**ЛАБОРАТОРНАЯ РАБОТА №5**

**««ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МОДУЛЕЙ»**

**Вариант 9**

**5.1 Цель работы**

Приобретение навыков разработки и отладки программ, использующих модули.

**5.2 Вариант задания**

Программу из лабораторной работы №3 и №4 необходимо оформить в виде модулей.

**5.3 Ход работы**

Программы третьей и четвёртой лабораторной работы были поделены на модули (листинги 5.1 – 5.9).

Листинг 5.1 – Содержимое файла time.h

#ifndef TIME\_H

#define TIME\_H

#include <stdio.h>

struct time {

int hours;

int minutes;

};

struct time setTime();

#endif

Листинг 5.2 – Содержимое файла time.c

#include "time.h"

struct time setTime() {

struct time time;

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

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

return time;

}

Листинг 5.3 – Содержимое файла train.h

#ifndef TRAIN\_H

#define TRAIN\_H

#include <stdio.h>

#include "time.h"

struct TRAIN {

char from[20];

char to[20];

int number;

struct time time;

};

struct TRAIN setTrain();

void printTrain (struct TRAIN train);

#endif

Листинг 5.4 – Содержимое файла train.c

#include "train.h"

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;

}

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;

}

Листинг 5.5 – Содержимое файла trainList.h

#ifndef TRAINLIST\_H

#define TRAINLIST\_H

#include "train.h"

#include <stdio.h>

#include <stdlib.h>

struct trainList {

struct TRAIN train;

struct trainList\* next;

};

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

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

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 run();

#endif

Листинг 5.6 – Содержимое файла trainList.c

#include "trainList.h"

#include "train.h"

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 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);

printf("%s\n", "Data loaded");

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);

printf("%s\n", "Data saved");

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;

}

int run() {

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");

printf("%s\n", "8 - Save list");

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:

freeList(trainList);

isActive = 0;

break;

case 8:

saveList(trainList);

break;

}

}

return 0;

}

Листинг 5.7 – Содержимое файла main.c

#include <stdio.h>

#include "time.h"

#include "train.h"

#include "trainList.h"

int main() {

return run();

}

Листинг 5.8 – Содержимое файла node.h

#ifndef NODE\_H

#define NODE\_H

#include <stdlib.h>

#include <stdio.h>

#include "train.h"

struct Node {

struct TRAIN train;

struct Node\* left;

struct Node\* right;

};

struct Node\* addTrain(struct Node\* Node);

struct Node\* addNode(struct Node\* Node, struct Node\* newNode);

void printAllTrains(struct Node\* train);

void printTree(struct Node\* node, int otstup);

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

void saveTree(struct Node\* train);

void freeTree(struct Node\* train);

struct Node\* makeTree(struct Node\* Node);

struct Node\* deleteNode(struct Node\* node, int number);

struct Node\* loadData(struct Node\* node);

struct Node\* loadTree(struct Node\* Node, FILE\* file);

struct Node\* changeNumbers (struct Node\* node, int number);

int printTrainsByLetter(struct Node\* node, char letter);

int run();

#endif

Листинг 5.9 – Содержимое файла node.c

#include "node.h"

struct Node\* addTrain(struct Node\* node) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->train = setTrain();

newNode->left=NULL;

newNode->right=NULL;

node = addNode(node, newNode);

return node;

}

struct Node\* addNode(struct Node\* node, struct Node\* newNode) {

if (!node)

node = newNode;

else if (node->train.number > newNode->train.number)

node->left = addNode(node->left, newNode);

else

node->right = addNode(node->right, newNode);

return node;

}

struct Node\* deleteNode(struct Node\* node, int number) {

if (!node)

return NULL;

if (node->train.number == number) {

struct Node\* left = node->left;

left = addNode(left, node->right);

free(node);

return left;

}

else if (node->train.number < number)

node->right = deleteNode(node->right, number);

else

node->left = deleteNode(node->left, number);

return node;

}

void printAllTrains(struct Node\* Node) {

if (Node == NULL)

return;

printAllTrains(Node->left);

printTrain(Node->train);

printAllTrains(Node->right);

return;

}

void printTree(struct Node \*node, int otstup) {

if (node) {

otstup+=3;

printTree(node->right, otstup);

for(int i=0; i < otstup; i++) printf(" ");

printf("%d\n",node->train.number);

printTree(node->left,otstup);

}

}

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

if (!Node)

return;

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

saveTrain(Node->left, file);

saveTrain(Node->right, file);

return;

}

void saveTree(struct Node\* Node) {

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

if (!file) {

printf("Open file error");

return;

}

saveTrain(Node, file);

printf("\t%s\n", "Data saved");

fclose(file);

return;

}

struct Node\* loadData(struct Node\* node) {

freeTree(node);

printf("\tOld tree deleted\n");

