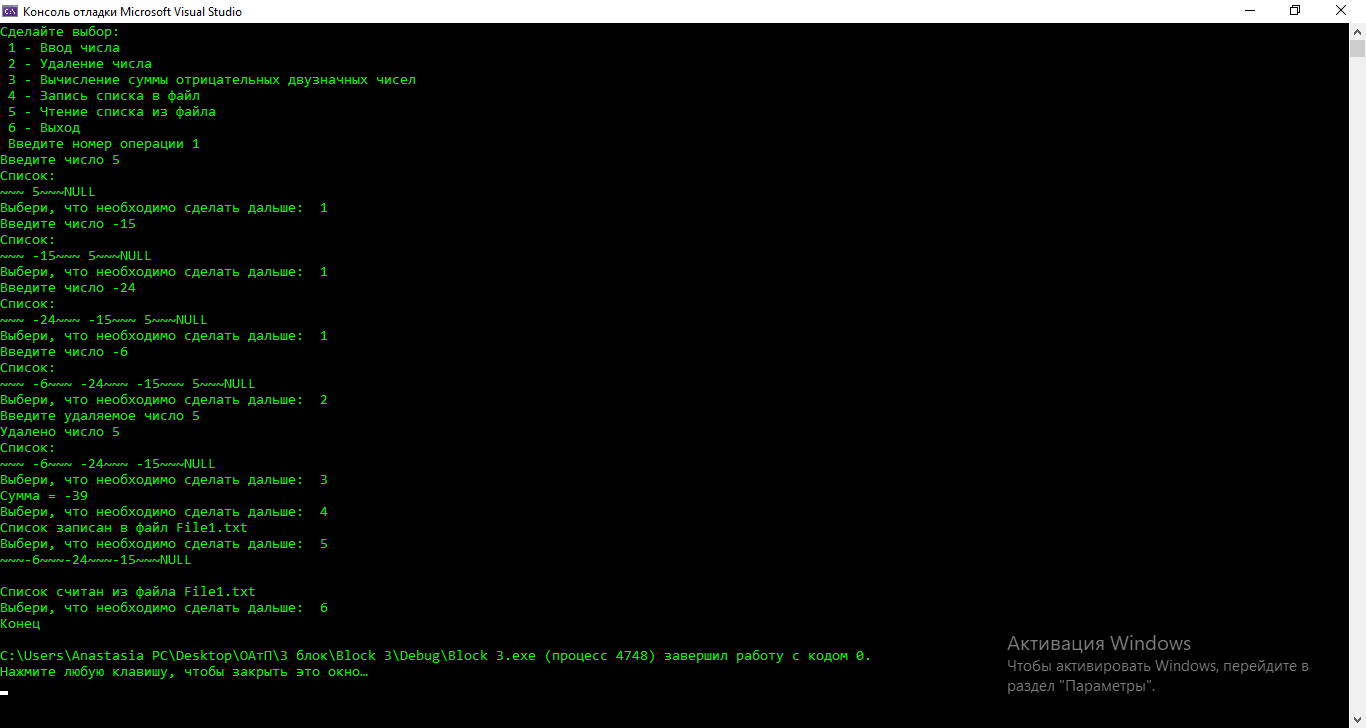
}

}

}

//Блок 3

*//Лабораторная работа 7*

// Создать список, содержащий элементы целого типа. Найти сумму отрицательных двухзначных элементов 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\*&, float); //функция добавления элемента, передается адрес списка и символ, который добавляется

float del(list\*&, float); //функция удаления, передается адрес списка и символ, который удаляется

int IsEmpty(list\*); //функция, которая проверяет, пуст ли список

void printList(list\*); //функция вывода

void menu(void); //функция, показывающая меню

void sum7(list\*); //функция подсчета суммы отрицательных двузначных чисел

void toFile(list\*& p); //Запись в файл

void fromFile(list\*& p); //Вывод из файла

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

list\* first = NULL;

int choice;

float value;

menu(); // вывести меню

cout << " Введите номер операции ";

cin >> choice;

while (choice != 6)

{

switch (choice)

{

case 1: cout << "Введите число "; // добавить число в список

cin >> value;

insert(first, value);

printList(first);

break;

case 2: if (!IsEmpty(first)) // удалить число из списка

{

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: sum7(first); // вычисление суммы

break;

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

menu();

break;

case 4: toFile(first);

break;

case 5: fromFile(first);

break;

}

cout << "Выбери, что необходимо сделать дальше: ";

cin >> choice;

}

cout << "Конец" << endl;

return 0;

}

void menu(void) //Вывод меню

{

cout << "Сделайте выбор:" << endl;

cout << " 1 - Ввод числа" << endl;

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

cout << " 3 - Вычисление суммы отрицательных двузначных чисел " << endl;

cout << " 4 - Запись списка в файл" << endl;

cout << " 5 - Чтение списка из файла" << endl;

cout << " 6 - Выход" << endl;

}

void insert(list\*& p, float value) //Добавление числа value в список

{

list\* newP = new list;

if (newP != NULL) //есть ли место?

