**7 ЛАБОРАТОРНАЯ РАБОТА №7**

**«ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ СПИСКОВ НА ЯЗЫКЕ С/С++»**

**7.1 Цель работы**

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

**7.2 Вариант задания – 20**

Требуется представить таблицу 7.1 в виде линейного списка, элементами которого являются строки таблицы. Написать процедуры организации, добавления элемента в список, исключения элемента из списка, просмотра списка, а также процедуру, которая удаляет из непустого списка первый элемент.

Значения и количество записей в таблице студент выбирает самостоятельно. Исходные данные после организации списка должны сохраняться в файле и при повторном запуске программы считываться из файла. Количество строк таблицы не задается.

**Таблица 7.1 – Расписание**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N поезда | Станция отправления | Станция назначения | Время отправления | Время прибытия | Стоимость билета |

**7.3 Порядок выполнения работы**

7.3.1 Разработать алгоритм решения задачи.

7.3.2 Разработать программу на языке Си.

7.3.3 Разработать тестовые примеры. Выполнить тестирование используя разработанные тестовые примеры. Выполнить отладку.

7.3.4 Сделать выводы по проделанной работе.

**7.4 Ход работы**

7.4.1 Для задания был разработан алгоритм решения задачи и отражен на структурных схемах, которые представлены на рисунках 7.1 и 7.2.