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

if (!file) {

printf("\tOpen file error\n");

return NULL;

}

node = loadTree(node, file);

printf("\tData loaded\n");

fclose(file);

return node;

}

struct Node\* loadTree(struct Node\* Node, FILE\* file) {

if (!feof(file)) {

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

fread(Node, sizeof(struct Node), 1, file);

if (Node->left)

Node->left = loadTree(Node->left, file);

if (Node->right)

Node->right = loadTree(Node->right, file);

}

else {

return NULL;

}

return Node;

}

void freeTree(struct Node\* Node) {

if (!Node)

return;

freeTree(Node->left);

freeTree(Node->right);

free(Node);

return;

}

struct Node\* makeTree(struct Node\* Node) {

freeTree(Node);

printf("\tOld tree deleted\n");

Node = NULL;

int choice = 0;

do

{

Node = addTrain(Node);

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

scanf("%d", &choice);

} while (choice);

return Node;

}

struct Node\* changeNumbers (struct Node\* node, int number) {

if (!node)

return NULL;

if (node->train.number < number) {

node->train.number = number;

}

node->left = changeNumbers(node->left, number);

node->right = changeNumbers(node->right, number);

return node;

}

int printTrainsByLetter(struct Node\* node, char letter) {

if (!node)

return 0;

int exist = printTrainsByLetter(node->right, letter) + printTrainsByLetter(node->left, letter);

if (node->train.from[0] == letter) {

printTrain(node->train);

exist++;

}

return exist;

}

int run() {

struct Node\* Node = NULL;

int isActive = 1;

while (isActive) {

int choice;

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

printf("%s\n", "2 - Print struct tree");

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

printf("%s\n", "4 - Make tree");

printf("%s\n", "5 - Delete train");

printf("%s\n", "6 - Change numbers under");

printf("%s\n", "7 - Print trains with letter");

printf("%s\n", "8 - Load data");

printf("%s\n", "9 - Save data");

printf("%s\n", "10 - Free memory and exit");

scanf("%d", &choice);

switch (choice) {

case 1:

Node = addTrain(Node);

break;

case 2:

printTree(Node, 0);

break;

case 3:

printAllTrains(Node);

break;

case 4:

Node = makeTree(Node);

break;

case 5:

int number = 0;

printf("Train number: "); scanf("%d", &number);

Node = deleteNode(Node, number);

break;

case 6:

int number1 = 0;

printf("Train number: "); scanf("%d", &number1);

Node = changeNumbers(Node, number1);

break;

case 7:

printf("Letter: ");

char letter;

scanf("%s", &letter);

if (!printTrainsByLetter(Node, letter))

printf("No trains with first letter: %c\n", letter);

break;

case 8:

Node = loadData(Node);

break;

case 9:

saveTree(Node);

break;

case 10:

freeTree(Node);

isActive = 0;

break;

}

}

return 0;

}

Были проведены тесты программ (рисунки И.1 – И.2 приложение И).

**5.4 Вывод**

В ходе лабораторной работы программы из лабораторных работ №3 и №4 были оформлены в виде модулей. Были приобретены навыки разработки и отладки программ, использующих модули.

**Приложение И**

**Тесты**

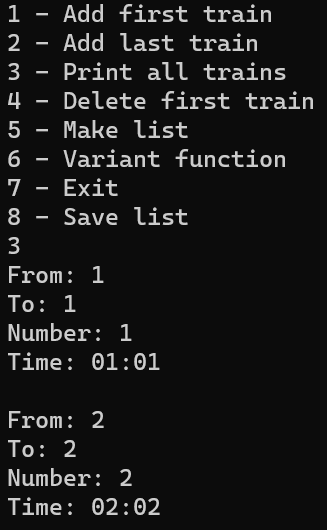
****

Рисунок И.1 – Тест программы из третьей лабораторной работы

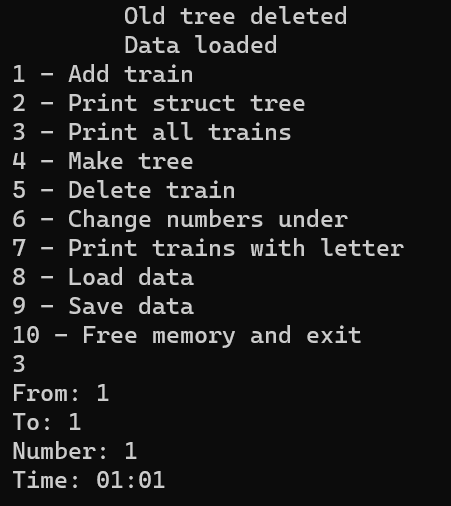


Рисунок И.2 – Тест программы из четвёртой лабораторной работы