МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВОЙ ПРОЕКТ
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

ассистент		А.Э. Зянчурин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

РАЗРАБОТКА ПРОГРАММЫ «Расписание поездов»

по дисциплине: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4932		Н. С. Иванов
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2020

Оглавление

1. Постановка задачи

1.1 Задачей курсового проекта является разработка программы «расписание поездов» с использованием заданных структур данных, которая позволяет вводить информацию, хранить её в файле (сохранять и загружать), осуществлять поиск, модификацию, сортировку и удаление данных.

1.2 Выбранный вариант

Вариант 8

Данные о рейсе хранятся в структуре TRAIN, содержащей следующие поля:

- название пункта назначения;
- номер поезда;
- время отправления.

Задание на поиск: найти поезд, номер которого введен с клавиатуры.

Предметная область

В начале в 19 века, человечество открыло для себя новый вид транспорта – железнодорожный транспорт. Именно в 1807 году были организованы регулярные пассажирские перевозки по железнодорожным путям между Суонси и Мамблза в Уэльсе [1]. Поначалу этот транспорт использовался лишь ресурсодобывающими и сельскохозяйственными организациями, однако постепенно сеть железных дорог развивалась. Примерно с 1865 года начинается «Золотой век» железных дорог в США и это означало, что железнодорожный транспорт дойдет и до обычных горожан. И уже во второй половине 1880-х годов был достигнут наивысший уровень прироста мировой железнодорожной сети в истории.

По мере развития железнодорожного транспорта обострялась необходимость в составлении расписания поездов. Причиной этой необходимости была экономия времени, ведь в ожидании поезда можно провести до нескольких часов.

На сегодняшний день, железнодорожный транспорт не потерял своей значимости и, следовательно, потребность в составлении расписания не исчезла. Таким образом тема до сих пор актуальна.

Роль расписания поездов в организации их движения состоит в предсказуемости их прибытия и отправления.

2. Описание структур данных

Данные о рейсе хранятся в структуре TRAIN

struct TRAIN{

```
    int number;
    wchar_t destination[57]{'\0'};
    u_time time;
    TRAIN* next;
    TRAIN* prev;
};

struct u_time
{
    wchar_t str[9] = {'0','0',':','0','0',':','0','0','\0'};
    int hour = 0;
    int minute = 0;
    int second = 0;
};
```

Структура TRAIN содержит поля: number, destination, time. И организована в двунаправленный линейный список (реализация с помощью класса).

Номер поезда хранится в целочисленном виде в поле number.

Пункт назначения хранится как массив символов (на самом деле wchar_t это целочисленный тип данных, но используется для представления символов юникода).

Время для каждого поезда хранится с помощью структуры u_time в текстовом и целочисленном формате.

Структура u_time содержит текстовое представление времени в виде массива символов и его целочисленное представление

База данных хранится на диске в виде тестового файла.

Ограничения на данные:

Номер поезда принадлежит промежутку $[0, 2^{31}]$. Таким образом отрицательного номера поезда не предусмотрено, а максимальный номер ограничен размером int.

Пункт назначения ограничен длинной массива. Длина в свою очередь подобрана так, чтобы ее хватило. (Т.е. Самое длинное название города в мире (57): Лланвайрпуллгуингиллгогерихуирндробуллллантисилиогогогóх, а в России (25): Александровск-Сахалинский)

Время же ограничено размером (продолжительностью) суток, часов и минут.

3. Описание программы и созданных функций

Программа реализована на языке C++ в виде консольного приложения. В главной функции main() реализовано меню пользователя, в котором каждому действию соответствует определенная цифра. Реализованы следующие функции для работы с данными: добавление, удаление, редактирование записи, сортировка, поиск поездов по номеру поезда, загрузка данных из файла и сохранение в текстовый файл, вывод на экран.

Класс LinkedList

Поля класса (Рис. 1, Приложение 1) содержат указатели на последний, первый и текущий (при переборе списка) элемент, а также количество элементов в списке.

Добавление записи

Функция (Рис. 2, Приложение 1) принимает элемент и добавляет его в список (фактически функция меняет указатели на структуре, преданной первым параметром, и связанных с ней структурах. Структура должна быть выделена динамически и затем передана как первый аргумент функции).

