**ЛАБОРАТОРНАЯ РАБОТА №3**

**«ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ**

**СПИСКОВ НА ЯЗЫКЕ С/С++»**

**Вариант 9**

**3.1 Цель работы**

Изучение списковых структур данных и приобретение навыков разработки и отладки программ, использующих динамическую память. Исследование особенностей организации списков средствами языка С/С++.

**3.2 Вариант задания**

Структуру данных из предыдущей, второй, лабораторной работы представить в виде линейного списка, элементами которого являются строки таблицы. Написать функции:

* организации списка;
* просмотра списка;
* добавления элемента в список;
* исключения элемента из списка;
* сохранения в файл;
* загрузки данных из файла и создания по этим данным нового списка;
* освобождения динамической памяти (обязательно вызывать при выходе из программы);
* функцию, которая вставляет два новых элемента перед последним элемента непустого списка.

Значения и количество записей в таблице пользователь выбирает

самостоятельно (т.е. количество строк таблицы не задается). Работу программы необходимо оформить в виде меню.

**3.3 Ход работы**

Были разработаны структурные схемы для функций программы на языке C (рисунки Д.1 – Д.14 приложение Д).

Была разработана программа на языке программирования С, которая решала поставленную задачу (листинг 3.1).

Листинг 3.1 – Код программы на языке С

#include <stdio.h>

#include <stdlib.h>

struct time {

int hours;

int minutes;

};

struct TRAIN {

char from[20];

char to[20];

int number;

struct time time;

};

struct trainList {

struct TRAIN train;

struct trainList\* next;

};

struct TRAIN setTrain();

struct time setTime();

struct trainList\* addTrainToStart(struct trainList\* trainList);

struct trainList\* addTrain(struct trainList\* trainList);

void printTrain (struct TRAIN train);

void printAllTrains(struct trainList\* train);

struct trainList\* setList();

void saveTrain(struct trainList\* train, FILE\* file);

void saveList(struct trainList\* train);

struct trainList\* deleteFirstTrain(struct trainList\* trainList);

void freeList(struct trainList\* train);

struct trainList\* makeList(struct trainList\* trainList);

void variantFunction(struct trainList\* trainList);

int main() {

struct trainList\* trainList = setList();

int isActive = 1;

while (isActive) {

int choice;

printf("%s\n", "1 - Add first train");

printf("%s\n", "2 - Add last train");

printf("%s\n", "3 - Print all trains");

printf("%s\n", "4 - Delete first train");

printf("%s\n", "5 - Make list");

printf("%s\n", "6 - Variant function");

printf("%s\n", "7 - Exit");

scanf("%d", &choice);

switch (choice) {

case 1:

trainList = addTrainToStart(trainList);

break;

case 2:

trainList = addTrain(trainList);

break;

case 3:

printAllTrains(trainList);

break;

case 4:

trainList = deleteFirstTrain(trainList);

break;

case 5:

trainList = makeList(trainList);

break;

case 6:

variantFunction(trainList);

break;

case 7:

saveList(trainList);

freeList(trainList);

isActive = 0;

break;

}

}

return 0;

}

struct TRAIN setTrain() {

struct TRAIN newTrain;

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;

}

struct trainList\* addTrainToStart(struct trainList\* trainList) {

struct trainList\* newTrain = (struct trainList\*)malloc(sizeof(struct trainList));

newTrain->train = setTrain();

newTrain->next = trainList;

return newTrain;

}

struct trainList\* addTrain(struct trainList\* trainList) {

if (!trainList) {

trainList = (struct trainList\*)malloc(sizeof(struct trainList));

trainList->train=setTrain();

trainList->next=NULL;

return trainList;

}

else {

trainList->next = addTrain(trainList->next);

}

return trainList;

}

void printTrain (struct TRAIN train) {

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 printAllTrains(struct trainList\* trainList) {

if (!trainList)

return;

printTrain(trainList->train);

printAllTrains(trainList->next);

return;

}

struct trainList\* setList() {

FILE\* file = fopen("trainData.dat", "rb");

if (!file)

return NULL;

fseek(file, 0, 2);

int trainCount = ftell(file)/sizeof(struct trainList);

struct trainList\* readTrain = NULL;

struct trainList\* trainList = NULL;

rewind(file);

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

if (!readTrain)

readTrain = (struct trainList\*)malloc(sizeof(struct trainList));

fread(readTrain, sizeof(struct trainList), 1, file);

if (!trainList) {

trainList = readTrain;