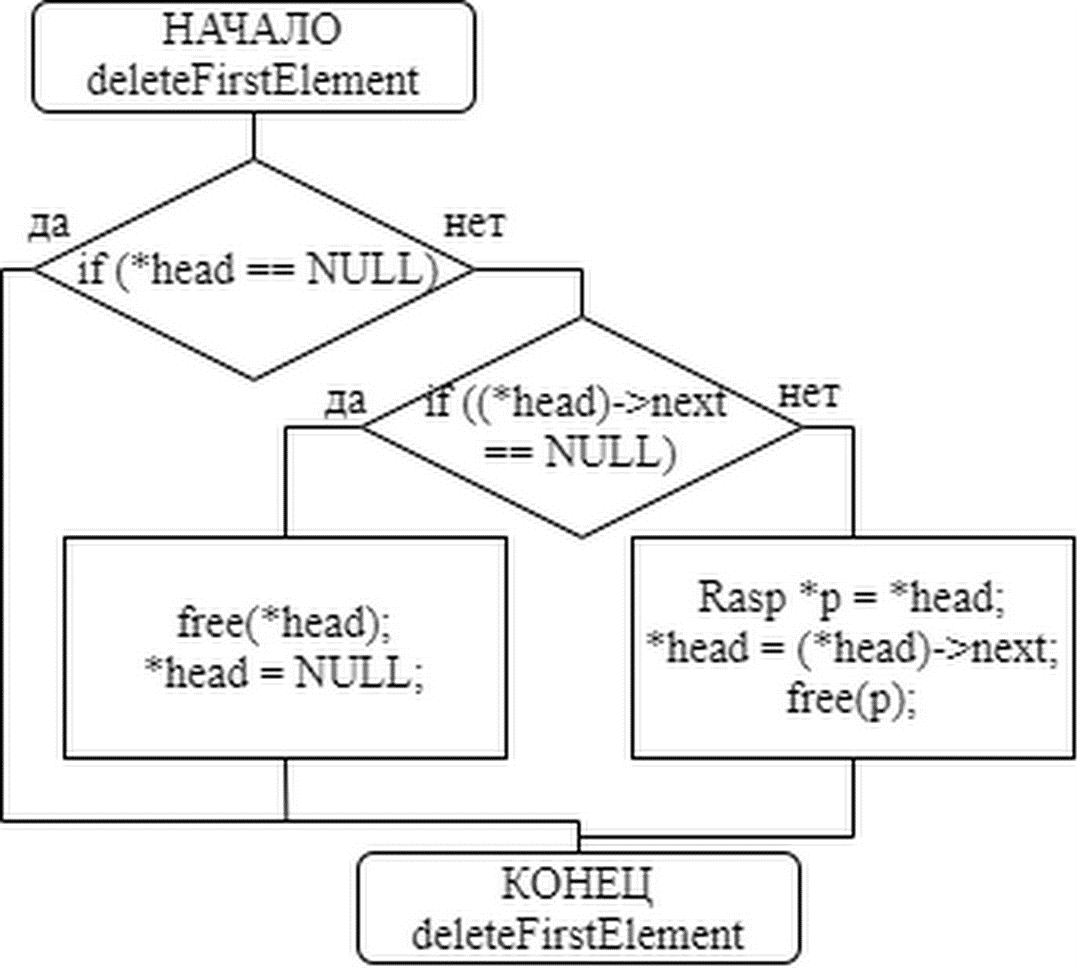
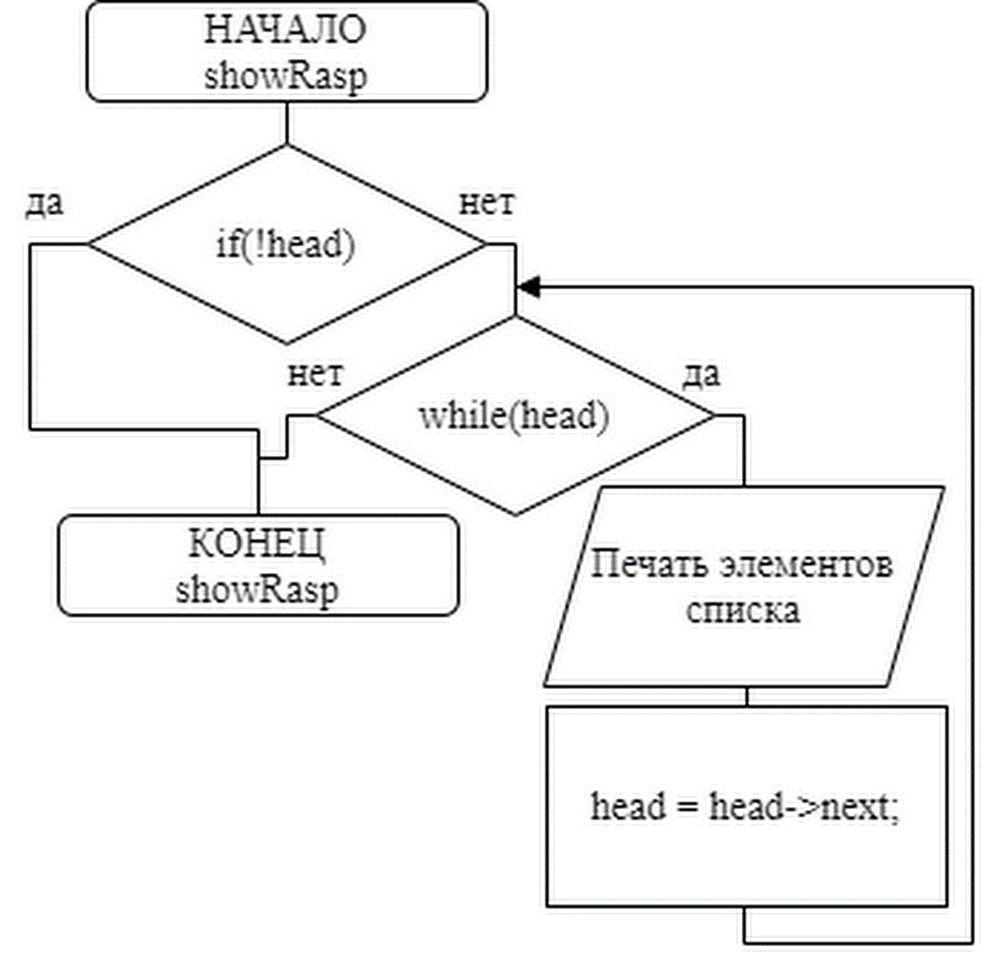
