**ЛАБОРАТОРНАЯ РАБОТА №2**

**«ПРОГРАММИРОВАНИЕ ОПЕРАЦИЙ НАД СТРУКТУРАМИ И**

**БИНАРНЫМИ ФАЙЛАМИ»**

**Вариант 9**

**2.1 Цель работы**

Исследование особенностей обработки бинарных файлов, хранящих

структурные типы данных.

**2.2 Вариант задания**

Описать структуру с именем TRAIN, содержащую следующие поля:

- порядковый номер;

- название пункта отправления;

- название пункта назначения;

- номер поезда;

- время отправления (часы и минуты).

Написать программу, выполняющую следующие действия с помощью функций:

- ввод с клавиатуры данных в файл, состоящий из элементов типа TRAIN;

- чтение данных из этого файла;

- корректировку данных в файле по номеру записи;

- вывод на экран информации о поезде, номер которого введен с клавиатуры, а если таких поездов нет, то выдать на дисплей соответствующее сообщение;

- отсортировать записи по номерам поездов.

**2.3 Ход работы**

Были разработаны структурные схемы для функций программы на языке C (рисунки В.1 – В.10 приложение В).

Была разработана программа на языке программирования С, которая решала поставленную задачу (листинг 2.1).

Листинг 2.1 – Код программы на языке С

#include <stdio.h>

#include <stdlib.h>

FILE\* file;

struct time {

int minutes;

int hours;

};

struct TRAIN {

int index;

char from[20];

char to[20];

int number;

struct time time;

};

struct time setTime();

void addTrain();

void openFile(char\* fileName);

void printAllTrains();

void printTrain (struct TRAIN train);

void printTrainByNumber(int number);

struct TRAIN setTrain();

void changeTrain(int index);

void sortTrains();

int main() {

int isActive = 1;

openFile("trainData.Dat");

int choice;

while (isActive) {

printf("1 - Add a train\n");

printf("2 - Print all trains\n");

printf("3 - Change train by index\n");

printf("4 - Print train by number\n");

printf("5 - Sort Trains\n");

printf("6 - Exit\n");

scanf("%d", &choice);

printf("\n");

switch (choice) {

case 1:

addTrain();

break;

case 2:

printAllTrains();

break;

case 3:

int index;

printf("Index: "); scanf("%d", &index);

changeTrain(index);

break;

case 4:

int number; scanf("%d", &number);

printTrainByNumber(number);

break;

case 5:

sortTrains();

break;

case 6:

isActive = 0;

break;

}

}

fclose(file);

return 0;

}

void openFile(char\* fileName) {

file = fopen(fileName, "rb+");

if (!file)

file = fopen(fileName, "wb+");

if (!file)

exit(1);

return;

}

void addTrain () {

fseek(file, 0, SEEK\_END);

struct TRAIN newTrain = setTrain();

fwrite(&newTrain, sizeof(struct TRAIN), 1, file);

}

struct TRAIN setTrain() {

struct TRAIN newTrain;

newTrain.index = ftell(file)/sizeof(struct TRAIN);

printf("From: "); scanf("%s", &newTrain.from);

printf("To: "); scanf("%s", &newTrain.to);

printf("Number: "); scanf("%d", &newTrain.number);

newTrain.time = setTime();

return newTrain;

}

struct time setTime() {

struct time time;

printf("Hours: "); scanf("%d", &time.hours);

printf("Minutes: "); scanf("%d", &time.minutes);

return time;

}

void printAllTrains() {

rewind(file);

struct TRAIN train;

while(fread(&train, sizeof(struct TRAIN), 1, file)) {

printTrain(train);

}

return;

}

void printTrain (struct TRAIN train) {

printf("Index: %d\n", train.index);

printf("From: %s\n", train.from);

printf("To: %s\n", train.to);

printf("Number: %d\n", train.number);

printf("Time: %02d:%02d\n", train.time.hours, train.time.minutes);

printf("\n");

return;

}

void printTrainByNumber(int number) {

struct TRAIN train;

rewind(file);

while (fread(&train, sizeof(struct TRAIN), 1, file)) {

if (train.number == number) {

printTrain(train);

return;

}

}

printf("No train :(\n");

return;

}

void changeTrain(int index) {

fseek(file, index\*sizeof(struct TRAIN), 0);

struct TRAIN newTrain = setTrain();

fwrite(&newTrain, sizeof(struct TRAIN), 1, file);

return;

}

void sortTrains() {

int numberOfINdexes = ftell(file)/sizeof(struct TRAIN);

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

fseek(file, sizeof(struct TRAIN)\*i, 0);

struct TRAIN tempTrain1, tempTrain2;

fread(&tempTrain1, sizeof(struct TRAIN), 1, file);

int iMin = i;

for (int j = i+1; j < numberOfINdexes; j++) {

fread(&tempTrain2, sizeof(struct TRAIN), 1, file);

if (tempTrain2.number < tempTrain1.number) {

iMin = j;

}

}

fseek(file, sizeof(struct TRAIN)\*iMin, 0);

fread(&tempTrain2, sizeof(struct TRAIN), 1, file);

tempTrain2.index += tempTrain1.index;

tempTrain1.index = tempTrain2.index - tempTrain1.index;

tempTrain2.index -= tempTrain1.index;

fseek(file, sizeof(struct TRAIN)\*iMin, 0);

fwrite(&tempTrain1, sizeof(struct TRAIN), 1, file);

fseek(file, sizeof(struct TRAIN)\*i, 0);

fwrite(&tempTrain2, sizeof(struct TRAIN), 1, file);

}

return;

}

Были проведены тесты программы (рисунки Г.1 – Г.2 приложение Г).