}

if (i < trainCount - 1) {

readTrain->next = (struct trainList\*)malloc(sizeof(struct trainList));

readTrain = readTrain->next;

}

}

fclose(file);

return trainList;

}

void saveTrain(struct trainList\* trainList, FILE\* file) {

if (!trainList)

return;

fwrite(trainList, sizeof(struct trainList), 1, file);

saveTrain(trainList->next, file);

return;

}

void saveList(struct trainList\* trainList) {

FILE\* file = fopen("trainData.dat", "wb");

saveTrain(trainList, file);

fclose(file);

}

struct trainList\* deleteFirstTrain(struct trainList\* trainList) {

if (!trainList)

return NULL;

struct trainList\* newFirst = trainList->next;

free(trainList);

return newFirst;

}

void freeList(struct trainList\* trainList) {

if (!trainList)

return;

freeList(trainList->next);

free(trainList);

return;

}

struct trainList\* makeList(struct trainList\* trainList) {

freeList(trainList);

trainList = NULL;

int choice = 0;

do

{

trainList = addTrain(trainList);

printf("1 - next, 0 - stop\n");

scanf("%d", &choice);

} while (choice);

return trainList;

}

void variantFunction(struct trainList\* trainList) {

if (!trainList)

return;

struct trainList\* train1 = trainList;

struct trainList\* train2 = NULL;

while (train1->next) {

train2 = train1;

train1 = train1->next;

}

struct trainList\* newTrain = (struct trainList\*)malloc(sizeof(struct trainList));

newTrain->train=setTrain();

newTrain->next=NULL;

train2->next = newTrain;

newTrain->next = (struct trainList\*)malloc(sizeof(struct trainList));

newTrain->next->train=setTrain();

newTrain->next->next=train1;

return;

}

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

**3.4 Вывод**

В ходе лабораторной работы были изучены списковые структуры данных и приобретены навыки разработки и отладки программ, использующих динамическую память. Было проведено исследование особенностей организации списков средствами языка С/С++.