{

newP->number = value;

newP->next = p;

p = newP;

}

else

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

}

float del(list\*& p, float value) // Удаление числа

{

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;

delete(temp);

return value;

}

}

return 0;

}

int IsEmpty(list\* p) //Список пустой? (1-да, 0-нет)

{

return p == NULL;

}

void printList(list\* p) //Вывод списка

{

if (p == NULL)

cout << "Список пуст" << endl;

else

{

cout << "Список:" << endl;

while (p != NULL)

{

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

p = p->next;

}

cout << "~~~NULL" << endl;

}

}

void sum7(list\* p) // Подсчет суммы отрицательных двузначных

{

float sm = 0;

if (p == NULL)

cout << "Таких элементов нет" << endl;

else

{

while (p != NULL)

{

if (p->number < -9)

sm = sm + (p->number);

p = p->next;

}

cout << "Сумма = " << sm << endl;

}

}

void toFile(list\*& p)

{

list\* temp = p;

list buf;

ofstream frm("File1.txt");

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 File1.txt\n";

}

void fromFile(list\*& p) //Считывание из файла

{

list buf, \* first = nullptr;

ifstream frm("File1.txt");

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();

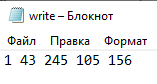
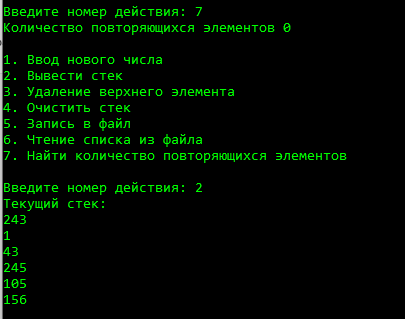
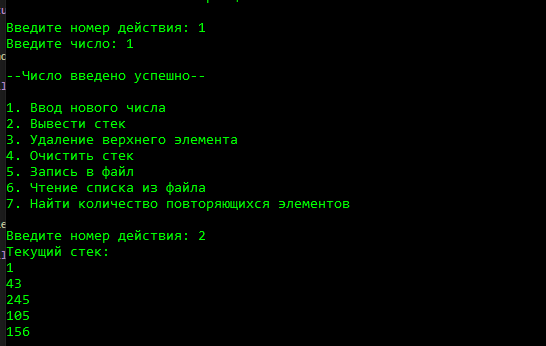
p = first;

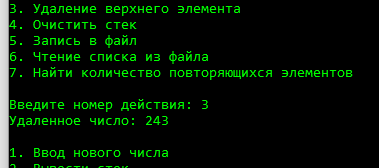
cout << "\nСписок считан из файла File1.txt\n";

}

*//Лабораторная работа 8*

// Разработать функцию подсчета количества повторяющихся элементов стека.



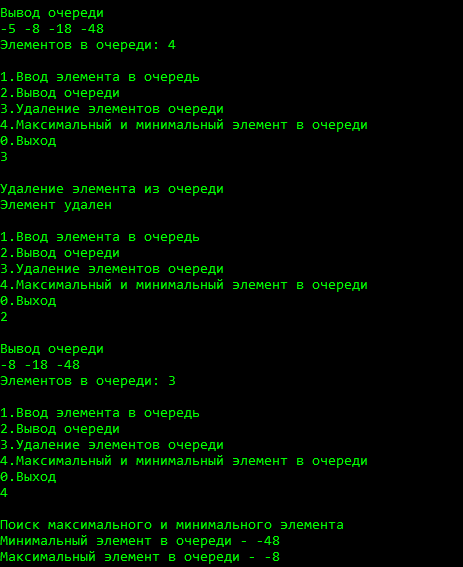


***proectStack.cpp myStack.cpp***

|  |  |
| --- | --- |
| #include <iostream>  #include <fstream>  #include "Header.h"  #include <windows.h>  using namespace std;  void main()  { SetConsoleCP(1251);  SetConsoleOutputCP(1251);  Stack\* head = NULL;  int choice;  do{  menu();  cout << "\nВведите номер действия: ";  cin >> choice;  switch (choice)  {  case 1:  int number;  cout << "Введите число: ";  cin >> number;  push(number, &head);  break;  case 2:  cout << "Текущий стек:";  cout << endl;  show(head);  break;  case 3:  cout << "Удаленное число: " << pop(head);  break;  case 4:  clear(head);  case 5:  cout << "--Запись в файл--";  cout << endl;  write(head);  break;  case 6:  cout << "--Чтение из файла--";  cout << endl;  read(head);  break;  case 7:  cout << "Количество повторяющихся элементов " << task(head);  break;  }  } while (choice != 0);  cout << "~~Выход~~"; return;  }  ***Header.h***  #pragma once  struct Stack  {  int number;  Stack\* next;  };  void show(Stack\*);  int pop(Stack\*);  void push(int, Stack\*\*);  void write(Stack\*);  void read(Stack\*);  void clear(Stack\*);  int task(Stack\*);  void menu(); | #include <iostream>  #include <fstream>  #include "Header.h"  using namespace std;  void menu()  {  cout << "\n\n1. Ввод нового числа\n";  cout << "2. Вывести стек\n";  cout << "3. Удаление верхнего элемента\n";  cout << "4. Очистить стек\n";  cout << "5. Запись в файл\n";  cout << "6. Чтение списка из файла\n";  cout << "7. Найти количество повторяющихся элементов\n";  }  void push(int number, Stack\*\* head)  {  Stack\* newStack = new Stack;  newStack->number = number;  newStack->next = \*head;  \*head = newStack;  cout << "\n--Число введено успешно--";  }  int pop(Stack\* head)  {  Stack\* temp = head;  int a = temp->number;  head = head->next;  delete temp;  return a;  }  void show(Stack\* head)  {  while (head != NULL)  {  cout << head->number;  cout << endl;  head = head->next;  }  }  void clear(Stack\* head)  {  while (head != NULL)  {  Stack\* temp = head;  head = head->next;  delete temp;  }  }  void write(Stack\* head)  {  ofstream fwrite("write.txt");  if (fwrite.fail())  {  cout << "\n --Ошибка открытия файла--";  exit(1);  }  else {  while (head != NULL)  {  fwrite << head->number << " ";  head = head->next;  }  }  fwrite.close();  }  void read(Stack\* head)  {  int value;  ifstream fread("read.txt");  if (fread.fail())  {  cout << "\n --Ошибка открытия файла--";  exit(1);  }  else {  while (!fread.eof())  {  fread >> value;  push(value, &head);  }  }  fread.close();  }  int task(Stack\* head)  { int count = 0;  Stack\* b = head;  Stack\* a = head;  b = b->next;  while (a != NULL){  b = a->next;  while (b != NULL)  {  if (head->number == b->number)  {  count++;  break; }  b = b->next; }  a = a->next;  }  return count;  } |