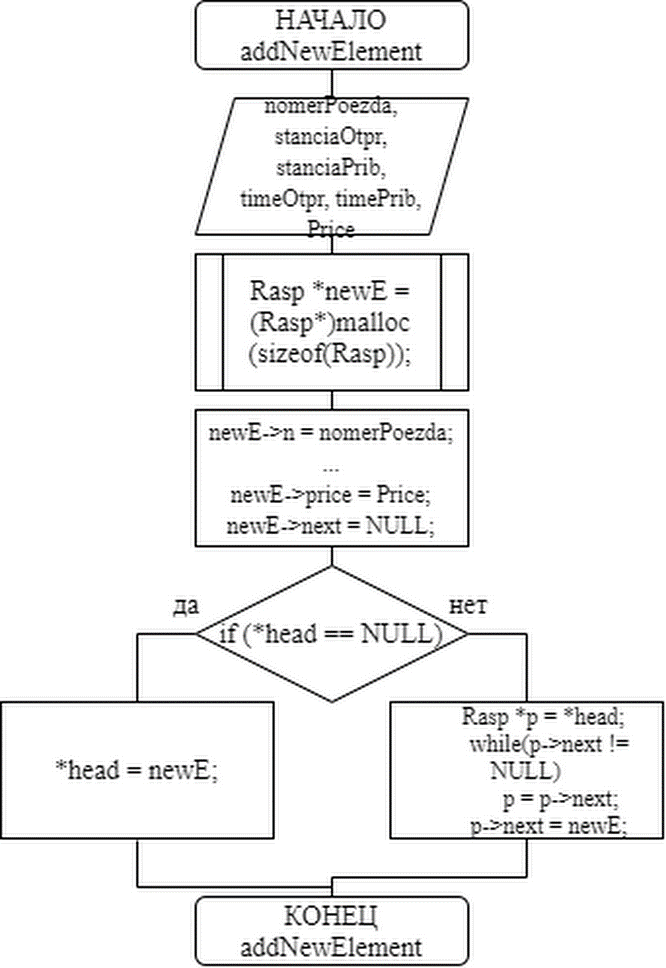
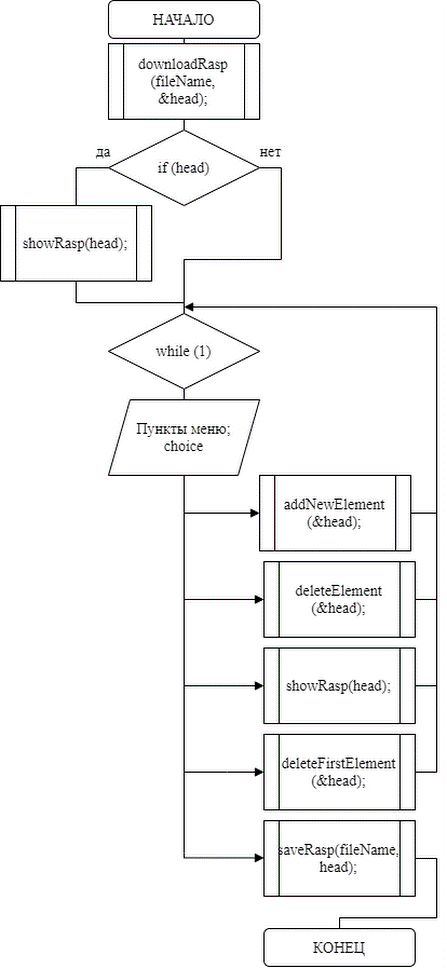


