Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра «Вычислительных методов и программирования»

ОТЧЕТ

По лабораторным работам №4-5

«Сортировка массивов. Поиск по ключу в одномерном массиве»

Выполнила:

Студентка гр. №820605

ФИО

Вариант № 9

Проверил:

Ассистент кафедры ВМИП

Беспалов С.А.

Минск 2019

1. **Задание:**

   В. *Дополнить программу, написанную при выполнении лабораторной работы №9, функциями упорядочения массива структур по неубыванию заданного ключа. Результат вывести на экран.*

   В9. Ключ: время отправления поезда. Методы сортировки: **QuickSort** и **метод Шелла**.

2. **Задание:**

   В. *Дополнить программу, написанную при выполнении лабораторной работы №10 функциями поиска элементов по ключу в массиве структур. Найти элемент с заданным ключом указанным методом поиска (для упрощения предполагается, что в массиве присутствует не более одного такого элемента). Если элемент не найден, то вывести соответствующее сообщение.*

   В9. Вывести на экран пункт назначения поезда, который отправляется в 11 часов. Методы поиска**: линейный с барьером** и **двоичный**.

Текст программы:

```
#include "pch.h"
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cstdio>
#include <string.h>
#include <conio.h>
#include <iomanip>
#include <io.h>

using namespace std;

int n = 0;
char name[30] = "";
FILE *trains;
struct Trn
{
	char deptime[6];
	char destination[50];
	int plcs;
} train, *list, *trainslist;

void input();
void read();
void outputsh();
void outputqs();
void outinf();
void qs(int, int, Trn*);
void lsearch();
void bsearch();
int menu();
char *fname();

int main()
{
	while (true)
	{
		switch (menu())
		{
		case 1: input(); break;
		case 2: read(); break;
		case 3: outputsh(); break;
		case 4: outputqs(); break;
```

```
                case 5: lsearch(); break;
                case 6: bsearch(); break;
                case 7: outinf(); break;
                case 8: return 0;
                default: cout << "This case doesn't exist" << endl;
                }
                puts("Press any key to continue");
                system("pause");
                system("cls");
        }
}

int menu()
{
        cout << "Choose:" << endl;
        cout << "1. Input data in the file." << endl;
        cout << "2. Read data from the file." << endl;
        cout << "3. Output result on screen. [Shells sortmethod]" << endl;
        cout << "4. Output result on screen. [Quicksort]" << endl;
        cout << "5. Linear search of train." << endl;
        cout << "6. Binar search of train." << endl;
        cout << "7. Output result in the file." << endl;
        cout << "8. Exit" << endl;
        int i; cin >> i;
        return i;
}

char* fname()
{
        if (strlen(name)) return name;
        cout << "Input file name" << endl;  cin >> name;
        return name;
}

void input()
{
        if ((trains = (fopen(fname(), "ab"))) == NULL)
                if ((trains = (fopen(fname(), "wb"))) == NULL)
                {
                        cout << "FILE CREATION ERROR" << endl; return;
                }
        char next;
        do
        {
                cout << "Input departure time: "; cin >> train.deptime;
                cout << "Input destination: "; cin >> train.destination;
                cout << "Input number of free places: "; cin >> train.plcs;
                fwrite(&train, sizeof(Trn), 1, trains);
                cout << endl << "Add a new train? (y/n) ";
                cin >> next; cout << endl;
        } while (next == 'y');

        fclose(trains);
}

void read()
{
        if ((trains = (fopen(fname(), "rb"))) == NULL)
        {
                cout << "OPENNING ERROR" << endl; return;
        }
        n = _filelength(_fileno(trains)) / sizeof(Trn);
        list = new Trn[n];
        fread(list, sizeof(Trn), n, trains);
```

```
	cout << endl;
	for (int i = 0; i < n; i++)
		cout << list[i].deptime << " Destination - " << list[i].destination << "
Number of free places - " << list[i].plcs << endl;
	cout << endl;
	delete[] list;
	fclose(trains);
}

void outputsh()
{
	if ((trains = fopen(fname(), "rb")) == NULL)
	{
		cout << "OPENNING ERROR" << endl; return;
	}
	n = _filelength(_fileno(trains)) / sizeof(Trn);
	trainslist = new Trn[n];
	for (int i = 0; i < n; i++)
		fread(&trainslist[i], sizeof(Trn), 1, trains);
	Trn extra;
	int j;
	for (int step = 3; step > 0; step--)
		for (int i = step; i < n; i++)
		{
			extra = trainslist[i];
			for (j = i - step; j >= 0 && strcmp(extra.deptime,
trainslist[j].deptime) == -1; j -= step) trainslist[j + step] = trainslist[j];
			trainslist[j + step] = extra;
		}
	for (int i = 0; i < n; i++)
	{
		if (!strcmp(trainslist[i].destination, "Brest"))
			cout << trainslist[i].deptime << " Number of free places: " <<
trainslist[i].plcs << endl;
	}
	fclose(trains);
	delete[] trainslist;
}

void outputqs()
{
	if ((trains = fopen(fname(), "rb")) == NULL)
	{
		cout << "OPENNING ERROR" << endl; return;
	}
	n = _filelength(_fileno(trains)) / sizeof(Trn);
	trainslist = new Trn[n];
	for (int i = 0; i < n; i++)
		fread(&trainslist[i], sizeof(Trn), 1, trains);
	qs (0, n - 1, trainslist);
	for (int i = 0; i < n; i++)
	{
		if (!strcmp(trainslist[i].destination, "Brest"))
			cout << trainslist[i].deptime << " Number of free places: " <<
trainslist[i].plcs << endl;
	}
	fclose(trains);
	delete[] trainslist;
}

void qs(int left, int right, Trn *list)
{
	int i = left; int j = right;
	Trn extra, middle;
```

```
	middle = list[(i + j) / 2];
	do
	{
		while (strcmp(middle.deptime, list[i].deptime)==1 && i < right) i++;
		while (strcmp(list[j].deptime, middle.deptime)==1 && j > left) j--;
		if (i <= j)
		{extra = list[i]; list[i] = list[j]; list[j] = extra; i++; j--;}

	} while (i <= j);
	if (left < j) qs(left, j, list);
	if (i < right) qs(i, right, list);
}

void lsearch()
{
	if ((trains = fopen(fname(), "rb")) == NULL)
	{
		cout << "OPENNING ERROR" << endl; return;
	}
	n = _filelength(_fileno(trains)) / sizeof(Trn);
	trainslist = new Trn[n+1];
	for (int i = 0; i < n; i++)
		fread(&trainslist[i], sizeof(Trn), 1, trains);
	puts ( "Input departure time of sought train");
	getchar();
	gets_s (trainslist[n].deptime);
	int i = 0;
	while (strcmp(trainslist[i].deptime, trainslist[n].deptime)) i++;
	if (i == n) cout << "This train doesn't exist" << endl;
	else cout << "Destination of this train: " << trainslist[i].destination << endl;
	delete[] trainslist;
	fclose(trains);
}

void bsearch()
{
	char deptime[6];
	if ((trains = fopen(fname(), "rb")) == NULL)
	{
		cout << "OPENNING ERROR" << endl; return;
	}
	n = _filelength(_fileno(trains)) / sizeof(Trn);
	trainslist = new Trn[n];
	for (int i = 0; i < n; i++)
		fread(&trainslist[i], sizeof(Trn), 1, trains);
	qs(0, n - 1, trainslist);
	cout << "Input departure time of sought train " << endl;
	getchar();
	gets_s(deptime);
	int i = 0, j = n - 1, m;
	while (i < j)
	{
		m = (i + j) / 2;
		if (strcmp(deptime, trainslist[m].deptime) == 1) i = m + 1; else j = m;
	}
	if ( !strcmp(trainslist[i].deptime,deptime)) cout << "Destination of this train: "
<< trainslist[i].destination << endl;
	else cout << "This train doesn't exist" << endl;
	delete[] trainslist;
	fclose(trains);
}

void outinf()
{
```

```
        char tfname[20];
        FILE *textfile;
        cout << "Input name of textfile: "; cin >> tfname;

        if ((textfile = fopen(tfname, "w")) == NULL)
        {
                cout << "TEXTFILE CREATION ERROR" << endl; return;
        }
        if ((trains = fopen(fname(), "rb")) == NULL)
        {
                cout << "OPENNING ERROR" << endl; return;
        }
        n = _filelength(_fileno(trains)) / sizeof(Trn);
        for (int i = 0; i < n; i++)
        {
                fread(&train, sizeof(Trn), 1, trains);
                if (!strcmp(train.destination, "Brest"))
                        fprintf(textfile, "%s, number of free places - %d\n", train.deptime,
  train.plcs);
        }
        fclose(trains);
        fclose(textfile);
}
```