//9 лабораторная работа

**//*Создать очередь для массива целых (положительных и отрицательных) чисел и функции для ввода, вывода и удаления элементов очереди. При вводе чисел в очередь помещать только отрицательные элементы. Найти минимальный и максимальный элемент очереди. Вывести все элементы очереди и посчитать их количество.***

******

***лабораторная 9.cpp MyQueue.cpp***

|  |  |
| --- | --- |
| #include "myQueue.h"  #include <iostream>  #include <fstream>  #include <Windows.h>  using namespace std;  void menu()  {  cout << endl << "1.Ввод элемента в очередь" << endl;  cout << "2.Вывод очереди" << endl;  cout << "3.Удаление элементов очереди" << endl;  cout << "4.Максимальный и минимальный элемент в очереди" << endl;  cout << "0.Выход" << endl;  }  int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  int number, size, currentSize = 0;  Queue\* head = NULL, \* tail = NULL;  cout << "Введите нужный размер очереди" << endl;  cin >> size;  int choice;  do  {  menu();  cin >> choice;  switch (choice)  {  case 1:  {  cout << endl << "Заполнение очереди: " << endl;  if (currentSize < size)  {  cout << "Введите элемент: " << endl;  cin >> number;  if (number >= 0)  {  cout << "Ввод только для отрицательных элементов!" << endl;  }  else  {  enter\_new(&head, &tail, number);  currentSize++;  }  }  else  {  cout << "Превышен размер очереди!!" << endl;  }  break;  }  case 2:  {  cout << endl << "Вывод очереди" << endl;  show(&head);  break;  }  case 3:  {  cout << endl << "Удаление элемента из очереди" << endl;  if (del(&head, &tail) == 0)  {  cout << "Очередь пуста!!" << endl;  }  else  {  cout << "Элемент удален" << endl;  currentSize--;  }  break;  }  case 4:  {  cout << endl << "Поиск максимального и минимального элемента" << endl;  max\_min(&head);  break;  }  default:  exit(1);  }  } while (choice != 0);  } | #include <iostream>  #include "myQueue.h"  using namespace std;  void enter\_new(Queue\*\* head, Queue\*\* tail, int number)  {  Queue\* temp = new Queue;  if (\*head == NULL)  {  \*head = temp;  (\*head)->number = number;  \*tail = \*head;  }  else  {  temp->next = NULL;  temp->number = number;  (\*tail)->next = temp;  \*tail = temp;  }  }  int del(Queue\*\* head, Queue\*\* tail)  {  Queue\* temp;  if (head == NULL)  {  return 0;  }  else  {  temp = \*head;  \*head = (\*head)->next;  delete temp;  return 0;  }  }  void show(Queue\*\* head)  {  Queue\* temp = \*head;  int count = 0;  if (temp == NULL)  {  cout << "Очередь пуста!" << endl;  }  else  {  while (temp != NULL)  {  cout << temp->number << " ";  temp = temp->next;  count++;  }  cout << endl << "Элементов в очереди: " << count << endl;  }  }  void max\_min(Queue\*\* head)  {  Queue\* temp = \*head;  int max = temp->number, min = temp->number;  if (temp == NULL)  {  cout << "Очередь пуста!" << endl;  }  else  {  while (temp != NULL)  {  if (temp->number > max)  {  max = temp->number;  }  if (temp->number < min)  {  min = temp->number;  }  temp = temp->next;  }  }  cout << "Минимальный элемент в очереди - " << min << endl;  cout << "Максимальный элемент в очереди - " << max << endl;  } |

#pragma once

struct Queue

{

int number;

Queue\* next;

};

void enter\_new(Queue\*\*, Queue\*\*, int);

void show(Queue\*\*);