Рисунок 7.1 – Структурная схема основной программы, функции добавления элемента в список, демонстрации списка, удаления первого элемента списка

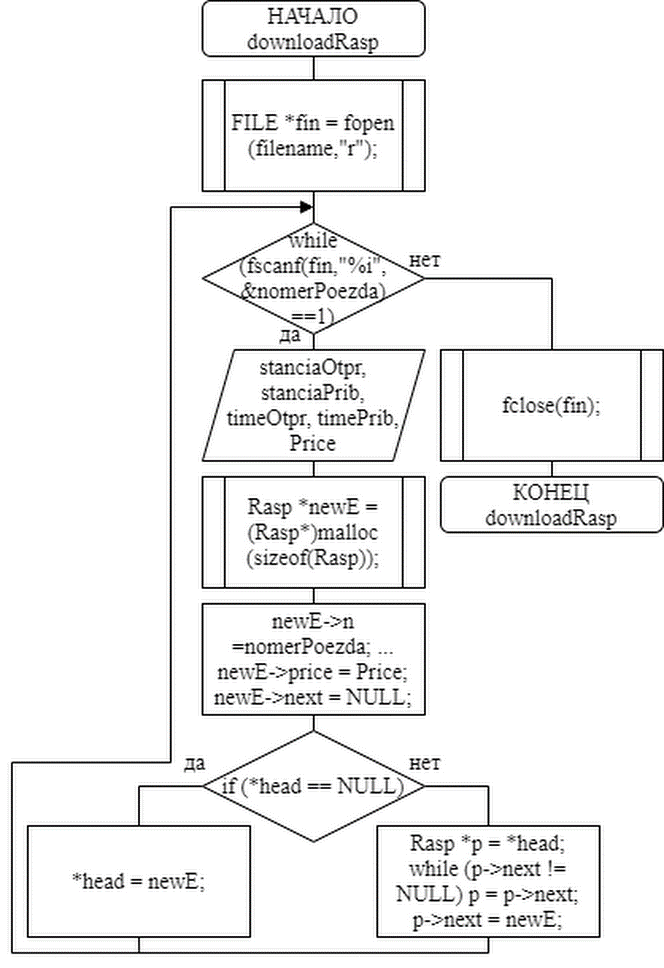
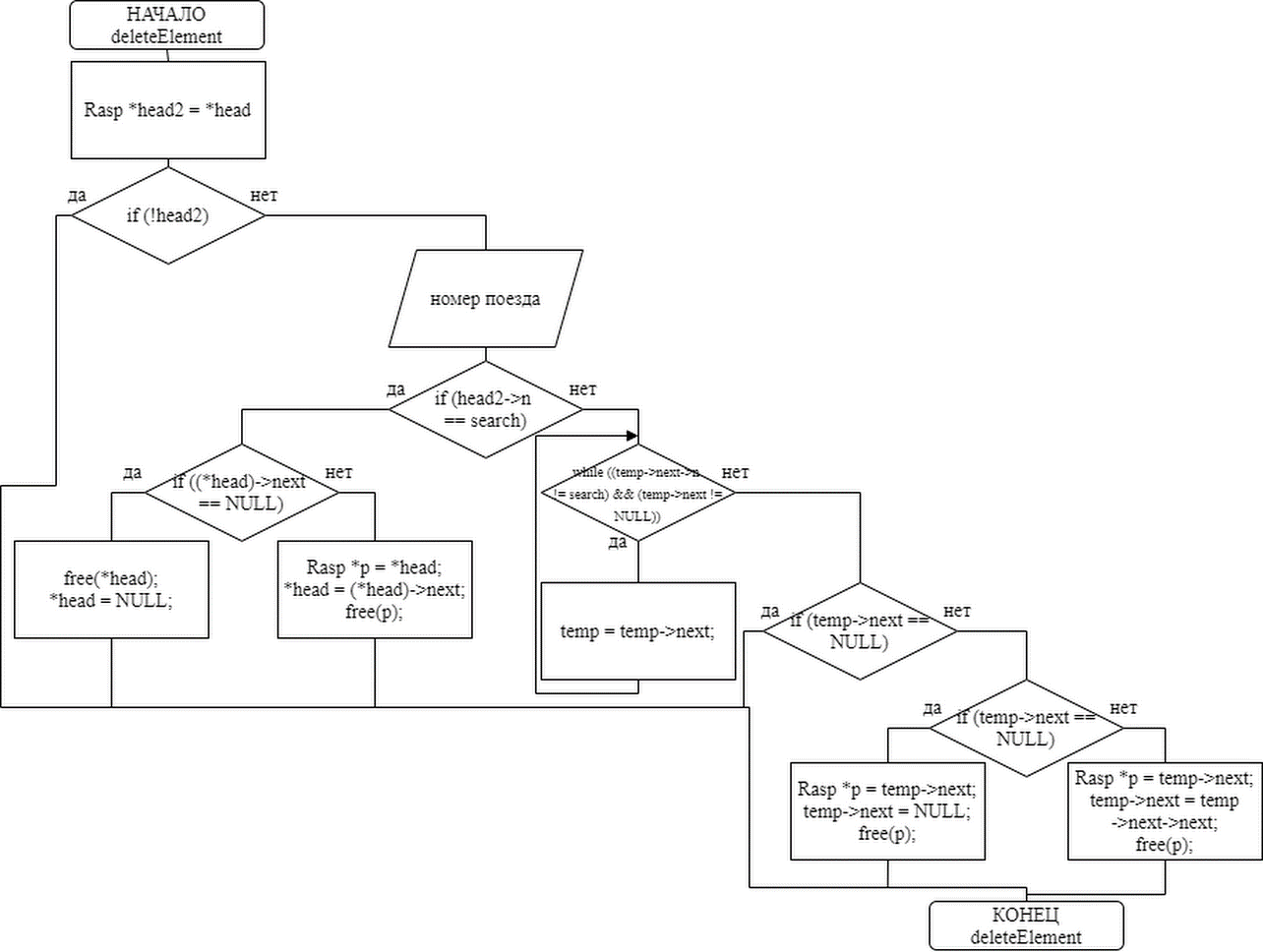
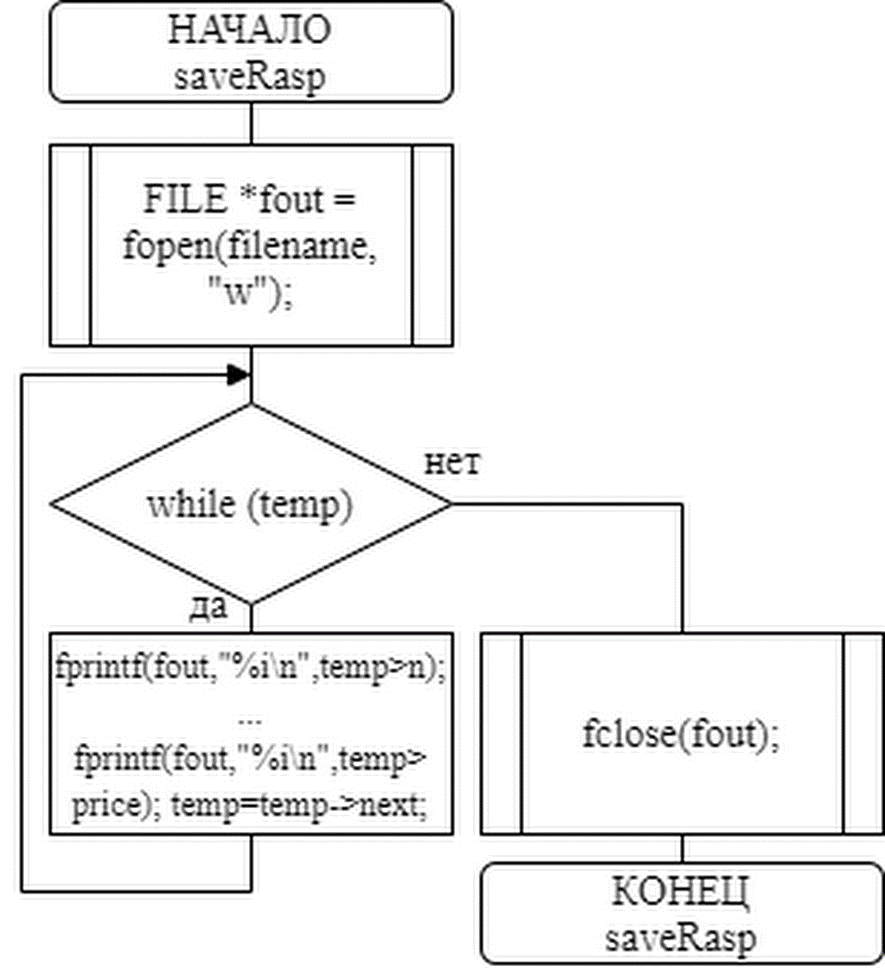