**2.4 Вывод**

В ходе лабораторной работы были исследованы особенности обработки бинарных файлов, хранящих структурные типы данных.

**Приложение В**

**Структурные схемы**

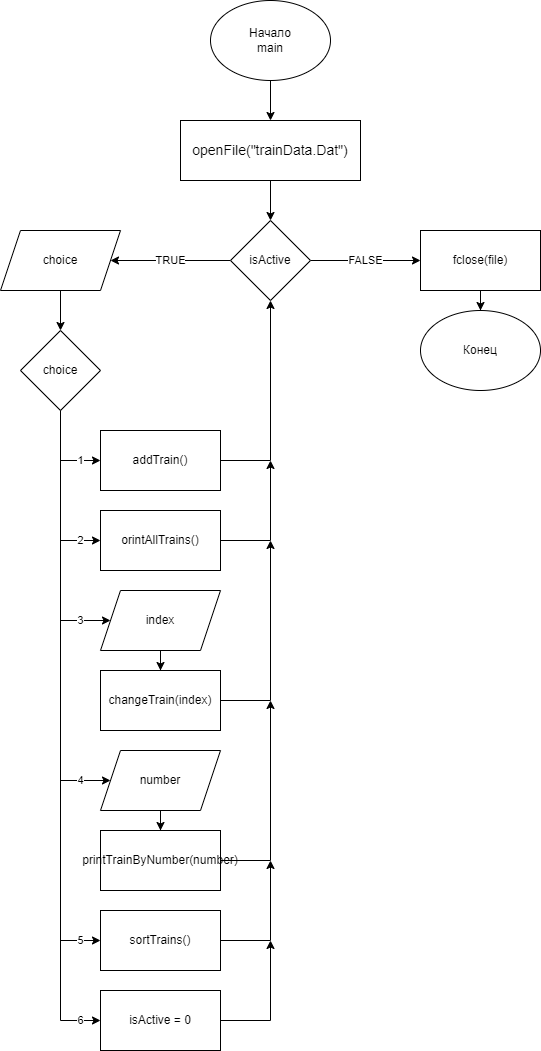


Рисунок В.1 – Структурная схема main

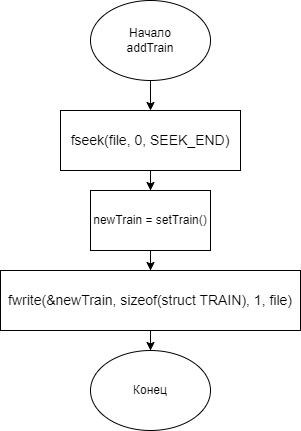


Рисунок В.2 – Структурная схема addTrain



Рисунок В.3 – Структурная схема changeTrain

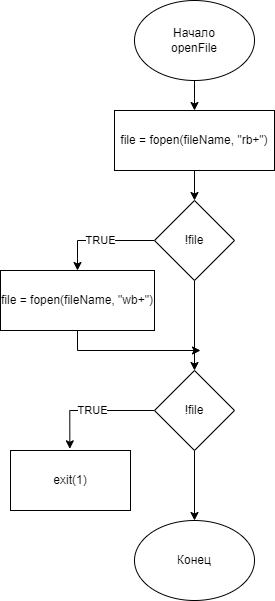


Рисунок В.4 – Структурная схема openFile

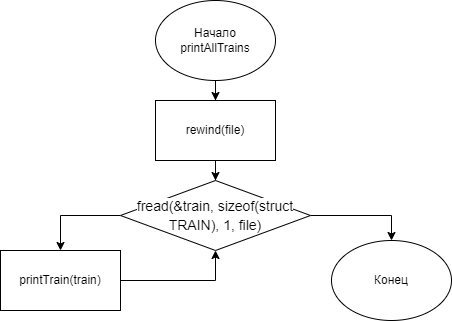


Рисунок В.5 – Структурная схема printAllTrains