Если передан второй аргумент (позиция куда добавить элемент), функция вставляет элемент на эту позицию, а элемент который был в этой позиции раньше, становится справа (следующим).

```
void СвязанныйСписок::Добавить(ПОЕЗД *новый_элемент, ЧИСЛО позиция)
{
    if (позиция == -1)
    {
        позиция = количество_элементов;
    }
    else if (позиция > количество_элементов)
    {
        позиция = количество_элементов;
    }
```

```
    else if(позиция < -1)
    {
        позиция = 0;
    }

    if(указатель_на_первый_элемент == nullptr)
    {
        указатель_на_первый_элемент = новый_элемент;
        указатель на последний элемент = новый_элемент;
        новый_элемент->следующий = nullptr;
        новый_элемент->предыдущий = nullptr;
    }
    else
    {
        if (позиция == 0)
        {
            новый_элемент->предыдущий = nullptr;
            указатель_на_первый_элемент->предыдущий = новый_элемент;
            новый_элемент->следующий = указатель_на_первый_элемент;

            указатель_на_первый_элемент = новый_элемент;
        }
        else if (позиция == количество_элементов)
        {
            новый_элемент->следующий = nullptr;
            указатель на последний элемент->следующий = новый_элемент;
            новый_элемент->предыдущий = указатель на последний элемент;

            указатель на последний элемент = новый_элемент;
        }
        else
        {
            указатель_на_текущий_элемент = указатель_на_первый_элемент;
            while (указатель_на_текущий_элемент->следующий != nullptr && позиция > 0)
            {
                указатель_на_текущий_элемент = указатель_на_текущий_элемент->следующий;
                позиция -= 1;
            }

            //          вставить после указатель_на_текущий_элемент
            указатель_на_текущий_элемент->предыдущий->следующий = новый_элемент;
            новый_элемент->предыдущий = указатель_на_текущий_элемент->предыдущий;
            указатель_на_текущий_элемент->предыдущий = новый_элемент;
            новый_элемент->следующий = указатель_на_текущий_элемент;
        }
    }

    количество_элементов += 1;
}
```

Удаление записи

Функция (Рис. 3, Приложение 1) принимает позицию удаляемого элемента и удаляет его, освобождая при этом память и меняя указатели на связанных с ним элементах.

```
void СвязанныйСписок::Удалить(ЧИСЛО позиция)
{
    if (позиция == -1)
    {
        позиция = количество_элементов - 1;
    }
    else if (позиция > количество_элементов - 1 && количество_элементов != 0)
    {
        позиция = количество_элементов - 1;
```

```
    }
    else if(позиция < -1)
    {
        позиция = 0;
        return;
    }

    if(указатель_на_первый_элемент == nullptr || количество_элементов == 0)
    {
        // Список пуст
        return;
    }
    else if(указатель_на_первый_элемент == указатель на последний элемент)
    {
        // только 1 элемент
        Удалить указатель_на_первый_элемент;
    }
    else
    {
        if (позиция == 0)
        {
            указатель_на_первый_элемент = указатель_на_первый_элемент->следующий;
            Удалить указатель_на_первый_элемент->предыдущий;
            указатель_на_первый_элемент->предыдущий = nullptr;
        }
        else if (позиция == количество_элементов - 1)
        {
            указатель на последний элемент = указатель на последний элемент->предыдущий;
            Удалить указатель на последний элемент->следующий;
            указатель на последний элемент->следующий = nullptr;
        }
        else
        {
            указатель_на_текущий_элемент = указатель_на_первый_элемент;
            while (указатель_на_текущий_элемент->следующий != nullptr && позиция > 0)
            {
                указатель_на_текущий_элемент = указатель_на_текущий_элемент->следующий;
                позиция -= 1;
            }

            указатель_на_текущий_элемент->предыдущий->следующий = указатель_на_теку-
щий_элемент->следующий;
            указатель_на_текущий_элемент->следующий->предыдущий = указатель_на_теку-
щий_элемент->предыдущий;
            Удалить указатель_на_текущий_элемент;
        }
    }

    количество_элементов -= 1;
}
```

Получение записи

Функция (Рис. 4, Приложение 1) возвращает копию элемента, номер которого передан в аргументе функции иначе вернуть элемент с полями «ошибка».

```
ПОЕЗД СвязанныйСписок::Получить(ЧИСЛО позиция)
{
    if (позиция == -1)
    {
        позиция = количество_элементов - 1;
    }
    else if (позиция > количество_элементов - 1 && количество_элементов != 0)
    {
```

```
        позиция = количество_элементов - 1;
    }
    else if(позиция < -1)
    {
        позиция = 0;
    }


    if(указатель_на_первый_элемент == nullptr || количество_элементов == 0)
    {
        // Список пуст
        return ошибка;
    }
    else if(указатель_на_первый_элемент == указатель на последний элемент)
    {
        // only 1 элемент
        return *указатель_на_первый_элемент;
    }
    else
    {
        if (позиция == 0)
        {
            return *указатель_на_первый_элемент;
        }
        else if (позиция == количество_элементов - 1)
        {
            return *указатель на последний элемент;
        }
        else
        {
            указатель_на_текущий_элемент = указатель_на_первый_элемент;
            while (указатель_на_текущий_элемент->следующий != nullptr && позиция > 0)
            {
                указатель_на_текущий_элемент = указатель_на_текущий_элемент->следующий;
                позиция -= 1;
            }
            return *указатель_на_текущий_элемент;
        }
    }
}
```