Рисунок 7.2 – Структурная схема функций удаления элемента по номеру поезда, сохранению и загрузке списка

7.4.2 Написана программа на Си согласно вышеописанного алгоритма.

#include <stdio.h>

#include <windows.h>

#include <stdlib.h>

#include <string.h>

typedef struct Raspisanie //расписание

{

int n; //номер поезда

char so[19], //станция отправления

sp[18], //станция прибытся

vo[6], //время отправления

vp[6]; //время прибытия

int price; //стоимость билета

struct Raspisanie \*next;

}Rasp;

//прототипы функций

void goodbye();

void showRasp(const Rasp \*head);

void addNewElement(Rasp \*\*head);

void deleteFirstElement(Rasp \*\*head);

void deleteElement(Rasp \*\*head);

void saveRasp(char\* filename, Rasp \*temp);

void downloadRasp(char\* filename, Rasp \*\*head);

//========================Главная\_программа========================

int main()

{

SetConsoleCP(1251);// установка кодовой страницы win-cp 1251 в поток ввода

SetConsoleOutputCP(1251); // установка кодовой страницы win-cp 1251 в поток вывода

char \*fileName = "Raspisanie.txt"; //имя файла

Rasp \*head = NULL;

int choice;

//загрузить список из файла и если он есть отобразить его.

downloadRasp(fileName, &head);

if (head)

{

printf("Послeднее сохранение:\n\n");

showRasp(head);

}

while (1)

{

printf("\nВыберите дальнейшее действие: \n");

printf(" 1 - Добавить элемент в список; \n");

printf(" 2 - Исключить элемент из списка; \n");

printf(" 3 - Просмотр списка; \n");

printf(" 4 - Удалить первый элемент (my); \n");

printf(" 5 - Сохранить и выйти; \n");

printf(" -> ");

scanf("%i", &choice);

printf("\n");

switch(choice)

{

case 1:

{

addNewElement(&head);

break;

}

case 2:

{

deleteElement(&head);

break;

}

case 3:

{

showRasp(head);

break;

}

case 4:

{

deleteFirstElement(&head);

break;

}

case 5:

{

saveRasp(fileName, head);

goodbye();

return 0;

}

default:

printf("Вы ввели неверный символ\n");

break;

}

}

}

//============Добавить в список новый элемент============

void addNewElement(Rasp \*\*head)

{

int nomerPoezda;

printf("Введите номер поезда:\n");

scanf("%i", &nomerPoezda);

while ((getchar()) != '\n');

char stanciaOtpr[21];

printf("Введите станцию отправления:\n");

gets(stanciaOtpr);

char stanciaPrib[21];

printf("Введите станцию прибытся:\n");

gets(stanciaPrib);

char timeOtpr[21];

printf("Введите время отправления:\n");

gets(timeOtpr);

char timePrib[21];

printf("Введите время прибытия:\n");

gets(timePrib);

int Price;

printf("Введите стоимость билета:\n");

scanf("%i", &Price);

Rasp \*newE = (Rasp\*)malloc(sizeof(Rasp));

newE->n = nomerPoezda;

strcpy(newE->so, stanciaOtpr);

strcpy(newE->sp, stanciaPrib);

strcpy(newE->vo, timeOtpr);

strcpy(newE->vp, timePrib);

newE->price = Price;

newE->next = NULL;

if (\*head == NULL)

\*head = newE;

else

{

Rasp \*p = \*head;

while(p->next != NULL)

p = p->next;

p->next = newE;

}

}

//========================Показать\_список========================

void showRasp(const Rasp \*head)

{

if(!head)

{

printf("Список отсутствует\n");

return;

}

printf("--------------------------------------------------------------------\n");

printf("| № Поезда | Станция отправления | Станция назначения | Время отправления | Время прибытия | Цена |\n");

printf("--------------------------------------------------------------------\n");

while(head)

{

printf("| %-8i | %-21s | %-21s | %-17s | %-14s | %-6i |\n",

head->n, head->so, head->sp, head->vo, head->vp, head->price);

printf("----------------------------------------------------------------\n");

head = head->next;

}

}

//==============Удалить первый элемент списка================

void deleteFirstElement(Rasp \*\*head)

{

if (\*head == NULL)

{

printf("Списка нет, удалять нечего\n");

return;

}

else if ((\*head)->next == NULL)

{

free(\*head);

\*head = NULL;

}

else

{

Rasp \*p = \*head;

\*head = (\*head)->next;

free(p);

}

printf("Первый элемент успешно удалён\n");

}

//===========Удалить элемент в списке по номеру поезда==========

void deleteElement(Rasp \*\*head)

{

Rasp \*head2 = \*head; //запасной head

//если нет списка

if (!head2)

{

printf("Список отсутствует\n");

return;

}

int search;

printf("Введите номер удаляемого поезда:");

scanf("%i", &search);

//если элемент с этим поездом первый

if (head2->n == search)

{

if ((\*head)->next == NULL)

{

free(\*head);

\*head = NULL;

}

else

{

Rasp \*p = \*head;

\*head = (\*head)->next;

free(p);

}

printf("Элемент списка с указанным поездом удалён\n");

return;

}

//ищем элемент с этим поездом

Rasp \*temp = head2;

while ((temp->next->n != search) && (temp->next != NULL))

temp = temp->next;

//если не нашли поезд

if (temp->next == NULL)

{

printf("Указываемый поезд отсутствует\n");

return;

}

//если последний элемент списка

if (temp->next == NULL)

{

Rasp \*p = temp->next;

temp->next = NULL;

free(p);

}

else //если элемент не первый и не последний

{

Rasp \*p = temp->next;

temp->next = temp->next->next;

free(p);

}

printf("Элемент списка с указанным поездом удалён\n");

}

//========================сохранить список========================

void saveRasp(char\* filename, Rasp \*temp)

{

FILE \*fout = fopen(filename, "w");

if (!fout)

{

printf("Файл с сохранением не создан :(\n");

return;

}

while (temp)

{

fprintf(fout, "%i\n", temp->n);

fprintf(fout, "%s\n", temp->so);

fprintf(fout, "%s\n", temp->sp);

fprintf(fout, "%s\n", temp->vo);

fprintf(fout, "%s\n", temp->vp);

fprintf(fout, "%i\n", temp->price);

temp=temp->next;