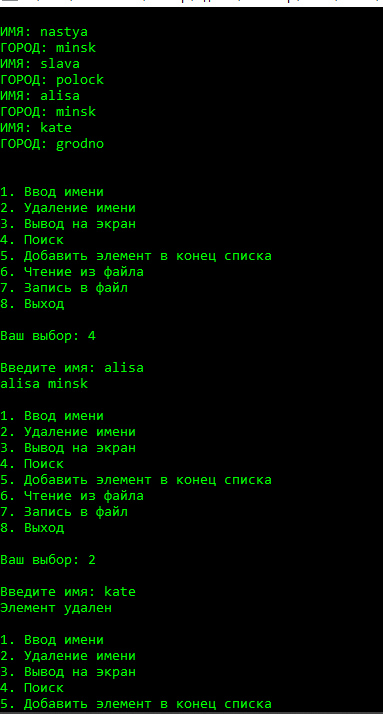
int del(Queue\*\*, Queue\*\*);

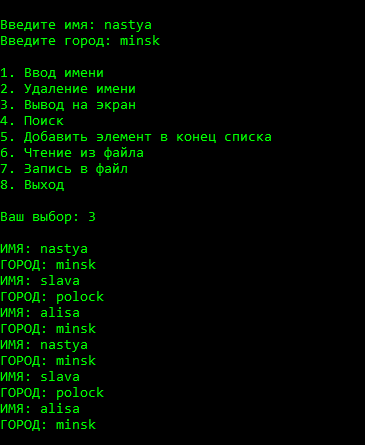
void max\_min(Queue\*\*);

//4 блок

Лабораторная работа 10

**addLBegin** – функция добавления в начало списка всех элементов некоторого списка **L**.





#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)

{

char s[80]; int c;

cout << endl;

cout << "1. Ввод имени" << endl;

cout << "2. Удаление имени" << endl;

cout << "3. Вывод на экран" << endl;

cout << "4. Поиск" << endl;

cout << "5. Добавить элемент в начало списка" << endl;

cout << "6. Чтение из файла" << endl;

cout << "7. Запись в файл" << 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)

{

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()

{

Address\* temp = new Address();

if (!temp)

{

cerr << "Ошибка выделения памяти памяти";

return NULL;

}

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 = NULL;

temp->prev = NULL;

return temp;

}

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

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

{

Address\* t = \*phead;

while (t)

{

cout << "ИМЯ: " << t->name << endl << "ГОРОД: " << 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)

cerr << "Имя не найдено" << endl;

else

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

}

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

void delet(char name[NAME\_SIZE], Address\*\* phead, Address\*\* plast) // Удаление имени

{

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

{

Address\* t;

FILE\* fp;

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

if (err)

{

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

exit(1);

}

while (\*phead) // Очистка списка(удаление элементов)

{

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

delete\* phead;

\*phead = \*plast;

}

\*phead = \*plast = NULL; // Первый и последний элемент списка - 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 addLBegin(Address \*\*phead, Address \*\*plast) { // Функция добавления элемента в начало списка

if (\*plast == NULL) { // Если список пустой, добавить туда элемент

insert(setElement(), phead, plast);

}

else { // В случае, если он не пустой

Address \*e = setElement();

e->next = \*phead;

(\*phead)->prev = e;

\*phead = e;

}

}

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

int main(void)

{

setlocale(0, "");

Address\* head = NULL;

Address\* last = NULL;

setlocale(LC\_CTYPE, "Rus");

int choice = 0;

while (choice != 8)

{

choice = menu();

switch (choice)

{

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();

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

cin.sync();

find(fname, &head);

}

break;

case 5:

addLBegin(&head, &last); // My program

break;

case 6: readFromFile(&head, &last);

break;

case 7: writeToFile(&head);

break;

default: exit(1);

}

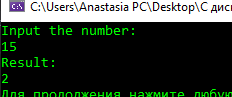
}

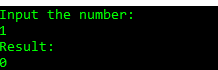
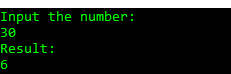
return 0;

}

**//**Лабораторная работа 11

Разработать программу, реализующую рекурсивную функцию подсчета количества всех положительных делителей натурального числа **n**.





#include <iostream>

using namespace std;

int f(int n) {

static int k = 0;

static int divisor = 2;

if (n / 2 < divisor) {

return k;

}

else if (n % divisor == 0) {

k++;

divisor++;

f(n);

}

else {

divisor++;

f(n);

}

}

int main() {

int n;

cout << "Input the number:\n";

cin >> n;

cout << "Result:\n";

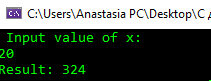
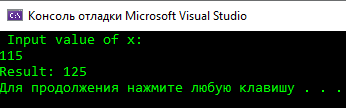
cout << f(n) << "\n";

system("pause");

return 0;

}

Разработать программу, реализующую рекурсивный алгоритм вычисления значений **S(x)** для любых целых не отрицательных значений **x**.



