ГУО “БГУИР”

Факультет компьютерного проектирования

Кафедра Инженерной психологии и эргономики

Отчёт по

Лабораторной работе №9

СРЕДА ПРОГРАММИРОВАНИЯ VISUAL C++. ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

Подготовил:

Студент гр.110101

Ладутько Я.Д.

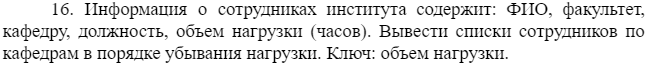
Проверила:

Семижон Е.А.

Минск 2022

***Цель работы:*** изучить способы сортировки и поиска в массивах структур и файлах.

Вариант 16.



Код:

**///LadiarusSpace.h**

#pragma once

#include <string>

#include <iostream>

#include <vector>

#include "Lab9.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);

**///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;

}

**///Lab9.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' || k >= 'а' && k <= 'я' || c >= 'A' && c <= 'Z' || k >= 'А' && k <= '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 EmployeeData

{

double workload;

string name, faculty, department, position;

};

enum class SortType

{

QSort,

ChooseSort

};

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, EmployeeData sd);

void sortSelect();

void sort(vector<EmployeeData> &v, SortType st, bool(\*comp)(EmployeeData\*, EmployeeData\*));

void addStringSelect();

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

EmployeeData editStudentDataViaGetch(EmployeeData sd);

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

EmployeeData getStudentDataByIdx(int idx);

EmployeeData getEmployeeDataByString(string &s);

bool isNameValid(string s);

bool isDoubleValid(string s);

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

void individualTaskSelect();

void deleteSelect();

void deleteRows(vector<int>& v);

**///Lab9.cpp**

#include "Lab9.h"

#include "LadiarusSpace.h"

int amount\_of\_lines;

short current\_pointer\_y;

bool is\_exit;

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

const string TEMP\_FILE\_NAME = "TEMP\_FILE\_DO\_NOT\_TOUCH.txt";

COORD curr\_coord;

CONSOLE\_SCREEN\_BUFFER\_INFO info;

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

const int OUT\_WEIGTHS[] = {19, 11, 9, 11, 10};

ifstream fin;

ofstream fout;

fstream finout;

HANDLE hCon;

int binSearch(vector<EmployeeData> &v, double x)

{

int l = 0, r = v.size() - 1, c;

while (l < r)

{

c = (l + r) / 2;

if (x > v[c].workload)

l = c + 1;

else

r = c;

}

if (abs(v[l].workload - x) < 1e-5)

return l;

else

return -1;

}

void searchInFileSelect()

{

double x;

cout << "Введите объем нагрузки для поиска: ";

input(x);

string s;

fin.close();

fin.open(curr\_in\_file);

amount\_of\_lines = 0;

while (!fin.eof())

{

getline(fin, s);

EmployeeData ed = getEmployeeDataByString(s);

if (abs(ed.workload - x) < 1e-5)

{

system("cls");

cout << centerString("SEARCH\_IN\_FILE", 65, '-') << "\n"

"| Имя | Факультет | Кафедра | Должность | Нагрузка |\n"

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

cout << '|' << setw(OUT\_WEIGTHS[0]) << ed.name << '|';

cout << setw(OUT\_WEIGTHS[1]) << ed.faculty << '|';

cout << setw(OUT\_WEIGTHS[2]) << ed.department << '|';

cout << setw(OUT\_WEIGTHS[3]) << ed.position << '|';

cout << setw(OUT\_WEIGTHS[4]) << ed.workload << '|' << endl;

cout << "\033[m";

system("pause");

return;

}

}

cout << "Ничего не найдено\n";

system("pause");

}

template <typename T>

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

{

int i, j, m;

for (i = beg; i <= end - 1; i++)

{

m = i;

for (j = i + 1; j <= end; j++)

{

if (!comp(&v[j], &v[m]))

m = j;