Поиск записи

Функция (Рис. 5, Приложение 1) ищет поезд по его номеру и возвращает его. Иначе возвращается nullptr.

```
ПОЕЗД *СвязанныйСписок::Поиск(ЧИСЛО номер_поезда)
{
    // Поиск ПОЕЗД по номер_поезда
    указатель_на_текущий_элемент = указатель_на_первый_элемент;
    if(указатель_на_текущий_элемент == nullptr || количество_элементов == 0) { return
nullptr; }

    while (указатель_на_текущий_элемент->следующий != nullptr)
    {
        if (указатель_на_текущий_элемент->номер_поезда == номер_поезда)
        {
            return указатель_на_текущий_элемент;
        }
        указатель_на_текущий_элемент = указатель_на_текущий_элемент->следующий;
    }

    if(указатель_на_текущий_элемент->номер_поезда == номер_поезда){ return указа-
тель_на_текущий_элемент; }
```

```
    else { return nullptr; }
}
```

Изменение записи

Функция (Рис. 6, Приложение 1) принимает измененный элемент и номер поезда, который изменялся. Затем функция ищет этот элемент в списке и если не находит, выводит ошибку.

```
void СвязанныйСписок::Изменить(ПОЕЗД новый_элемент, ЧИСЛО позиция)
{
    ПОЕЗД *элемент;
    элемент = Поиск(позиция);

    if (элемент == nullptr)
    {
        return;
    }

    элемент->номер_поезда = новый_элемент.номер_поезда;
    элемент->time = новый_элемент.time;
    ЧИСЛО i = 0;
    for(auto e : элемент->пункт_назначения)
    {
        элемент->пункт_назначения[i] = новый_элемент.пункт_назначения[i];
        i++;
    }
}
```

Сортировка записей

Функция (Рис. 7, Приложение 1) сортирует список по возрастанию номера поезда.

```
void СвязанныйСписок::Обмен(ПОЕЗД *l, ПОЕЗД *r)
{
    ПОЕЗД *nex = r->следующий;
    ПОЕЗД *указатель на последний элемент = l->предыдущий;

    if (l == указатель_на_первый_элемент)
    {
        if (указатель на последний элемент != nullptr) { указатель на последний элемент-
>следующий = r; }
        l->следующий = nex;
        l->предыдущий = r;
        r->следующий = l;
        r->предыдущий = указатель на последний элемент;
        if (nex != nullptr) { nex->предыдущий = l; }
        указатель_на_первый_элемент = r;
    }
    else if(r == указатель на последний элемент)
    {
        if (указатель на последний элемент != nullptr) { указатель на последний элемент-
>следующий = r; }
        l->следующий = nex;
        l->предыдущий = r;
        r->следующий = l;
        r->предыдущий = указатель на последний элемент;
        if (nex != nullptr) { nex->предыдущий = l; }
        указатель на последний элемент = l;
    }
    else
    {
        if (указатель на последний элемент != nullptr) { указатель на последний элемент-
>следующий = r; }
        l->следующий = nex;
```

```
        l->предыдущий = r;
        r->следующий = l;
        r->предыдущий = указатель на последний элемент;
        if (nex != nullptr) { nex->предыдущий = l; }
    }
}

void СвязанныйСписок::СортировкаПузырьком()
{
    bool Обмены = 1;
    while (Обмены){
        Обмены = false;
        for (ПОЕЗД *ptr = указатель_на_первый_элемент; ptr != nullptr && ptr->следующий
!= nullptr; ptr = ptr->следующий) {
            if ( ptr->номер_поезда > ptr->следующий->номер_поезда )
            {
                Обмен(ptr, ptr->следующий);
                Обмены = true;
            }
        }
    }
}
```

Сохранение списка

Функция (Рис. 8, Приложение 1) сохраняет список в текстовый файл.

```
void СвязанныйСписок::Сохранить()
{
    wofstream fout("out.txt");  // выходной поток

    Объект.указатель_на_текущий_элемент = Объект.указатель_на_первый_элемент;
    ЧИСЛО позиция = Объект.количество_элементов;
    while (Объект.указатель_на_текущий_элемент != nullptr && позиция > 0)
    {
        fout << Объект.указатель_на_текущий_элемент->номер_поезда << endl;
        fout << Объект.указатель_на_текущий_элемент->пункт_назначения << endl;
        fout << Объект.указатель_на_текущий_элемент->время << endl;
        Объект.указатель_на_текущий_элемент = Объект.указатель_на_текущий_элемент->следу-
ющий;
        позиция--;
    }
    fout << L"\n";

    fout.close();
}
```