}

printf("Данные успешно сохранены\n\n");

fclose(fout);

}

//========================загрузить список========================

void downloadRasp(char\* filename, Rasp \*\*head)

{

FILE \*fin = fopen(filename, "r");

if (!fin)

{

printf("Файл с сохранением не удалось прочесть :(\n");

return;

}

int nomerPoezda;

while (fscanf(fin, "%i", &nomerPoezda) == 1)

{

char stanciaOtpr[21];

fscanf(fin, "%s", &stanciaOtpr);

char stanciaPrib[21];

fscanf(fin, "%s", &stanciaPrib);

char timeOtpr[6];

fscanf(fin, "%s", &timeOtpr);

char timePrib[6];

fscanf(fin, "%s", &timePrib);

int Price;

fscanf(fin, "%i", &Price);

Rasp \*newE = (Rasp\*)malloc(sizeof(Rasp));

newE->n = nomerPoezda;

strcpy(newE->so, stanciaOtpr);

strcpy(newE->sp, stanciaPrib);

strcpy(newE->vo, timeOtpr);

strcpy(newE->vp, timePrib);

newE->price = Price;

newE->next = NULL;

if (\*head == NULL)

\*head = newE;

else {

Rasp \*p = \*head;

while (p->next != NULL)

p = p->next;

p->next = newE;

}

}

printf("Данные успешно загружены\n");

fclose(fin);

}

//========================Прощание========================

void goodbye()

{

printf("================================= \n ");

printf(" Программа окончила свою работу \n");

printf("================================= \n");

}

7.4.3 Выполнена отладка программы.

Результаты тестирования отображены на рисунках 7.3- 7.6.