}

swap(v[m], v[i]);

}

}

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);

}

bool intComp(int\* a, int\* b)

{

return \*a > \*b;

}

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 });

}

EmployeeData 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 getEmployeeDataByString(s);

}

EmployeeData getEmployeeDataByString(string &s)

{

EmployeeData sd;

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

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

sd.faculty = v[2];

sd.department = v[3];

sd.position = v[4];

sd.workload = stod(v[5]);

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 << "1 - Add\n2 - Change\n3 - Delete\n4 - Search\n5 - Sort\n6 - Individual Task\n";

chooseOption({addStringSelect, editRowSelect, deleteSelect, searchInFileSelect, 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, 65, '-') << "\n"

" | Имя | Факультет | Кафедра | Должность | Нагрузка |\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() == 6)

{

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 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);

}

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

editRow(selected\_idx, sd);

}

void editRow(int idx, EmployeeData sd)

{

string s;

fin.close();

fin.open(curr\_in\_file);

createFile(TEMP\_FILE\_NAME);

fout.open(TEMP\_FILE\_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.faculty << ' ';

fout << sd.department << ' ';

fout << sd.position << ' ';

fout << sd.workload;

while(!fin.eof())

{

getline(fin, s);

fout << "\n" << s;

}

fin.close();

fout.close();

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

rename(TEMP\_FILE\_NAME.c\_str(), curr\_in\_file.c\_str());

}

void sortSelect()

{

int c, a;

if (amount\_of\_lines == 0)

return;

vector<EmployeeData> v;

fin.close();

fin.open(curr\_in\_file);

string s;

while(!fin.eof())

{

getline(fin, s);

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

}

cout << "Имя - 1\nФакультет - 2\nКафедра - 3\nДолжность- 4\nНагрузка - 5\nEXIT - 0\n";

input(c, 0, 8);

if (c == 0)

return;

cout << "Выберите способ сортировки:\n1 - Быстрая\n2 - Прямого выбора\n";

int d;

input(d, 1, 2);

SortType st;

switch (d)

{

case 1:

st = SortType::QSort;

break;

case 2:

st = SortType::ChooseSort;

break;

}

switch(c)

{

case 1:

sort(v, st, [](EmployeeData\* sd1, EmployeeData\* sd2) ->bool

{

return sd1->name > sd2->name;

});

break;

case 2:

sort(v, st, [](EmployeeData\* sd1, EmployeeData\* sd2) ->bool

{

return sd1->faculty < sd2->faculty;

});

break;

case 3:

sort(v, st, [](EmployeeData\* sd1, EmployeeData\* sd2) ->bool

{

return sd1->department < sd2->department;

});

break;

case 4:

sort(v, st, [](EmployeeData\* sd1, EmployeeData\* sd2) ->bool

{

return sd1->position < sd2->position;

});

break;

case 5:

sort(v, st, [](EmployeeData\* sd1, EmployeeData\* sd2) ->bool

{

return sd1->workload < sd2->workload;

});

break;

default:;

}

}

void sort(vector<EmployeeData>& v, SortType st, bool(\*comp)(EmployeeData\*, EmployeeData\*))

{

switch (st)

{

case SortType::QSort:

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

break;

case SortType::ChooseSort:

myChooseSort<EmployeeData>(v, 0, v.size() - 1, comp);

break;

}

fin.close();

fout.open(curr\_in\_file);

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

{

EmployeeData sd = v[i];

fout << sd.name << ' ';

fout << sd.faculty << ' ';

fout << sd.department << ' ';

fout << sd.position << ' ';

fout << sd.workload;

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++;

}

EmployeeData editStudentDataViaGetch(EmployeeData sd)

{

cout << "Имя\n";

cout << sd.name;

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

CLEAR\_LINE();

UP\_LINE();

CLEAR\_LINE();

TO\_ROW(0);

cout << "Факультет\n";

cout << sd.faculty;

sd.faculty = editString(sd.faculty, dataType::String, false);