Чтение списка

Функция (Рис. 9, Приложение 1) добавляет в список элементы, сохраненные в файл.

```
void СвязанныйСписок::Загрузить()
{
    wifstream fin("in.txt");   // входной поток

    while (!fin.eof())
    {
        ПОЕЗД* новый_элемент = new ПОЕЗД;
        fin >> новый_элемент->номер_поезда;
        fin >> новый_элемент->пункт_назначения;
        fin >> новый_элемент->time;
        Объект.Добавить(новый_элемент);
    }
    Объект.Удалить();
```

```
    fin.close();
}
```

Печать списка

Функция (Рис. 10, Приложение 1) выводит на экран содержимое массива в виде таблицы

```
void СвязанныйСписок::Печать()
{
    указатель_на_текущий_элемент = указатель_на_первый_элемент;
    ЧИСЛО позиция = количество_элементов;
    wcout << L"\n";
    wcout << L"╔═══════╦═══════════╦═══════════════════════════════════════╦═══════════╗ \n";
    wcout << L"║  id   ║ номер_поезда ║        пункт_назначения          ║   time    ║ \n";
    while (указатель_на_текущий_элемент != nullptr && позиция != 0)
    {
        wcout << L"╠═══════╬═══════════╬═══════════════════════════════════════╬═══════════╣ \n";
        выровнять_по_левой_стороне; wcout << L"╠ " << установить_ширину_поля(5)  << коли-
чество_элементов - позиция;
        выровнять_по_левой_стороне; wcout << L"╠ " << установить_ширину_поля(9)  << ука-
затель_на_текущий_элемент->номер_поезда;
        repair_rus_symbol_str(указатель_на_текущий_элемент->пункт_назначения);
        выровнять_по_левой_стороне; wcout << L"╠ " << установить_ширину_поля(26) << ука-
затель_на_текущий_элемент->пункт_назначения;
        выровнять_по_левой_стороне; wcout << L"╠ " << установить_ширину_поля(9)  << ука-
затель_на_текущий_элемент->time;
                                    wcout << L"╠\n" ;
        указатель_на_текущий_элемент = указатель_на_текущий_элемент->следующий;
        позиция--;
    }
    wcout << L"╚═══════╩═══════════╩═══════════════════════════════════════╩═══════════╝ \n";
    wcout << L"\n";
}
```

Пример выполнения (Рис. 11, Приложение 1)

4. Описание пользовательского интерфейса

При запуске программы на экран выводится консольное приложение с меню пользователя. При нажатии на клавиатуре на определенную цифру выполняется соответствующая функция.

Пример пользовательского интерфейса (Рис. 12 и Рис 13, Приложение 1)

5. Результаты тестирования программы

При тестировании была проверена работа программы, а также надежность проверок пользовательского ввода.