#include <iostream>

#include <cstdlib>

using namespace std;

unsigned int s(int x)

{

if (x > 100)

{

return x+10 ;

}

else

{

return s(s(x+4));

}

}

int main()

{

int x;

cout << " Input value of x:" << endl;

cin >> x;

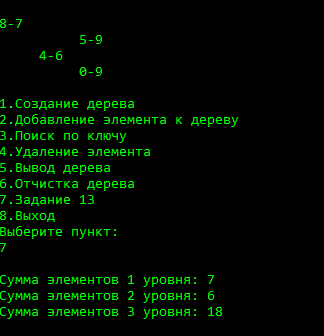
cout <<"Result: "<< s(x) << endl;

system("PAUSE");

}

//Лабораторная работа 12

Дан указатель **p1** на корень непустого дерева. Написать функцию, в которой для каждого из уровней данного дерева, начиная с нулевого, вывести сумму значений вершин, находящихся на этом уровне. Считать, что глубина дерева не превосходит 10.



#include <iostream>

#include <Windows.h>

using namespace std;

struct Tree

{

int key;

int data;

Tree\* left;

Tree\* right;

};

Tree\* createTree(Tree\*);

Tree\* createList(int, int);

Tree\* insertList(Tree\*);

Tree\* searchKey(Tree\*, int);

Tree\* deleteKey(Tree\*, int);

void showTree(Tree\*, int);

void delTree(Tree\*);

int var\_13(Tree\*, int\*, int);

void menu();

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Tree\* root = NULL, \* add;

int enteredKey, level;

int counter[15], i = 0, range;

float summ, amount;

float\* psumm = &summ, \* pamount = &amount;

int listSumm, listAmount;

int\* plistSumm = &listSumm, \* plistAmount = &listAmount;

int choice;

do

{

menu();

cin >> choice;

switch (choice)

{

case 1:

root = createTree(root);

if (root == NULL)

{

cout << endl << "Дерево уже создано" << endl;

}

break;

case 2:

if (root == NULL)

{

cout << endl << "Дерева не существует" << endl;

}

else

{

insertList(root);

}

break;

case 3:

if (root == NULL)

{

cout << endl << "Дерева не существует" << endl;

}

else

{

cout << endl << "Введите ключ элемента, который необходимо найти: " << endl;

cin >> enteredKey;

add = searchKey(root, enteredKey);

if (add == NULL)

{

cout << endl << "Элемент с введеным ключом не найден" << endl;

}

else

{

cout << endl << "Число, найденное по ключу : " << add->data << endl;

}

}

break;

case 4:

if (root == NULL)

{

cout << endl << "Дерева не существует" << endl;

}

else

{

cout << endl << "Введите ключ элемента, который необходимо удалить" << endl;

cin >> enteredKey;

if (!deleteKey(root, enteredKey));

{

cout << endl << "Элемент с веденным ключом не найден!" << endl;

}

}

break;

case 5:

level = 0;

add = root;

if (root == NULL)

{

cout << endl << "Дерева не существует" << endl;

}

else

{

showTree(add, level);

}

break;

case 6:

if (root == NULL)

{

cout << endl << "Дерева не существует" << endl;

}

else

{

add = root;

delTree(add);

root = NULL;

cout << endl << "Удаление завершено" << endl;

}

break;

case 7:

add = root;

i = 0;

for (int j = 0; j <= 14; j++)

{

counter[j] = 0;

}

range = var\_13(add, counter, i);

cout << endl;

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

{

cout << "Сумма элементов " << j + 1 << " уровня: " << counter[j] << endl;

}

break;

default:

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

cout << "2.Добавление элемента к дереву" << endl;

cout << "3.Поиск по ключу" << endl;

cout << "4.Удаление элемента" << endl;

cout << "5.Вывод дерева" << endl;

cout << "6.Отчистка дерева" << endl;

cout << "7.Задание 13" << endl;

cout << "8.Выход" << endl;

cout << "Выберите пункт: " << endl;

}

Tree\* createTree(Tree\* root)

{

int key, data;

if (root == NULL)

{

cout << "Введите ключ корня дерева: " << endl;

cin >> key;

cout << "Введите данные корня дерева: " << endl;

cin >> data;

root = createList(key, data);

return root;

}

else

{

return NULL;

}

}

Tree\* createList(int key, int data)

{

Tree\* temp = new Tree;

temp->key = key;

temp->data = data;

temp->left = temp->right = NULL;

return temp;

}

Tree\* insertList(Tree\* root)

{

int key, data;

bool find = false;

cout << endl << "Ввод элементов в дерево (введите ключ '-1' для остановки ввода) " << endl;

while (1)

{

Tree\* temp = root, \* prev = temp;

cout << endl << "Введите ключ:" << endl;

cin >> key;

if (key < 0)

{

cout << endl << "Остановка ввода" << endl;

break;

}

else

{

while (temp != NULL && !find)

{

prev = temp;

if (key == temp->key)

{

find = true;

cout << endl << "Совпадение ключей (ошибка)" << endl;

return NULL;

}

else

{

if (key < temp->key)

{

temp = temp->left;

}

else

{

temp = temp->right;

}

}

}

if (!find)

{

cout << endl << "Введите данные" << endl;

cin >> data;

temp = createList(key, data);

if (key < prev->key)

{

prev->left = temp;

}

else

{

prev->right = temp;

}

}

}

}

}

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

{

Tree\* temp = root;

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

{

if (key < temp->key)

{

temp = temp->left;

}

else

{

temp = temp->right;

}

}

if (temp == NULL)