**Приложение Д**

**Структурные схемы**

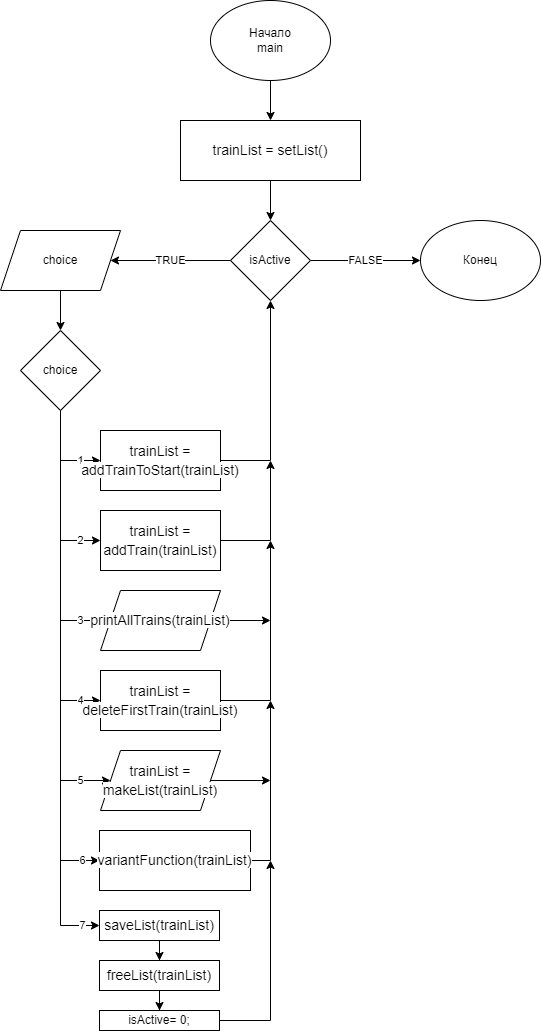


Рисунок Д.1 – Структурная схема main

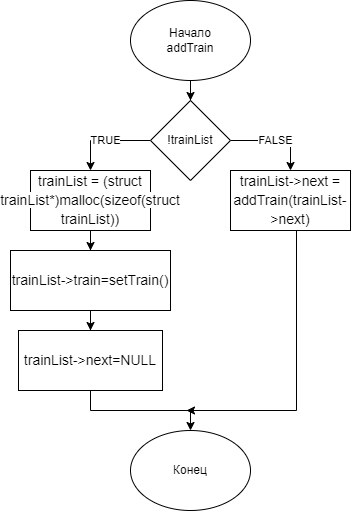


Рисунок Д.2 – Структурная схема addTrain

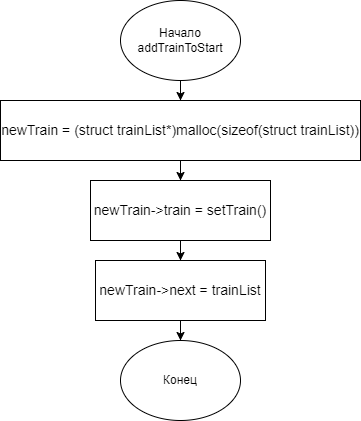


Рисунок Д.3 – Структурная схема addTrainToStart

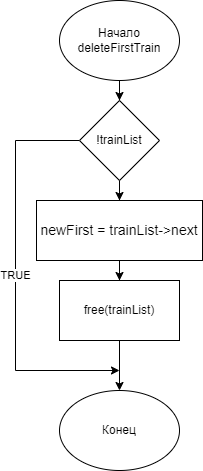


Рисунок Д.4 – Структурная схема deleteFirstTrain

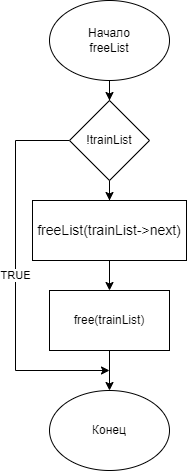


Рисунок Д.5 – Структурная схема freeList

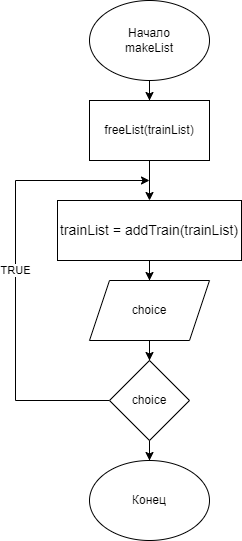


Рисунок Д.6 – Структурная схема makeList



Рисунок Д.7 – Структурная схема setTime

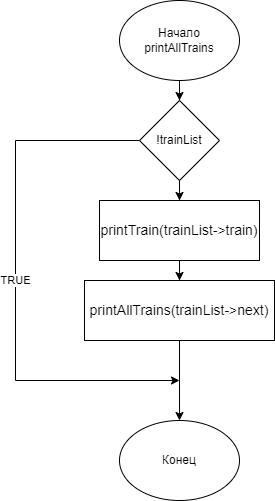


