ГУО “БГУИР”

Факультет компьютерного проектирования

Кафедра Инженерной психологии и эргономики

Отчёт по

Лабораторной работе №8

СРЕДА ПРОГРАММИРОВАНИЯ VISUAL C++. ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

Подготовил:

Студент гр.110101

Ладутько Я.Д.

Проверила:

Семижон Е.А.

Минск 2021

*Цель работы:* изучить правила создания и обработки данных структурного типа с использованием файлов.

Написать программу обработки файла типа запись, содержащую следующие пункты меню: «Создание», «Просмотр», «Коррекция» (добавление новых данных или редактирование старых), «Решение индивидуального задания».

Каждая запись должна содержать следующую информацию о студентах:

– фамилия и инициалы;

– год рождения;

– номер группы;

– оценки за семестр: по физике, математике, информатике, химии;

– средний балл.

Организовать ввод исходных данных, средний балл рассчитать по введенным оценкам.

Содержимое всего файла и результаты решения индивидуального задания записать в текстовый файл.

Вариант 15.



Код:

# //Lab8.cpp

#include "Lab8.h"

#include "LadiarusSpace.h"

int amount\_of\_lines;

short current\_pointer\_y;

bool is\_exit;

const string INDIVIDUAL\_TASK\_FILE\_NAME = "Individual Task.txt";

COORD curr\_coord;

CONSOLE\_SCREEN\_BUFFER\_INFO info;

string curr\_in\_file, curr\_out\_file;

const int OUT\_WEIGTHS[] = {19, 5, 8, 5, 5, 5, 5, 7};

ifstream fin;

ofstream fout;

fstream finout;

HANDLE hCon;

template <typename T>

void myQSort(vector<T>& v, int beg, int end, bool(\*comp)(T\*, T\*))

{

int curr\_main\_idx = end;

if (end <= beg)

{

return;

}

int i = beg;

while (i < curr\_main\_idx)

{

if (comp(&v[i], &v[curr\_main\_idx]))

{

swap(v[i], v[curr\_main\_idx - 1]);

swap(v[curr\_main\_idx - 1], v[curr\_main\_idx]);

curr\_main\_idx--;

}

else

{

i++;

}

}

myQSort(v, beg, curr\_main\_idx - 1, comp);

myQSort(v, curr\_main\_idx + 1, end, comp);

}

void movePointer(short y)

{

SetConsoleCursorPosition(hCon, {0, current\_pointer\_y});

cout << ' ';

current\_pointer\_y = y;

SetConsoleCursorPosition(hCon, { 0, y });

cout << "\033[0;32m#\033[0;m";

SetConsoleCursorPosition(hCon, { 0, y });

}

void resetCursor()

{

SetConsoleCursorPosition(hCon, { 0, current\_pointer\_y });

cout << ' ';

SetConsoleCursorPosition(hCon, { curr\_coord });

}

StudentData getStudentDataByIdx(int idx)

{

string s;

fin.close();

fin.open(curr\_in\_file);

for(int i = 0; i < idx-1; i++)

{

fin.ignore(INT\_MAX, '\n');

}

getline(fin, s);

return getStudentDataByString(s);

}

StudentData getStudentDataByString(string &s)

{

StudentData sd;

vector <string> v = splitStr(s, ' ');

sd.name = v[0] + " " + v[1];

sd.year = stoi(v[2]);

sd.number = stoi(v[3]);

for (int i = 4; i < 8; i++)

{

sd.marks[i - 4] = stod(v[i]);

}

sd.average\_mark = stod(v[8]);

return sd;

}

bool isNameValid(string s)

{

vector <string> v = splitStr(s, ' ');

if (v.size() != 2)

{

cout << "\nНеверный ввод, введите через пробел фамилию и инициалы\n";

return false;

}

for (char c : s)

{

if (!(isalpha(c) || (c >= 'а' && c <= 'я') || (c >= 'А' && c <= 'Я')) && c != ' ' && c != '.')