{

return NULL;

}

if (key == temp->key)

{

return temp;

}

}

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

{

Tree\* del, \* prevDel, \* repl, \* prevRepl;

del = root;

prevDel = NULL;

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

{

prevDel = del;

if (key < del->key)

{

del = del->left;

}

else

{

del = del->right;

}

}

if (del == NULL)

{

return NULL;

}

if (del->right == NULL)

{

repl = del->left;

}

else

{

if (del->left == NULL)

{

repl = del->right;

}

else

{

prevRepl = del;

repl = del->left;

while (repl->right != NULL)

{

prevRepl = repl;

repl = repl->right;

}

if (prevRepl == del)

{

repl->right = del->right;

}

else

{

repl->right = del->right;

prevRepl->right = repl->left;

repl->left = prevRepl;

}

}

}

if (del = root)

{

root = repl;

}

else

{

if (del->key < prevDel->key)

{

prevDel->left = repl;

}

else

{

prevDel->right = repl;

}

}

cout << endl << "Удален элемент с ключом " << del->key << " и значением " << del->data << endl;

delete del;

}

void showTree(Tree\* temp, int level)

{

if (temp != NULL)

{

showTree(temp->right, level + 1); //вывод правого поддерева

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

{

cout << " ";

}

cout << temp->key << "-";

cout << temp->data;

showTree(temp->left, level + 1); //вывод левого поддерева

}

else

{

cout << endl;

}

}

void delTree(Tree\* root) //Очистка дерева

{

if (root != NULL)

{

delTree(root->left);

delTree(root->right);

delete root;

}

}

int var\_13(Tree\* temp, int\* counter, int i)

{

if (temp != NULL)

{

if (i == 0)

{

counter[i] = temp->data;

var\_13(temp->right, counter, i + 1);

var\_13(temp->left, counter, i + 1);

}

else

{

counter[i] += temp->data;

var\_13(temp->right, counter, i + 1);

var\_13(temp->left, counter, i + 1);

}

}

else

{

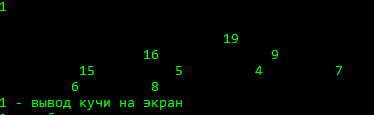
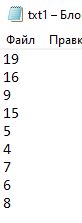
return i;