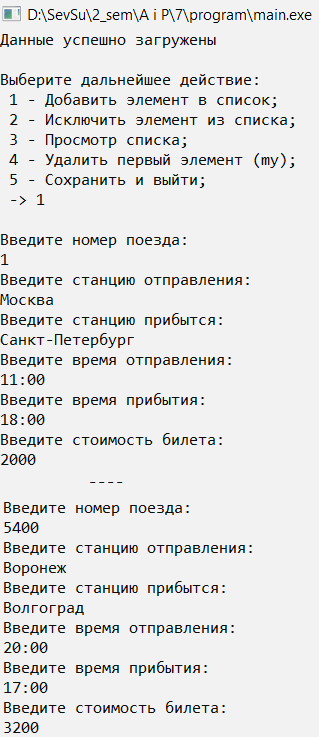


Рисунок 7.3 – Меню и добавление элементов структуры

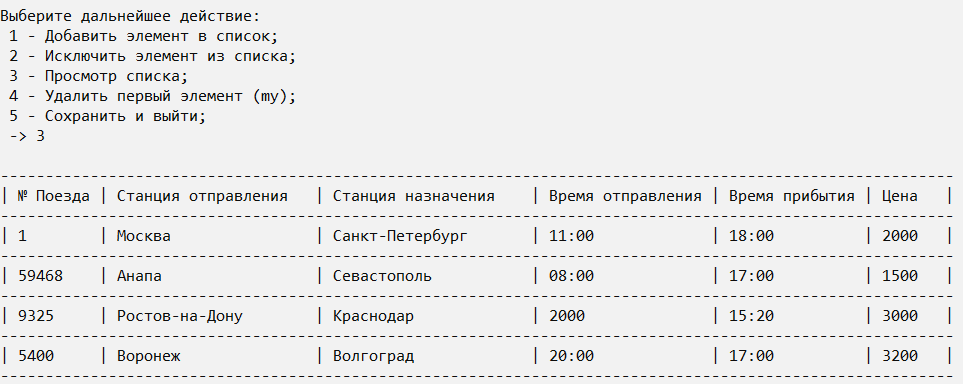


Рисунок 7.4 – Вывод элементов списка в виде таблицы

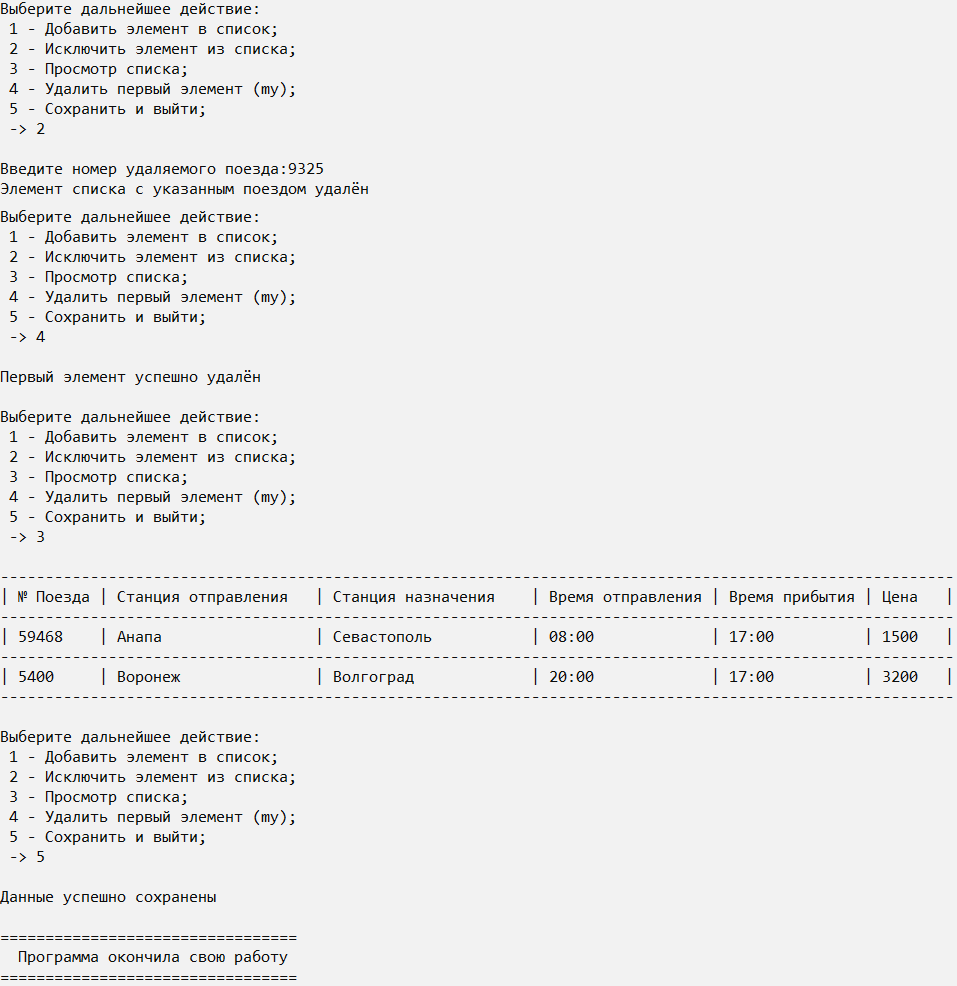


Рисунок 7.5 – Удаление элемента списка по номеру, удаление первого элемента и результат, а так же сохранение и выход

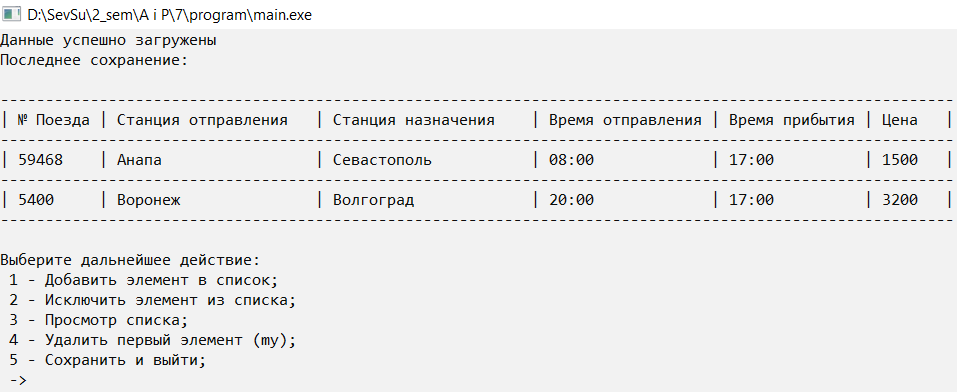


Рисунок 7.6 – Загруженные данные при новом запуске программы

Результаты тестирования полностью соответствуют ожиданиям.

**Выводы**

В ходе выполнения данной лабораторной работы были получены навыки разработки и отладки программ, использующих динамическую память. Были изучены способы организации списков на языке Си, а также способы удаления элементов списка, добавления элементов в список и просмотр списка. Исследованы способы сохранения и загрузки элементов списка с помощью текстовых файлов. Полученные во время разработки навыки помогут разрабатывать более сложные программы со структурами данных и файлами, более эффективные по времени выполнения алгоритмы.