{

return false;

}

}

if (s.length() > 19)

{

cout << "\nМаксимальная длина " << OUT\_WEIGTHS[0] << " символов\n";

return false;

}

return true;

}

bool isDoubleValid(string &s, double a, double b)

{

int num\_of\_dots = 0;

if (s[0] == '.')

s = '0' + s;

if (s.length() > 15)

return false;

for (auto c : s)

if (c == '.')

num\_of\_dots++;

if (num\_of\_dots > 1)

return false;

double x = stod(s);

if (x < a || x > b)

return false;

return true;

}

bool isIntValid(string s, int a, int b)

{

if (s.length() > 10)

return false;

long long x = stol(s);

if (x < a || x > b)

return false;

return true;

}

void viewOptions()

{

cout << "Create File - 1\nChoose File - 2\nAdd - 3\nChange - 4\nSort - 5\nIndividual Task - 6\n";

chooseOption({createFileSelect, chooseFileSelect, addStringSelect, editRowSelect, sortSelect, individualTaskSelect});

}

bool isFileExists(string s)

{

finout.close();

finout.open(s, ios::in);

if (finout.is\_open())

{

finout.close();

return true;

}

else

{

finout.close();

return false;

}

}

void viewFile()

{

system("cls");

cout << " " << centerString(curr\_in\_file, 67, '-') << "\n"

" | Имя | Год | Группа | Физ | Мат | Инф | Хим | Средн |\n"

" |-------------------|-----|--------|-----|-----|-----|-----|-------|\n";

string s;

fin.close();

fin.open(curr\_in\_file);

amount\_of\_lines = 0;

while (!fin.eof())

{

getline(fin, s);

vector <string>v = splitStr(s, ' ');

if (v.size() == 9)

{

cout << " ";

if (amount\_of\_lines % 2 != 0)

{

cout << "\033[48;5;235m"; // background color

}

cout << "|" << setw(OUT\_WEIGTHS[0]) << v[0] + " " + v[1] << "|";

for (int i = 2; i < v.size(); i++)

{

cout << setw(OUT\_WEIGTHS[i - 1]) << v[i] << '|';

}

cout << ' ' << "\033[34m" << amount\_of\_lines + 1 << "\033[m" << endl ;

}

if(!s.empty())

amount\_of\_lines++;

}

viewOptions();

}

void clearFileSelect()

{

bool b;

cout << "Вы точно хотите очистить этот файл? y/n\n";

input(b);

if (b)

clearFile(curr\_in\_file);

}

void clearFile(string s)

{

fin.close();

fin.open(s, ios::trunc);

fin.close();

fin.open(s);

}

void createFileSelect()

{

bool b;

string s;

cout << "Введите название файла, оканчивающееся на .txt\n";

do {

input(s, "/<>?|\*:\"\\");

if (s.substr(s.length() <= 4 || s.length() - 4, 4) != ".txt")

cout << "Неверный ввод\n\a";

} while (s.length() <= 4 || s.substr(s.length() - 4, 4) != ".txt");

if (isFileExists(s))

{

cout << "Файл существует, очистить его? y/n\n";

input(b);

if (b)

clearFile(s);

}

else

{

createFile(s);

}

cout << "Файл создан, открыть его? y/n\n";

input(b);

if(b)

{

chooseFile(s);

}

}

void createFile(string s)

{

finout.close();

finout.open(s, ios::out);

finout.close();

}

void chooseFileSelect()

{

string s;

cout << "Введите название файла, оканчивающееся на .txt\n";

do {

input(s, "/<>?|\*:\"\\");

if (s.length() <= 4 || s.substr(s.length() - 4, 4) != ".txt")

cout << "Неверный ввод\a\n";

} while (s.length() <= 4 || s.substr(s.length() - 4, 4) != ".txt");

if (!isFileExists(s))

{

bool b;

cout << "Файла не существует, создать его? y/n\n";

input(b);

if (b)

{

createFile(s);

}

}

else

{

chooseFile(s);

}

}

void chooseFile(string s)

{

if (s != INDIVIDUAL\_TASK\_FILE\_NAME)

{

fin.close();

fin.open(s);

curr\_in\_file = s;