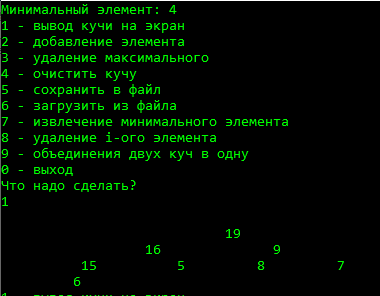
}

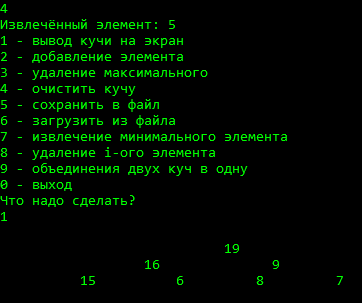
}

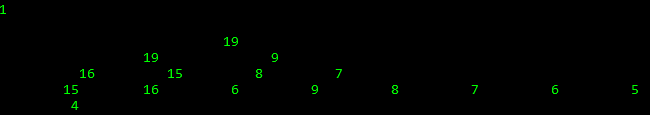
//5 БЛОК

//Лабораторная 13









|  |
| --- |
| #include "Heap.h"  ОСНОВНОЙ МОДУЛЬ 5 БЛОК.CPP  #include <iostream>  #include <fstream>  using namespace std;  heap::CMP cmpAAA(void\* a1, void\* a2) // функция сравнения 2х элементов типа ААА  {  #define A1 ((AAA\*)a1)  #define A2 ((AAA\*)a2)  heap::CMP rc = heap::EQUAL; // объявляем переменную типа CMP, то есть результат сравнения двух структур (=двух элементов кучи)  if (A1->x > A2->x) // если значение первого элемента больше значения второго  rc = heap::GREAT; // записываем в переменную, что первый элемент больше 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  #undef A1  }  bool BuildHeap(char\* FileName, heap::Heap& h) // построение кучи из файла, char \*FileName - имя файла, heap::Heap& h - куча, на которую берётся указатель  {  bool rc = true;  int n;  ifstream inFile;  inFile.open(FileName);  if (!inFile)  {  cout << "Невозможно открыть файл" << endl;  exit(1);  }  cout << " Исходные данные" << endl;  while (inFile >> n)  {  int\* a = new int;  \*a = n; // переписываем прочитанный элемент из буфера в выделенную память  h.Insert((void\*)a);  cout << \*a << endl;  }  inFile.close();  return rc;  }  void SaveHeap(heap::Heap& h, char\* FileName)  {  ofstream outFile(FileName);  if (!outFile)  {  std::cout << "Ошибка открытия выходного файла" << std::endl;  return;  }  int\* a;  for (int u = 0; u < h.Size; u++)  {  a = (int\*)h.Storage[u];  outFile << \*a;  outFile << endl;  }  outFile.close();  }  void main()  {  setlocale(LC\_ALL, "rus");  char buf[255];  int k;  heap::Heap h1 = heap::Create(30, cmpAAA), h2 = heap::Create(30, cmpAAA);  int choise;  AAA\* a;  for (;;)  {  cout << "1 - вывод кучи на экран" << endl;  cout << "2 - добавление элемента" << endl;  cout << "3 - удаление максимального" << endl;  cout << "4 - очистить кучу" << endl;  cout << "5 - сохранить в файл" << endl;  cout << "6 - загрузить из файла" << endl;  cout << "7 - извлечение минимального элемента" << endl;  cout << "8 - удаление i-ого элемента" << endl;  cout << "9 - объединения двух куч в одну" << endl;  cout << "0 - выход" << endl;  cout << "Что надо сделать?" << endl;  cin >> choise;  switch (choise)  {  case 0: exit(0);  case 1: h1.Scan(0);  break;  case 2:  a = new AAA;  cout << "Введите элемент " << endl;  cin >> k;  a->x = k;  h1.Insert(a);  break;  case 3:  a = (AAA\*)h1.ExtractMax();  if (a != NULL)  cout << "Максимальный элемент: " << a->x << endl;  else cout << "Куча пуста." << endl;  break;  case 4: h1.DeleteHeap();  break;  case 5:  cout << "Введите имя файла" << endl;  cin >> buf;  SaveHeap(h1, buf);  break;  case 6:  h1.DeleteHeap();  cout << "Введите имя файла" << endl;  cin >> buf;  BuildHeap(buf, h1);  break;  case 7:  a = (AAA\*)h1.ExtractMin();  if (a != NULL)  cout << "Минимальный элемент: " << a->x << endl;  else cout << "Куча пуста." << endl;  break;  case 8:  cout << "Введите номер элемента " << endl;  cin >> choise;  a = (AAA\*)h1.ExtractI(choise);  if (a != NULL)  cout << "Извлечённый элемент: " << a->x << endl;  else cout << "Куча пуста." << endl;  break;  case 9:  cout << "Введите имя файла" << endl;  cin >> buf;  h2.DeleteHeap();  BuildHeap(buf, h2);  h1.Union(h2);  break;  default:  cout << endl << "Введена неверная команда!"; cout << endl;  }  }  } |
| Модуль с функциями Heap.cpp  #include "Heap.h"  #include <iostream>  #include <iomanip>  using namespace std;  void AAA::Print()  {  std::cout << x;  }  int AAA::GetPriority()  {  return x;  }  namespace heap  {  Heap Create(int maxsize, CMP(\*f)(void\*, void\*)) // функция создания кучи с заданным максимальным размером и функцией сравнения двух элементов  {  return \*(new Heap(maxsize, f)); // создаём новую кучу и возвращаем её (по указытелю на выделенную память получаем кучу и возвращаем её)  }  int Heap::Left(int ix) // функция получения номера левого потомка элемента ix  {  return (2 \* ix + 1 >= Size) ? -1 : (2 \* ix + 1); // если номер левого потомка выходит за пределы текущего количества элементов в куче, то возвращаем -1, иначе возвращаем номер левого элемента  }  int Heap::Right(int ix) // функция получения номера правого потомка элемента ix  {  return (2 \* ix + 2 >= Size) ? -1 : (2 \* ix + 2); // если номер правого потомка выходит за пределы текущего количества элементов в куче, то возвращаем -1, иначе возвращаем номер правого элемента  }  int Heap::Parent(int ix) // функция получения номера родительского элемента ix  {  return (ix + 1) / 2 - 1; // возврат номера родительского элемента  }  void Heap::Swap(int i, int j) // функция перестановки двух элементов  {  void\* buf = Storage[i]; // запоминаем i-тый элемент массивы в буфере buf (Storage - массив элементов произвольного типа)  Storage[i] = Storage[j]; // на место i-того элемента записываем j-тый  Storage[j] = buf; // на место j-того элемента записываем то, что созранили в buf (i-тый элемент)  }  void Heap::Heapify(int ix) // функция формирования кучи в массиве (ix - номер текущего элемента)  {  int l = Left(ix), r = Right(ix), irl = ix; // irl = ix - переменная для хранения номера нибольшего из трёх элементов (теущего, левого и парвого)  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) // функция вставки элемента в кучу  {  int i;  if (!isFull())  {  i = Size;  Storage[i] = x;  Size++;  while (i > 0 && isLess(Storage[Parent(i)], Storage[i])) // пока не дойдём до вершины и пока родительский элемент меньше вставленного  {  Swap(Parent(i), i);  i = Parent(i); // новой позицией вставленного элемента становится позиция родительского элемента  }  }  }  void\* Heap::ExtractMax()  {  void\* rc = NULL;  if (!isEmpty())  {  rc = Storage[0]; // возвращаемый элемент равен нулевому элементу массива  Storage[0] = Storage[Size - 1]; // на место нулевого элемента массива записываем последний элемент  Size--;  Heapify(0); // перестраиваем кучу начиная с кучи  }  Heapify(0);  return rc;  }  void Heap::Scan(int i) // функция вывода кучи  {  int probel = 20;  std::cout << '\n';  if (Size == 0)  std::cout << "Куча пуста";  for (int u = 0, y = 0, level = 1; u < Size; u++)  {  cout << setw(probel + 10) << setfill(' ');  ((AAA\*)Storage[u])->Print();  if (u == y) // если мы дошли доканца текущего уровня (y-номер последнего элемента в текущем уровне)  {  level++;  cout << '\n';  if (y == 0) // если текущий уровень нулевой  y = 2; // то номер последнего элемента в следующем первом уровне = 2  else y = pow(2, level) - 2; // иначе расчитываем номер последнего элемента для следующего уровня (level- номер текущего уровня)  }  probel /= 2;  }  cout << '\n';  }  void Heap::DeleteHeap()  {  if (!isEmpty())  {  Size = 0;  this->~Heap();  }  }  void\* Heap::ExtractMin()  {  void\* rc = NULL;  if (Size > 0)  {  int j = Size / 2; // номер минамального элемента  for (int i = Size / 2; i < Size; i++)  // Size/2 - это номер первого элемента в последнем слое, Size - количество элементов в куче  {  if (Compare(Storage[i], Storage[j]) == LESS)  // если и-тый элемент меньше минимального (Storage[j])  {  j = i; // запоминаем номер найденного минимального элемента  }  }  rc = Storage[j]; // значение минимального элемента  Swap(j, Size - 1); // меняем местами последний элемент с найденным  Size--; // уменьшаем размер кода  }  Heapify(0);  return rc; // возвращаем значение найденного минимального элемента  }  // функция удаление i-ого элемента  void\* Heap::ExtractI(int i)  {  void\* rc = NULL;  if (Size > 0 && i >= 0 && i < Size)  {  rc = Storage[i];  Swap(i, Size - 1); // переставляем местами удаляемый элемент и последний  Size--;  }  Heapify(0);  return rc;  }  // функция объединения двух куч в одну  void Heap::Union(Heap& h2)  {  while (!h2.isEmpty()) // пока не опустеет вторая куча  Insert(h2.ExtractMax());  // извлекаем элемент из второй кучи и вставляем в первую  }  } |
| ЗАГОЛОВОЧНЫЙ ФАЙЛ  #pragma once  struct AAA  {  int x;  void Print();  int GetPriority();  };  namespace heap  {  enum CMP { LESS = -1, EQUAL = 0, GREAT = 1 };  struct Heap  {  int Size;  int MaxSize;  void\*\* Storage;  CMP(\*Compare)(void\*, void\*); // функция сравнения двух элементов  Heap(int maxsize, CMP(\*f)(void\*, void\*)) // конструктор кучи заданного размера и с заданной функцией сравнения элементов  {  Size = 0;  Storage = new void\* [MaxSize = maxsize];  Compare = f;  };  Heap(int maxsize, CMP(\*f)(void\*, void\*), void\* x[]) // конструктор кучи заданного размера с заданной функцией сравнения элементов и содержащие заданные элементы х  {  Size = 0;  Storage = x;  MaxSize = maxsize;  Compare = f;  };  int Left(int ix); // функция, возвращающая номер левого элемента от текущего (от заданного) ix  int Right(int ix); // функция, возвращающая номер правого элемента от текущего (от заданного) ix  int Parent(int ix); // функция, возвращающая номер родительского элемента от текущего (от заданного) ix  bool isFull() // функция проверки полноты кучи  {  return (Size >= MaxSize); // возвращает результат сравнения текущего размера с максимальным  };  bool isEmpty() // функция проверки кучи на пустоту  {  return (Size <= 0); // возвращает результат сравнения текущего размера с нулём  };  bool isLess(void\* x1, void\* x2) // функция сравнения является ли элемент х1 меньше элемента х2  {  return Compare(x1, x2) == LESS; // вызов функции сравнения для элементов х1 и х2 и возврат сравнения результат функции с элементом перечисления LESS  };  bool isGreat(void\* x1, void\* x2) // функция сравнения является ли элемент х1 больше элемента х2  {  return Compare(x1, x2) == GREAT;// вызов функции сравнения для элементов х1 и х2 и возврат сравнения результат функции с элементом перечисления GREAT  };  bool isEqual(void\* x1, void\* x2) // функция проверки равенства элементов х1 и х2  {  return Compare(x1, x2) == EQUAL;// вызов функции сравнения для элементов х1 и х2 и возврат сравнения результат функции с элементом перечисления EQUAL  };  void Swap(int i, int j); // функция перестановки двух элементов с номерами i и j  void Heapify(int ix); // функция формирования кучи из массива  void Insert(void\* x); // функция вставки элемента х в кучу  void\* ExtractMax(); // функция извлечения максимаьного элемента, т.е. вершины кучи  void DeleteHeap(); // функция очистки кучи  void Scan(int i); // функция вывода кучи начиная с заданного элемента  void\* ExtractMin(); // функция удаления минимального элемента  void\* ExtractI(int i); // функция удаление i-ого элемента  void Union(Heap&); // функция объединения двух куч в одну (Heap& - функция принамает в качестве аргумента кучу и берёт указатель на неё)  };  Heap Create(int maxsize, CMP(\*f)(void\*, void\*)); // функция для создания кучи с заданным размером и функцией сравнения (void\* - данные произвольного типа)  }; |

//ЛАБОРАТОРНАЯ 14

Лабораторная 15

Лабораторная 16