Данные для тестирования

1

7

7

2

111

SPB

100

-12

10

100

-12

10

100

-12

10

7

3

222

MSK

-2

24

23

-1

60

20

-1

60

20

-5

5

7

4

8

7

4

10

4

-10

0

7

5

-10

50

5

2

6

190

6

1212

8

7

9

1

2

4

2

400

3

AAA

4

1

5

5

6

10

0

7

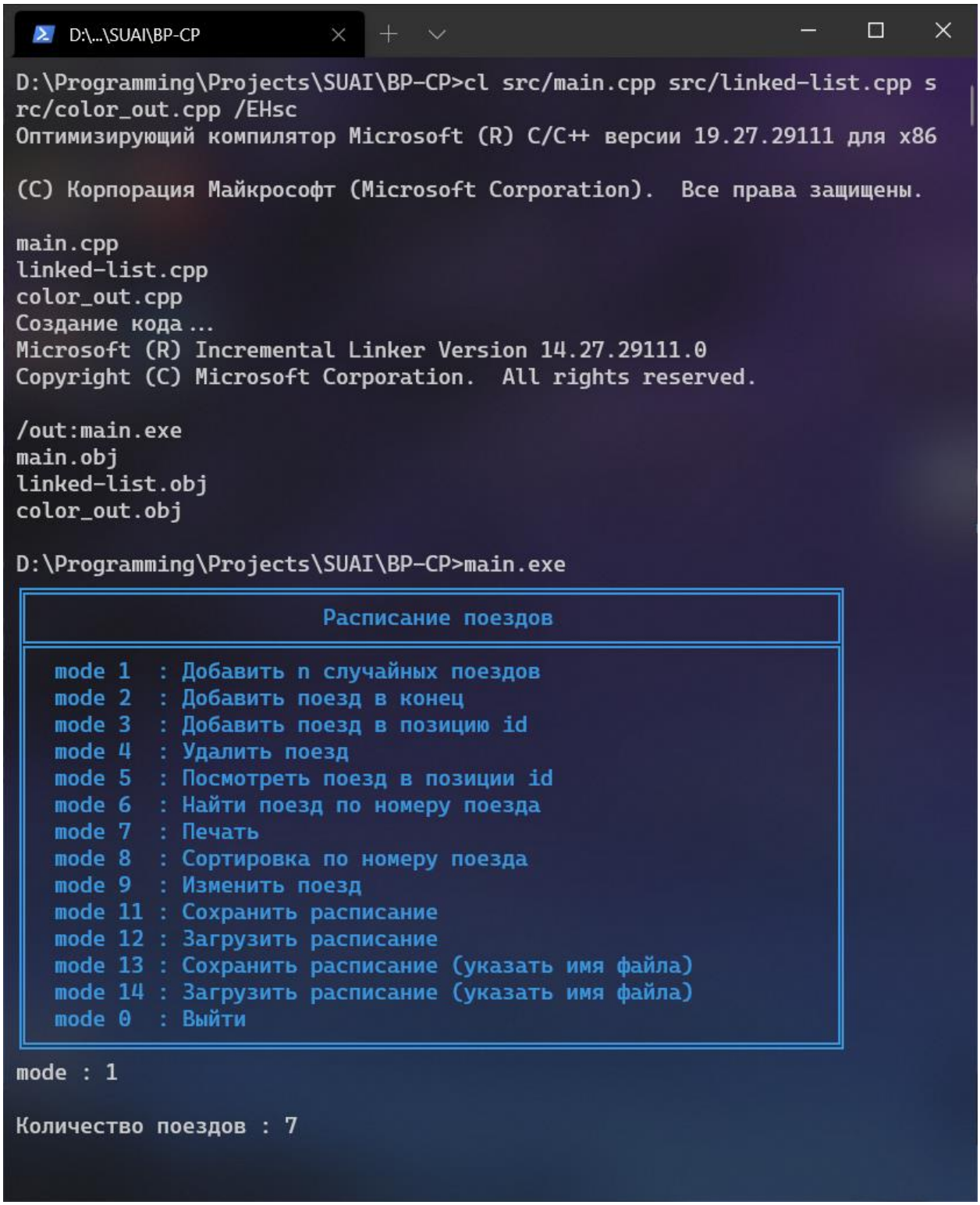
11

12

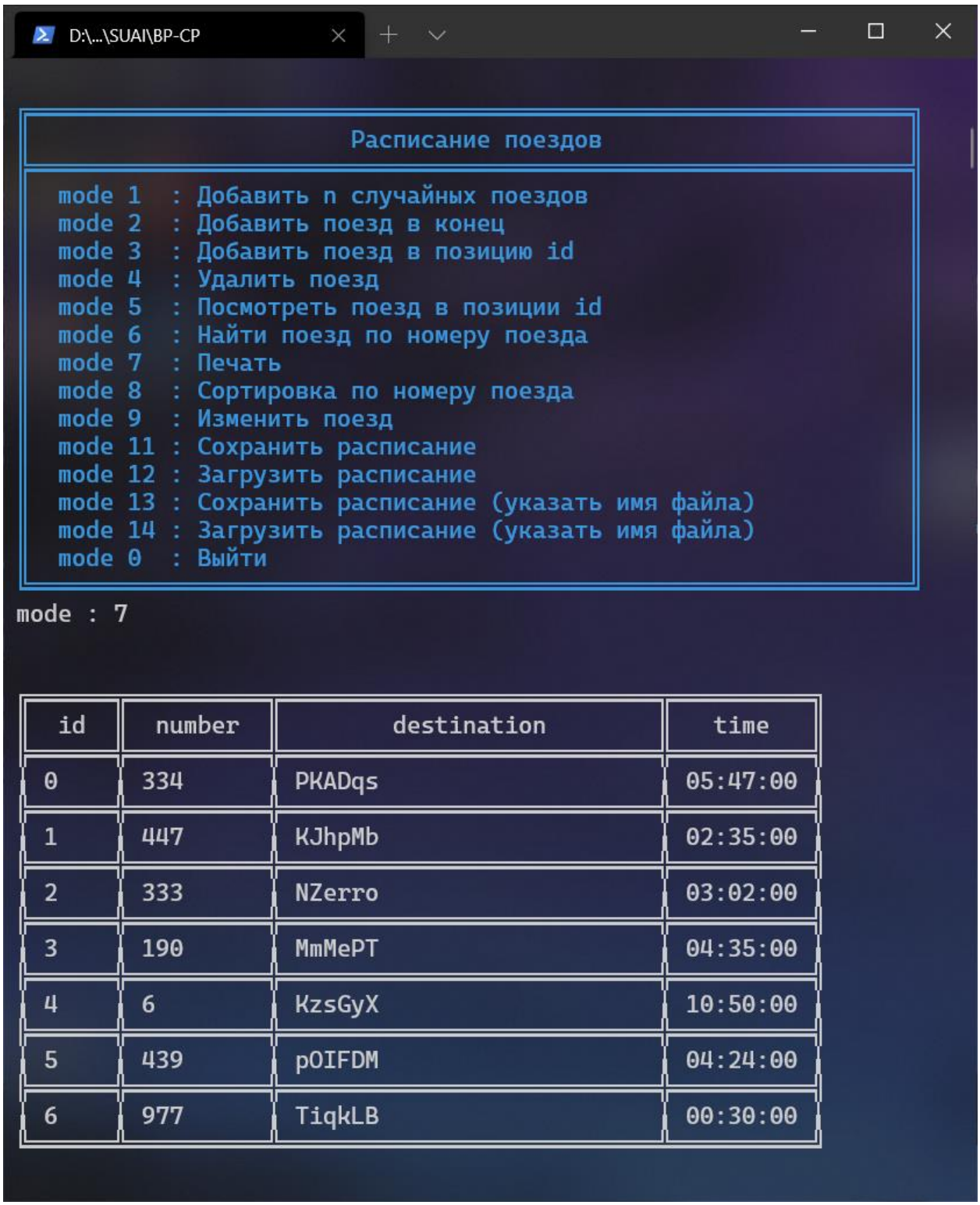
7

0

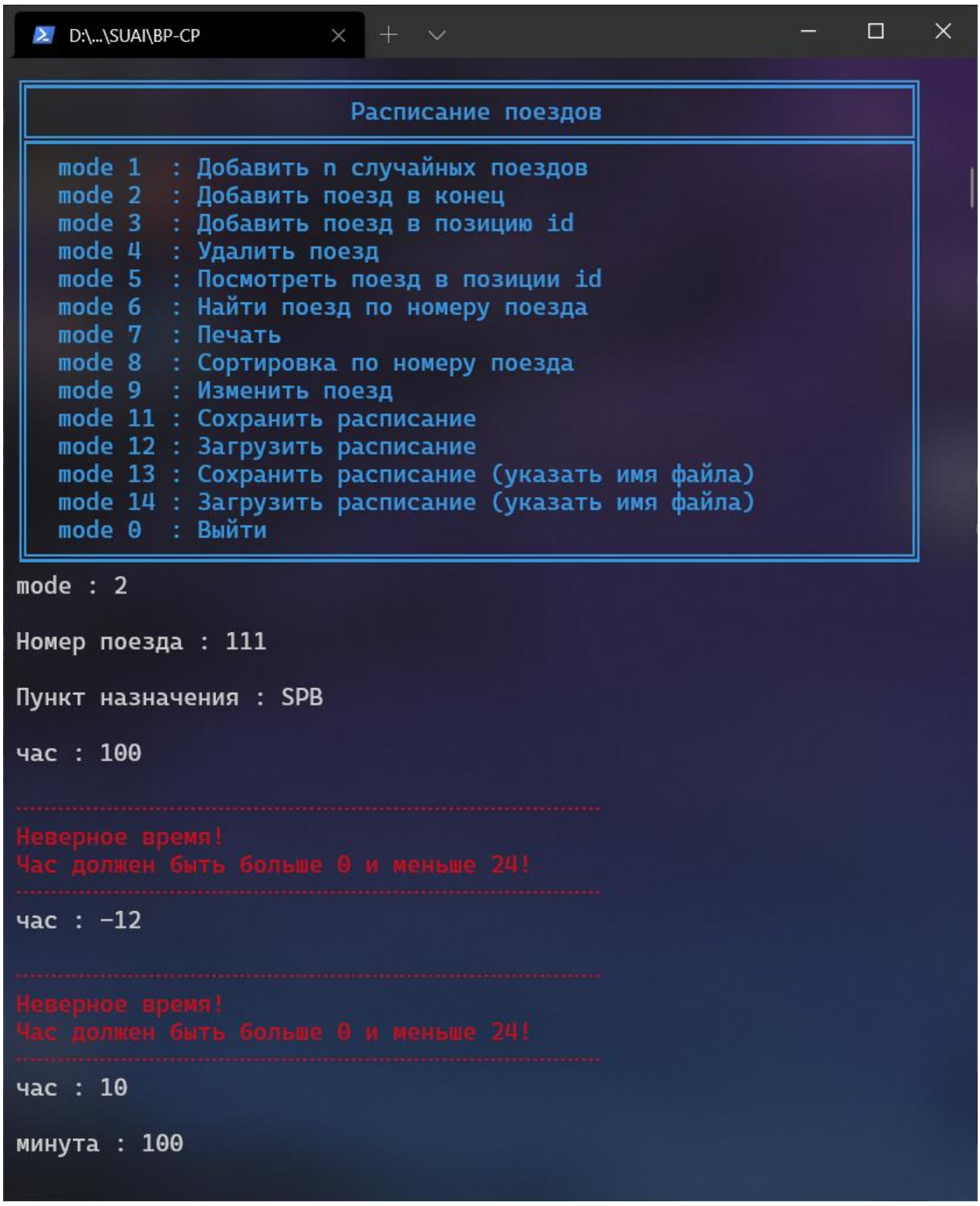
Ход тестирования