}

else

{

cout << "Этот файл нельзя открывать";

}

}

void editRowSelect()

{

if (amount\_of\_lines == 0)

return;

int c = 0, y, selected\_idx = -1;

bool isInPointerMode = false;

cout << "Выберите строку, которую хотите изменить (Стрелка Вверх(Вниз), чтобы переместить указатель)\n";

cout << "Нажмите ESC, чтобы выйти из режима указателя и ввести номер строки\n";

GetConsoleScreenBufferInfo(hCon, &info);

curr\_coord = info.dwCursorPosition;

while (true)

{

int c = \_getch();

if (c == 224) {

switch (\_getch())

{

case KEY\_UP:

y = clamp(current\_pointer\_y - 1, 3, amount\_of\_lines + 2);

movePointer(y);

break;

case KEY\_DOWN:

y = clamp(current\_pointer\_y + 1, 3, amount\_of\_lines + 2);

movePointer(y);

break;

default:

;

}

}

else if(c == KEY\_ENTER) //если нажат Enter

{

selected\_idx = current\_pointer\_y - 2;

resetCursor();

break;

}

else if (c == 27)//если нажат ESC

{

resetCursor();

break;

}

}

if (selected\_idx == -1)

{

input(selected\_idx, 1, amount\_of\_lines);

}

StudentData sd = editStudentDataViaGetch(getStudentDataByIdx(selected\_idx));

editRow(selected\_idx, sd);

}

void editRow(int idx, StudentData sd)

{

string s, temp\_name = "TEMP\_FILE\_DO\_NOT\_TOUCH.txt";

fin.close();

fin.open(curr\_in\_file);

createFile(temp\_name);

fout.open(temp\_name);

for(int i = 0; i < idx-1; i++)

{

getline(fin, s);

fout << s << endl;

}

fin.ignore(INT\_MAX, '\n');

fout << sd.name << ' ';

fout << sd.year << ' ';

fout << sd.number << ' ';

fout << round(sd.marks[0] \* 10) / 10 << ' ';

fout << round(sd.marks[1] \* 10) / 10 << ' ';

fout << round(sd.marks[2] \* 10) / 10 << ' ';

fout << round(sd.marks[3] \* 10) / 10 << ' ';

fout << round(sd.average\_mark \* 10) / 10;

while(!fin.eof())

{

getline(fin, s);

fout << "\n" << s;

}

fin.close();

fout.close();

remove(curr\_in\_file.c\_str());

rename(temp\_name.c\_str(), curr\_in\_file.c\_str());

}

void sortSelect()

{

int c, a;

if (amount\_of\_lines == 0)

return;

vector<StudentData> v;

fin.close();

fin.open(curr\_in\_file);

string s;

while(!fin.eof())

{

getline(fin, s);

v.push\_back(getStudentDataByString(s));

}

cout << "Имя - 1\nГод - 2\nНомер - 3\nФизика- 4\nМатематика - 5\nИнформатика - 6\nХимия - 7\nСредняя - 8\nEXIT - 0\n";

input(c, 0, 8);

if (c == 0)

return;

cout << "От большего к меньшему - 1\nОт меньшего к большему - 2\n";

input(a, 1, 2);

bool reverse = a - 1;

switch(c)

{

case 1:

sort(v, reverse, [](StudentData\* sd1, StudentData\* sd2) ->bool

{

return sd1->name < sd2->name;

});

break;

case 2:

sort(v, reverse, [](StudentData\* sd1, StudentData\* sd2) ->bool

{

return sd1->year < sd2->year;

});

break;

case 3:

sort(v, reverse, [](StudentData\* sd1, StudentData\* sd2) ->bool

{

return sd1->number < sd2->number;

});

break;

case 4:

sort(v, reverse, [](StudentData\* sd1, StudentData\* sd2) ->bool

{

return sd1->marks[0] < sd2->marks[0];

});

break;

case 5:

sort(v, reverse, [](StudentData\* sd1, StudentData\* sd2) ->bool

{

return sd1->marks[1] < sd2->marks[1];