Рисунок Д.8 – Структурная схема printAllTrains

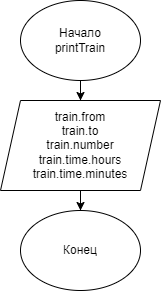


Рисунок Д.9 – Структурная схема printTrain



Рисунок Д.10 – Структурная схема saveList

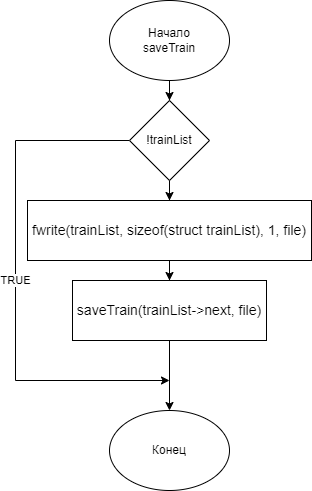


Рисунок Д.11 – Структурная схема saveTrain

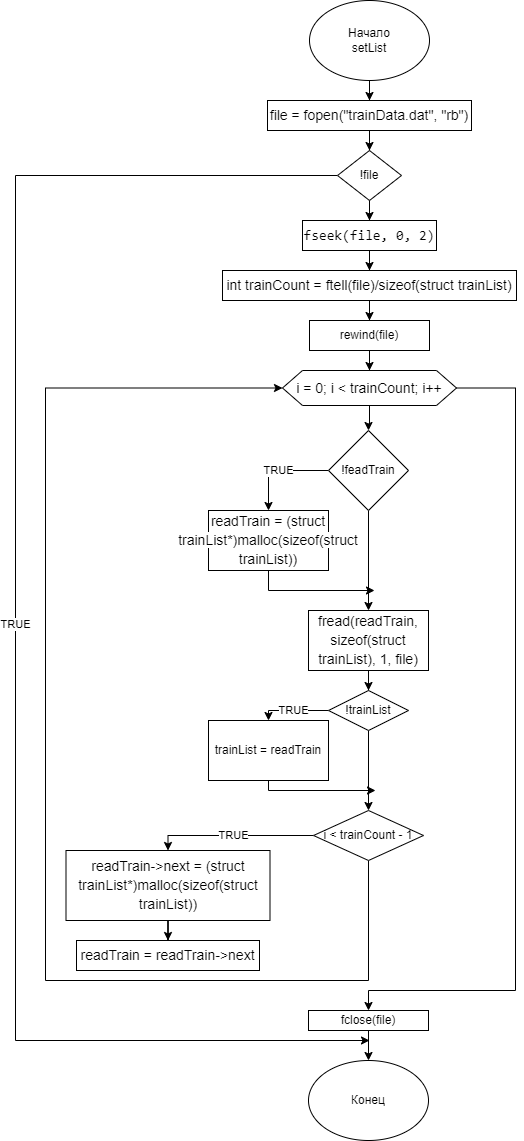


Рисунок Д.12 – Структурная схема setList

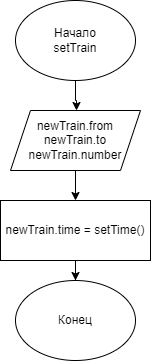


Рисунок Д.13 – Структурная схема setTrain

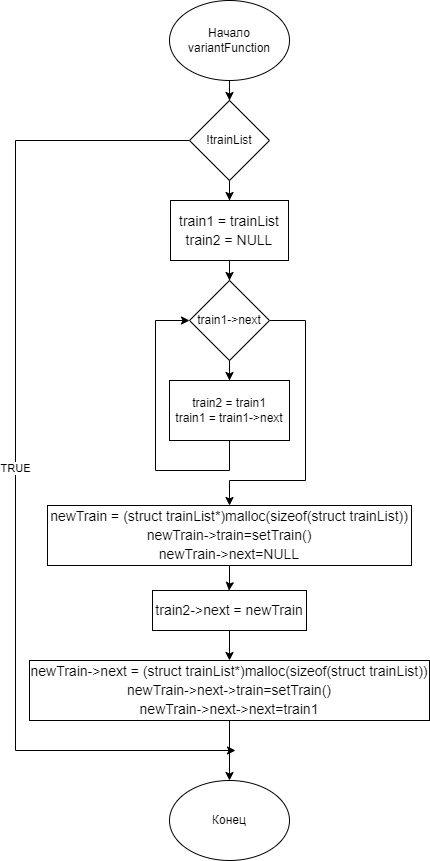


Рисунок Д.14 – Структурная схема variantFunction

**Приложение Е**

**Тесты**

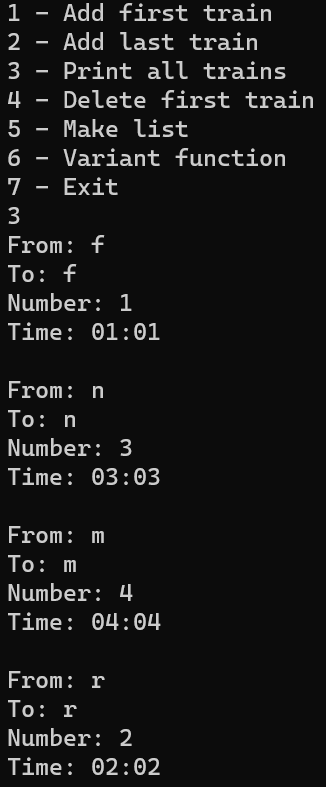
****

Рисунок Е.1 – Тест 1

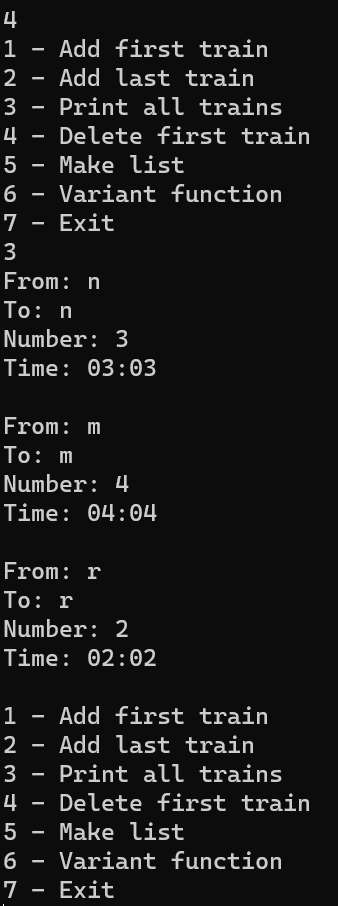
****

Рисунок Е.2 – Тест 2