CLEAR\_LINE();

UP\_LINE();

CLEAR\_LINE();

TO\_ROW(0);

cout << "Кафедра\n";

cout << sd.department;

sd.department = editString(sd.department, dataType::String, false);

CLEAR\_LINE();

UP\_LINE();

CLEAR\_LINE();

TO\_ROW(0);

cout << "Должность\n";

cout << sd.position;

sd.position = editString(sd.position, dataType::String, false);

CLEAR\_LINE();

UP\_LINE();

CLEAR\_LINE();

TO\_ROW(0);

cout << "Нагрузка\n";

cout << to\_string(sd.workload);

sd.workload = stod(editString(to\_string(sd.workload), dataType::Double, false, 0, 300));

CLEAR\_LINE();

UP\_LINE();

CLEAR\_LINE();

TO\_ROW(0);

return sd;

}

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

{

int curr\_pos = 0;

TO\_ROW(0);

while (true)

{

bool isA;

int c = \_getch();

if (c == 224) {

if(\_kbhit()) // если ожидает ввода (не 'а')

{

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;

}

continue;

}

}

else if (c == KEY\_ENTER)

{

if (!curr\_s.empty())

{

switch (data\_type)

{

case String:

if (isName && isNameValid(curr\_s))

{

return curr\_s;

}

else if (curr\_s.length() > 1 && curr\_s.find(' ') == string::npos)

{

return curr\_s;

}

else

{

cout << "\nНеверный ввод\n";

}

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

continue;

}

else if (c == KEY\_BACKSPACE)

{

cmdDeleteLeft(curr\_pos, curr\_s);

continue;

}

bool is\_ok = false;

int k;

if('р' < 0 ) k = c - 256; // 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;

EmployeeData \*sd = new EmployeeData();

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 факультет\n";

getline(cin, s);

sd->faculty = s;

cout << "Введите кафедру\n";

getline(cin, s);

sd->department = s;

cout << "Введите должность\n";

getline(cin, s);

sd->position = s;

cout << "Введите объем нагрузки\n";

input(sd->workload, 0, 300);

addString(curr\_in\_file, sd);

}

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

{

fin.close();

fout.close();

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

if(amount\_of\_lines!=0)

fout << endl;

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

fout << sd->faculty << ' ';

fout << sd->department << ' ';

fout << sd->position << ' ';

fout << sd->workload;

fout.close();

delete sd;

}

void individualTaskSelect()

{

if (amount\_of\_lines == 0)

return;

string s;

cout << "Выберите способ сортировки:\n1 - Быстрая\n2 - Прямого выбора\n";

int c;

input(c, 1, 2);

SortType st;

switch (c)

{

case 1:

st = SortType::QSort;

break;

case 2:

st = SortType::ChooseSort;

break;

}

system("cls");

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

"| Имя | Факультет | Кафедра | Должность | Нагрузка |\n"

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

vector<EmployeeData> v;

vector<string>departmentsVector;

fin.close();

fin.open(curr\_in\_file);

while (!fin.eof())

{

getline(fin, s);

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

}

sort(v, st, [](EmployeeData\* sd1, EmployeeData\* sd2) ->bool

{

return sd1->workload < sd2->workload;

});

for(EmployeeData &ed : v)

{

if(find(departmentsVector.begin(), departmentsVector.end(), ed.department) == departmentsVector.end())

{

departmentsVector.push\_back(ed.department);

}

}

int i = 0;

for (string s : departmentsVector)

{

for (EmployeeData& ed : v)

{

if (s == ed.department)

{

if (i % 2 != 0)

{

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

}

cout << '|' << setw(OUT\_WEIGTHS[0]) << ed.name << '|';

cout << setw(OUT\_WEIGTHS[1]) << ed.faculty << '|';

cout << setw(OUT\_WEIGTHS[2]) << ed.department << '|';

cout << setw(OUT\_WEIGTHS[3]) << ed.position << '|';

cout << setw(OUT\_WEIGTHS[4]) << ed.workload << '|' <<endl;

cout << "\033[m";

i++;