});

break;

case 6:

sort(v, reverse, [](StudentData\* sd1, StudentData\* sd2) ->bool

{

return sd1->marks[2] < sd2->marks[2];

});

break;

case 7:

sort(v, reverse, [](StudentData\* sd1, StudentData\* sd2) ->bool

{

return sd1->marks[3] < sd2->marks[3];

});

break;

case 8:

sort(v, reverse, [](StudentData\* sd1, StudentData\* sd2) ->bool

{

return sd1->average\_mark < sd2->average\_mark;

});

break;

default:;

}

}

void sort(vector<StudentData> v, bool reverse, bool(\*comp)(StudentData\*, StudentData\*))

{

myQSort<StudentData>(v, 0, v.size() - 1, comp);

if(reverse)

{

std::reverse(v.begin(), v.end());

}

fin.close();

fout.open(curr\_in\_file);

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

{

StudentData sd = v[i];

fout << sd.name << ' ';

fout << sd.year << ' ';

fout << sd.number << ' ';

fout << round(sd.marks[0] \* 10) / 10 << ' ';

fout << round(sd.marks[1] \* 10) / 10 << ' ';

fout << round(sd.marks[2] \* 10) / 10 << ' ';

fout << round(sd.marks[3] \* 10) / 10 << ' ';

fout << round(sd.average\_mark \* 10) / 10;

if (i < v.size() - 1)

fout << endl;

}

fout.close();

}

void cmdDeleteLeft(int &idx, string &s)

{

if (idx <= 0)

return;

idx = clamp(idx, 1, s.length());

idx --;

char c;

s = s.substr(0, idx) + s.substr(idx + 1, s.length() - idx);

TO\_ROW(s.length() + 1);

cout<< " "; //в n столбец

TO\_ROW(0);

cout << s;

TO\_ROW(idx+1);

}

void cmdDeleteRight(int& idx, string& s)

{

if (idx >= s.length())

return;

idx = clamp(idx, 0, s.length()-1);

char c;

s = s.substr(0, idx) + s.substr(idx + 1, s.length() - idx);

TO\_ROW(s.length()+1);

cout << " ";

TO\_ROW(0);

cout << s;

TO\_ROW(idx + 1);

}

void cmdAdd(int& idx, string& s, int c)

{

string temp\_s = s.substr(0, idx);

s = temp\_s + static\_cast<char>(c) + s.substr(idx, s.length() - idx);

TO\_ROW(0);

cout << s;

TO\_ROW(idx + 2);

idx++;

}

StudentData editStudentDataViaGetch(StudentData sd)

{

cout << "Имя\n";

cout << sd.name;

sd.name = editString(sd.name, dataType::String);

CLEAR\_LINE();

UP\_LINE();

CLEAR\_LINE();

TO\_ROW(0);

cout << "Год\n";

cout << sd.year;

sd.year = stoi(editString(to\_string(sd.year), dataType::Int, 1900, 2021));

CLEAR\_LINE();

UP\_LINE();

CLEAR\_LINE();

TO\_ROW(0);

cout << "Номер\n";

cout << sd.number;

sd.number = stoi(editString(to\_string(sd.number), dataType::Int, 0, 99999999));

CLEAR\_LINE();

UP\_LINE();

CLEAR\_LINE();

TO\_ROW(0);

for (int i = 0; i < 4; i++)

{

switch(i)

{

case 0:

cout << "Физика\n";

break;

case 1:

cout << "Математика\n";

break;

case 2:

cout << "Информатика\n";

break;

case 3:

cout << "Химия\n";

break;

}

cout << to\_string(sd.marks[i]);

sd.marks[i] = stod(editString(to\_string(sd.marks[i]), dataType::Double, 0, 10));

CLEAR\_LINE();

UP\_LINE();

CLEAR\_LINE();

TO\_ROW(0);

}

return sd;

}

string editString(string curr\_s, dataType data\_type, double min\_val, double max\_val)

{

int curr\_pos = 0;

TO\_ROW(0);

while (true)

{

int c = \_getch();

if (c == 224) {

switch (\_getch())