Результат выполнения программы :

```
Choose:
1. Input data in the file.
2. Read data from the file.
3. Output result on screen. [Shells sortmethod]
4. Output result on screen. [Quicksort]
5. Linear search of train.
6. Binar search of train.
7. Output result in the file.
8. Exit
```

```
2

09.43 Destination - Brest Number of free places - 4
09.32 Destination - Brest Number of free places - 1
05.02 Destination - Brest Number of free places - 23
13.12 Destination - Brest Number of free places - 5
21.03 Destination - Brest Number of free places - 3
12.30 Destination - Moscow Number of free places - 9
04.21 Destination - Brest Number of free places - 10
18.04 Destination - Warsaw Number of free places - 84
11.00 Destination - Vilnius Number of free places - 27
```

```
3
04.21 Number of free places: 10
05.02 Number of free places: 23
09.32 Number of free places: 1
09.43 Number of free places: 4
13.12 Number of free places: 5
21.03 Number of free places: 3
```

```
4
04.21 Number of free places: 10
05.02 Number of free places: 23
09.32 Number of free places: 1
09.43 Number of free places: 4
13.12 Number of free places: 5
21.03 Number of free places: 3
```

```
5
Input departure time of sought train
11.00
Destination of this train: Vilnius
```

```
6
Input departure time of sought train
11.00
Destination of this train: Vilnius
```