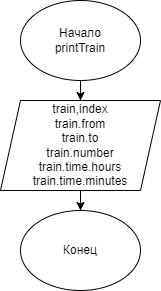


Рисунок В.6 – Структурная схема printTrain

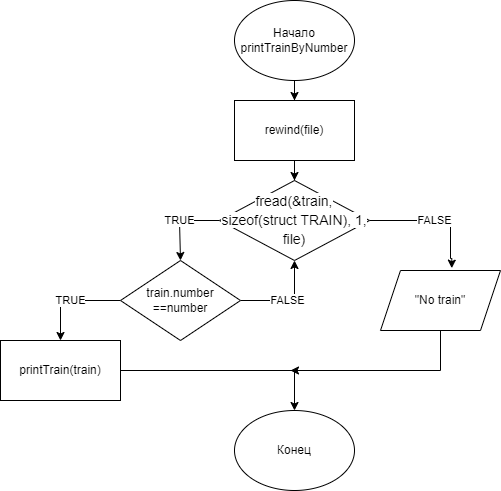


Рисунок В.7 – Структурная схема printTrainByNumber



Рисунок В.8 – Структурная схема setTime

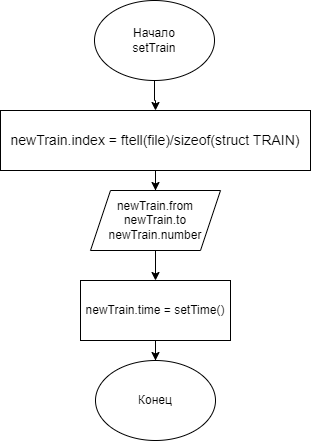


Рисунок В.9 – Структурная схема setTrain

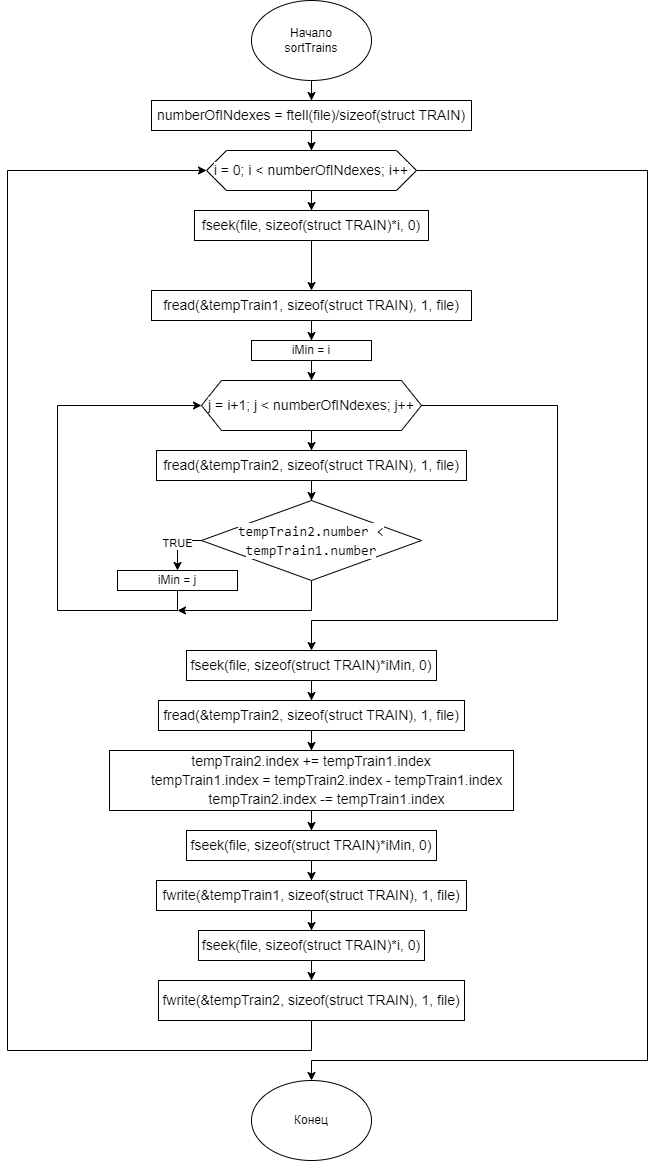


Рисунок В.10 – Структурная схема sortTrains

**Приложение Г**

**Тесты**

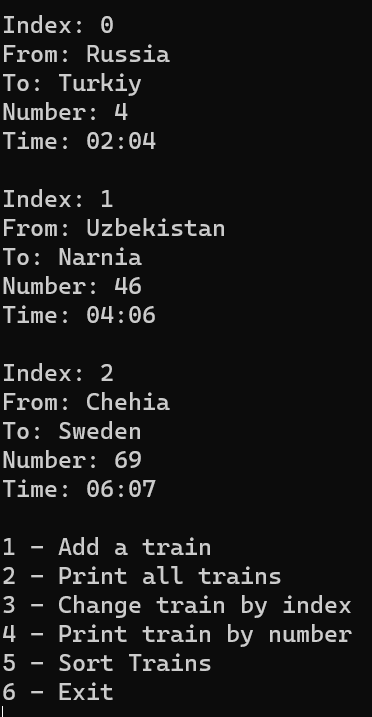
****

Рисунок Г.1 – Тест 1

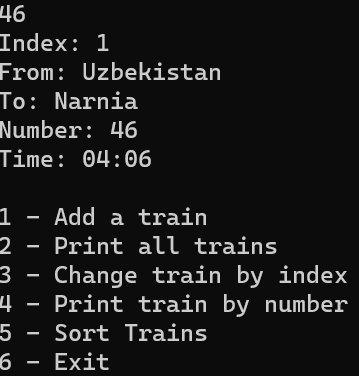
****

Рисунок Г.2 – Тест 2