{

case KEY\_LEFT:

if (curr\_pos > 0)

{

cout << "\033[1D"; //влево

curr\_pos--;

}

break;

case KEY\_RIGHT:

if (curr\_pos < curr\_s.length())

{

cout << "\033[1C"; //вправо

curr\_pos++;

}

break;

case KEY\_DELETE:

cmdDeleteRight(curr\_pos, curr\_s);

break;

}

}

else if (c == KEY\_ENTER)

{

if (!curr\_s.empty())

{

switch (data\_type)

{

case String:

if (isNameValid(curr\_s))

return curr\_s;

break;

case Double:

if (isDoubleValid(curr\_s, min\_val, max\_val))

return curr\_s;

break;

case Int:

if (isIntValid(curr\_s, min\_val, max\_val))

{

return curr\_s;

}

break;

}

TO\_ROW(0);

cout << curr\_s;

TO\_ROW(curr\_pos + 1);

}

cout << "\a"; // звук

}

else if (c == KEY\_BACKSPACE)

{

cmdDeleteLeft(curr\_pos, curr\_s);

}

else

{

bool is\_ok = false;

int k = c - 255; // char почему-то signed ¯\\_(ツ)\_/¯

switch (data\_type)

{

case String:

if (NAME\_INPUT\_CHECK)

{

is\_ok = true;

}

break;

case Double:

if (DOUBLE\_INPUT\_CHECK)

{

is\_ok = true;

}

break;

case Int:

if (INT\_INPUT\_CHECK)

{

is\_ok = true;

}

break;

}

if(is\_ok)

cmdAdd(curr\_pos, curr\_s, c);

}

}

}

void chooseOption(vector<void(\*)()> args)

{

int choice;

cout << "EXIT - 0\n";

input(choice, 0, args.size());

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

{

if (i + 1 == choice)

args[i]();

}

if(choice == 0)

{

is\_exit = true;

}

}

void addStringSelect()

{

string s;

vector <string> v;

StudentData \*sd = new StudentData();

bool is\_valid = false;

cout << "Введите фамилию и через пробел инициалы\n";

cin.ignore(1000, '\n');

while (!is\_valid)

{

int sum = 0;

getline(cin, s);

is\_valid = isNameValid(s);

}

sd->name = s;

cout << "Введите год рождения\n";

input(sd->year, 1900, 2021);

cout << "Введите номер группы\n";

input(sd->number, 0, 100000000-1);

cout << "Введите оценку по физике\n";

input(sd->marks[0], 0, 10);

cout << "Введите оценку по математике\n";

input(sd->marks[1], 0, 10);

cout << "Введите оценку по информатике\n";

input(sd->marks[2], 0, 10);

cout << "Введите оценку по химии\n";

input(sd->marks[3], 0, 10);

sd->average\_mark = (sd->marks[0] + sd->marks[1] + sd->marks[2] + sd->marks[3]) / 4;

addString(curr\_in\_file, sd);

}

void addString(string path, const StudentData\* sd)

{

fin.close();

fout.close();

fout.open(path, ios::app);

if(amount\_of\_lines!=0)

fout << endl;

fout << sd->name << ' ';

fout << sd->year << ' ';

fout << sd->number << ' ';

fout << round(sd->marks[0] \* 10) / 10 << ' ';

fout << round(sd->marks[1] \* 10) / 10 << ' ';

fout << round(sd->marks[2] \* 10) / 10 << ' ';

fout << round(sd->marks[3] \* 10) / 10 << ' ';

fout << round(sd->average\_mark \* 10) / 10;

fout.close();

delete sd;

}

void individualTaskSelect()

{

if (amount\_of\_lines == 0)

return;

string s;

double average;

createFile(INDIVIDUAL\_TASK\_FILE\_NAME);

fout.open(INDIVIDUAL\_TASK\_FILE\_NAME);

cout << "Введите средний балл\n";

input(average, 0, 10);

system("cls");

cout << centerString(INDIVIDUAL\_TASK\_FILE\_NAME, 67, '-') << "\n"