}

}

}

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

cout << "1 - Бинарный поиск\n0 - EXIT\n";

input(c, 0, 1);

if (c == 0)

return;

sort(v, st, [](EmployeeData\* sd1, EmployeeData\* sd2) ->bool

{

return sd1->workload > sd2->workload;

});

double x;

cout << "Введите объем нагрузки для поиска: ";

input(x);

int f = binSearch(v, x);

if (f == -1)

{

cout << "Ничего не найдено\n";

}

else

{

system("cls");

cout << centerString("INDIVIDUAL\_TASK", 65, '-') << "\n"

"| Имя | Факультет | Кафедра | Должность | Нагрузка |\n"

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

EmployeeData& ed = v[f];

cout << '|' << setw(OUT\_WEIGTHS[0]) << ed.name << '|';

cout << setw(OUT\_WEIGTHS[1]) << ed.faculty << '|';

cout << setw(OUT\_WEIGTHS[2]) << ed.department << '|';

cout << setw(OUT\_WEIGTHS[3]) << ed.position << '|';

cout << setw(OUT\_WEIGTHS[4]) << ed.workload << '|' << endl;

cout << "\033[m";

}

system("pause");

fout.close();

}

void deleteSelect()

{

fin.close();

cout << "Введите через пробел номера сотрудников, которых хотите удалить\n";

string s;

int a = 0;

bool isValid;

cin.ignore();

getline(cin, s);

vector<int> linesRaw;

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

for(string x : v)

{

isValid = true;

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

{

if(x[i] < '0' || x[i] > '9')

{

cout << "Неверный ввод: " + x + " будет пропущено" << endl;

isValid = false;

break;

}

}

if(isValid)

{

a = stoi(x);

if(a > amount\_of\_lines || a < 1)

{

cout << "Неверный ввод, выход за пределы: " + x + " будет пропущено" << endl;

}

else

{

linesRaw.push\_back(a);

}

}

}

if (linesRaw.empty())

return;

myQSort<int>(linesRaw, 0, linesRaw.size() - 1, intComp);

vector<int> lines;

lines.push\_back(linesRaw[0]);

for(int i = 1; i < linesRaw.size(); i++)

{

if(linesRaw[i] != linesRaw[i - 1])

{

lines.push\_back(linesRaw[i]);

}

}

int arrIdx = 0, fileIdx = 1;

fin.open(curr\_in\_file);

string g, k;

cout << "Будут удалены:\n";

while(!fin.eof() && arrIdx < lines.size())

{

if (fileIdx == lines[arrIdx])

{

fin >> g;

fin >> k;

cout << g + " " + k << endl;

arrIdx++;

}

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

fileIdx++;

}

bool b;

cout << "Вы уверены, что хотите удалить эти записи? y/n\n";

input(b);

if(b)

deleteRows(lines);

}

void deleteRows(vector<int>& v)

{

string s;

v.push\_back(-1); //Ограничитель

fin.close();

fin.open(curr\_in\_file);

createFile(TEMP\_FILE\_NAME);

fout.open(TEMP\_FILE\_NAME);

int fileIdx = 1;

int arrIdx = 0;

bool is\_first\_write = true;

while (!fin.eof())

{

if(fileIdx != v[arrIdx])

{

getline(fin, s);

if(!is\_first\_write) //если не первая строка

{

fout << "\n";

}

fout << s;

is\_first\_write = false;

}

else

{

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

arrIdx++;

}

fileIdx++;

}

fin.close();

fout.close();

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

rename(TEMP\_FILE\_NAME.c\_str(), curr\_in\_file.c\_str());

}

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);

SetConsoleOutputCP(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;

}

}

***Вывод****:* изучил способы сортировки и поиска в массивах структур и файлах.