"| Имя | Год | Группа | Физ | Мат | Инф | Хим | Средн |\n"

"|-------------------|-----|--------|-----|-----|-----|-----|-------|\n";

fin.close();

fin.open(curr\_in\_file);

int i = 0;

while (!fin.eof())

{

if(i % 2 == 0)

cout << "\033[48;5;235m"; // background color

getline(fin, s);

StudentData sd = getStudentDataByString(s);

if(sd.average\_mark > average)

{

cout << "|" << setw(OUT\_WEIGTHS[0]) << sd.name;

cout << "|" << setw(OUT\_WEIGTHS[1]) << sd.year;

cout << "|" << setw(OUT\_WEIGTHS[2]) << sd.number;

cout << "|" << setw(OUT\_WEIGTHS[3]) << sd.marks[0];

cout << "|" << setw(OUT\_WEIGTHS[4]) << sd.marks[1];

cout << "|" << setw(OUT\_WEIGTHS[5]) << sd.marks[2];

cout << "|" << setw(OUT\_WEIGTHS[6]) << sd.marks[3];

cout << "|" << setw(OUT\_WEIGTHS[7]) << sd.average\_mark << "|";

cout << "\033[m" << endl;

fout << sd.name;

fout << " " << sd.year;

fout << " " << sd.number;

fout << " " << sd.marks[0];

fout << " " << sd.marks[1];

fout << " " << sd.marks[2];

fout << " " << sd.marks[3];

fout << " " << sd.average\_mark << " ";

fout << endl;

i++;

}

}

cout << "\033[m" << endl;

system("pause");

fout.close();

}

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

{

setlocale(LC\_ALL, "Russian");

setlocale(LC\_NUMERIC, "en-US"); // устанавливаю locale для цифр и знаков, чтобы работал stod

SetConsoleCP(1251);

hCon = GetStdHandle(STD\_OUTPUT\_HANDLE); //Присваивание дескриптора консоли

COORD size{ 100, 50 };

SetConsoleScreenBufferSize(hCon, size);//размер буфера

fin.open("input.txt");

curr\_in\_file = "input.txt";

while (true)

{

viewFile();

if(is\_exit)

break;

}

}

# //Lab8.h

#pragma once

#define KEY\_UP 72

#define KEY\_DOWN 80

#define KEY\_LEFT 75

#define KEY\_RIGHT 77

#define KEY\_ENTER 13

#define KEY\_DELETE 83

#define KEY\_BACKSPACE 8

#define NAME\_INPUT\_CHECK (c >= 'a' && c <= 'z' || c-255 >= 'а' && c-255 <= 'я' || c >= 'A' && c <= 'Z' || c-255 >= 'А' && c-255 <= 'Я' || c == ' ')

#define DOUBLE\_INPUT\_CHECK (c >= '0' && c <= '9' || c == '.')

#define INT\_INPUT\_CHECK (c >= '0' && c <= '9')

#define TO\_ROW(num) (cout<<"\033["+to\_string(num)+"G")

#define CLEAR\_LINE() (cout<<"\033[2K")

#define UP\_LINE() (cout << "\033[1A")

enum dataType

{

String,

Double,

Int

};

#include <iostream>

#include <string>

#include <algorithm>

#include <cmath>

#include <conio.h>

#include <cstdlib>

#include <fstream>

#include <iomanip>

#include <list>

#include <vector>

#include <Windows.h>

using namespace std;

struct StudentData

{

double marks[4]{}, average\_mark = 0;

string name;

int year = 0, number = 0;

};

void moveChar(int x, int y);

void viewOptions();

bool isFileExists(string s);

void viewFile();

void clearFileSelect();

void clearFile(string s);

void createFileSelect();

void createFile(string s);

void chooseFileSelect();

void chooseFile(string s);

void chooseOption(vector<void(\*)()> args);

void editRowSelect();

void editRow(int idx, StudentData sd);

void sortSelect();

void sort(vector<StudentData> v, bool reverse, bool(\*comp)(StudentData\*, StudentData\*));

void addStringSelect();

void addString(string path, const StudentData\* sd);

StudentData editStudentDataViaGetch(StudentData sd);

string editString(string curr\_s, dataType data\_type, double min\_val = 0, double max\_val = 0);

StudentData getStudentDataByIdx(int idx);

StudentData getStudentDataByString(string &s);

bool isNameValid(string s);

bool isDoubleValid(string s);

bool isIntValid(string s , int a, int b);

void individualTaskSelect();

//LadiarusSpace.cpp

#include "LadiarusSpace.h"

template <class T>

void input(T& a)

{

while (true)

{

cin >> a;

if (cin.fail() || (cin.peek() != '\n' && cin.peek() != ' ' && cin.peek() != '\t'))

{

cin.clear();

cin.ignore(10000, '\n');

cout << "Неверный ввод" << endl;

}

else

{

return;

}

}

}

void input(int& x, int a, int b)

{

while (true)

{

input(x);

if (x < a || x > b)

{

cout <<

"Неверный ввод\n"

"Введите число на промежутке от " << a << " до " << b << endl;

}

else

return;

}

}

void input(double& x, double a, double b)

{

while (true)

{

input(x);

if (x < a || x > b)

{

cout <<

"Неверный ввод\n"

"Введите число на промежутке от " << a << " до " << b << endl;

}

else

return;

}

}

void input(bool& b)

{

char c;

while (true)

{

input(c);

if (c != 'y' && c != 'n')

{

cout <<

"Неверный ввод\n"

"Введите y или n\n";

}

else

{

b = (c == 'y' ? true : false);

return;

}

}

}

void input(string &s, string exclude)

{

while (true)

{

bool is\_ok = true;

cin >> s;

for(auto c : exclude)

{

if (s.find(c) != -1)

is\_ok = false;

}

if (cin.fail() || (cin.peek() != '\n') || !is\_ok)

{

cin.clear();

cin.ignore(10000, '\n');

cout << "Неверный ввод, нельзя использовать символы: \""<< exclude << "\"" << endl;

}

else

{

return;

}

}

}

vector<string> splitStr(string s, char delim)

{

string out;

vector<string> v;

int min\_length\_ind = -1;

bool is\_out\_null = true;

s += delim;

for (char i : s)

{

if (i != delim)

{

if (is\_out\_null)

{

is\_out\_null = false;

}

out += i;

}

else if (!is\_out\_null)

{

v.push\_back(out);

out = "";

is\_out\_null = true;

}

}

return v;

}

int clamp(int n, int lower, int upper) {

return max(lower, min(n, upper));

}

string centerString(const string& s, int width, char fill\_with)

{

string ret\_str;

int x = (width - s.length())/2;

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

{

ret\_str += fill\_with;

}

ret\_str += s;

for (int i = ret\_str.length() - 1; i < width; i++)

{

ret\_str += fill\_with;

}

return ret\_str;

}

template <typename T>

void myQSort(vector<T>& v, int beg, int end, bool(\*comp)(T\*, T\*))

{

int curr\_main\_idx = end;

if (end - beg == 2)

{

if (comp(&(v[beg]), &(v[end])))

{

swap(v[beg], v[end]);

}

return;

}

int i = beg;

while (i < end)

{

if (comp(&v[i], &v[end]))

{

swap(v[i], v[curr\_main\_idx - 1]);

swap(v[curr\_main\_idx - 1], v[curr\_main\_idx]);

curr\_main\_idx--;

}

else

{

i++;

}

}

myQSort(v, beg, curr\_main\_idx - 1, comp);

myQSort(v, curr\_main\_idx + 1, end, comp);

}

# //LadiarusSpace.h

#pragma once

#include <string>

#include <iostream>

#include <vector>

#include "Lab8.h"

template <class T>

void input(T& a);

void input(int& x, int a, int b);

void input(double& x, double a, double b);

void input(bool& b);

void input(string &s, string exclude);

vector<string> splitStr(string s, char delim);

int clamp(int n, int lower, int upper);

string centerString(const string& s, int width, char fill\_with);

Вывод: изучил правила создания и обработки данных структурного типа с использованием файлов.