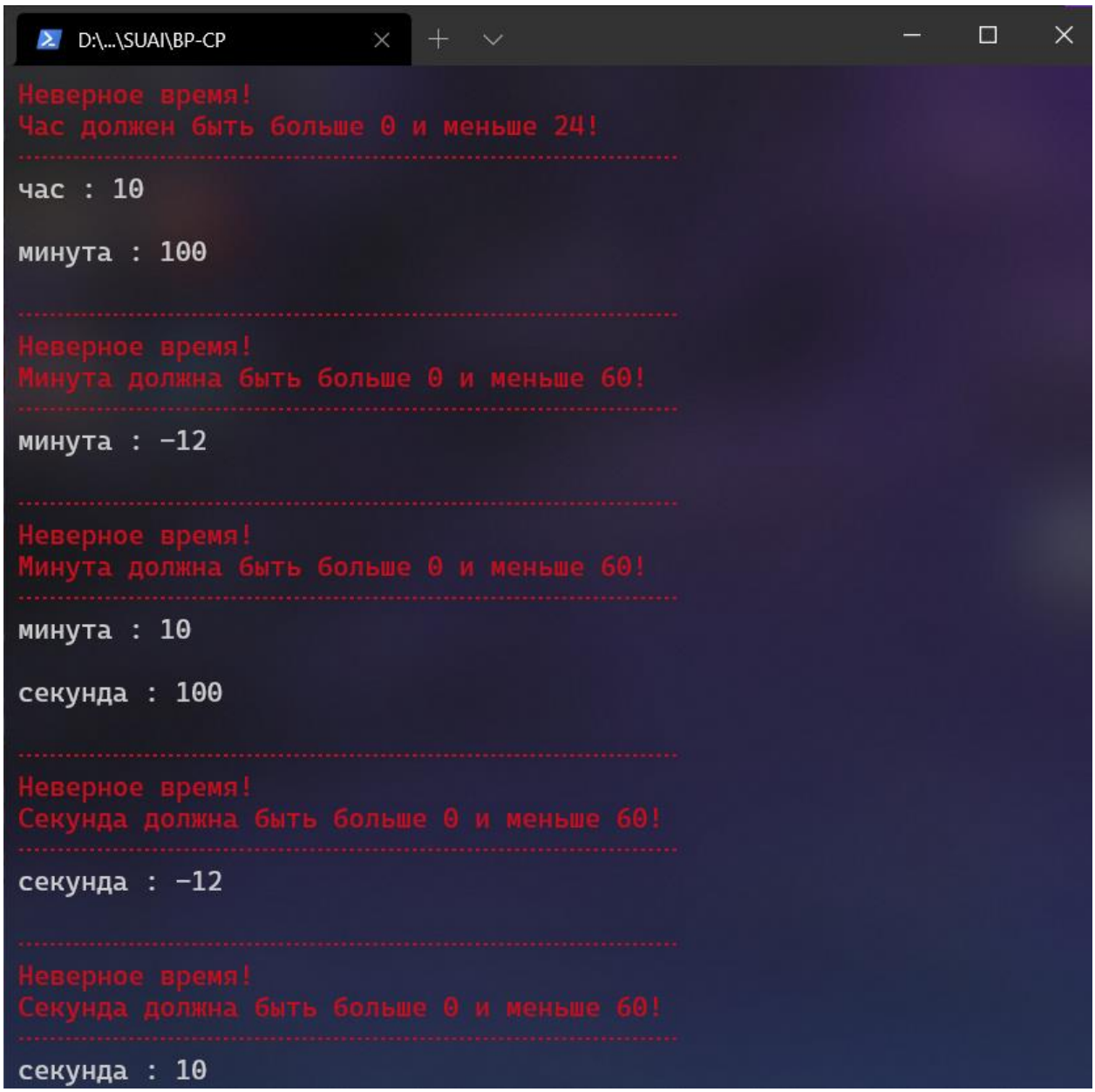
Создание случайного списка из 7 поездов



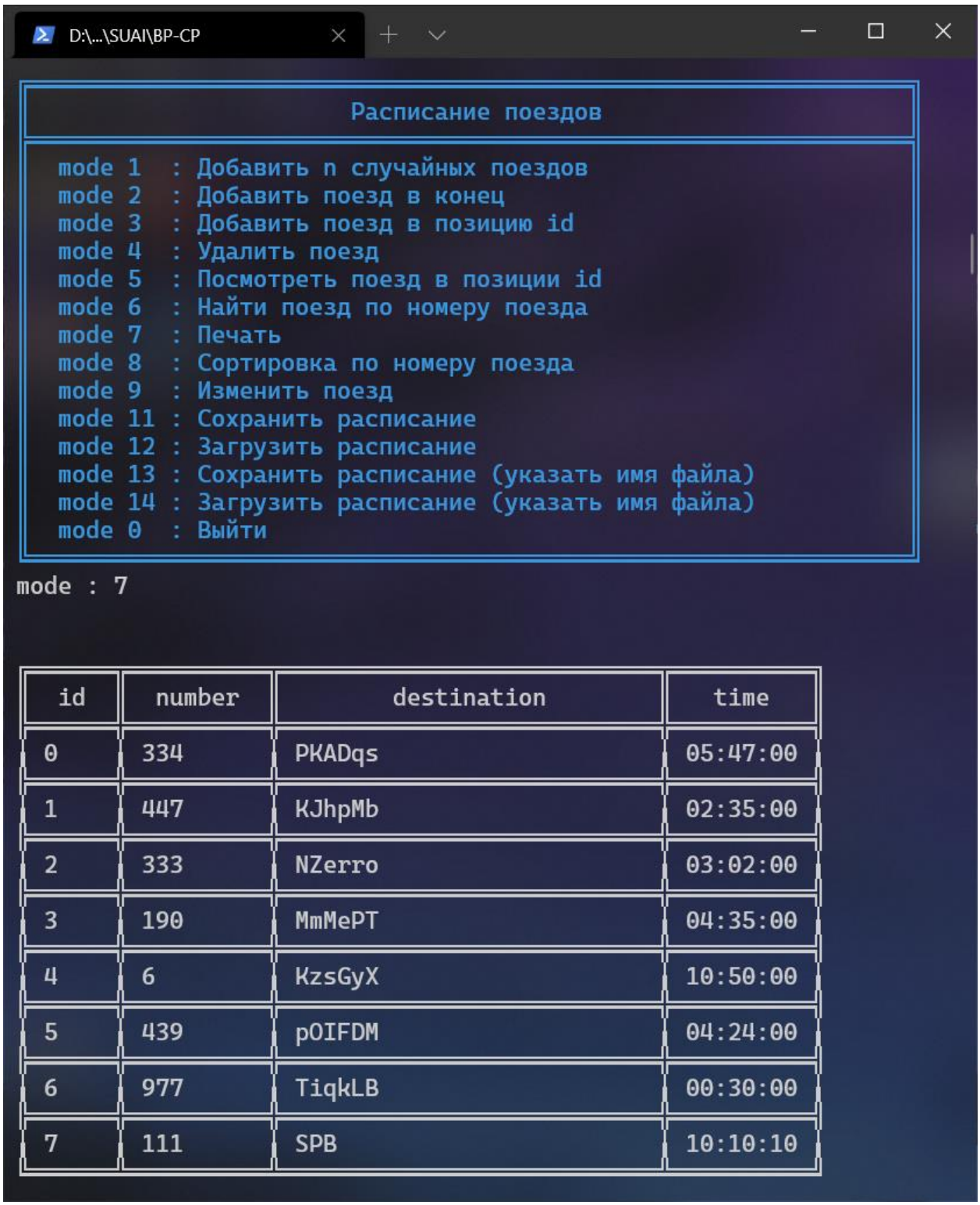
Вывод списка на экран



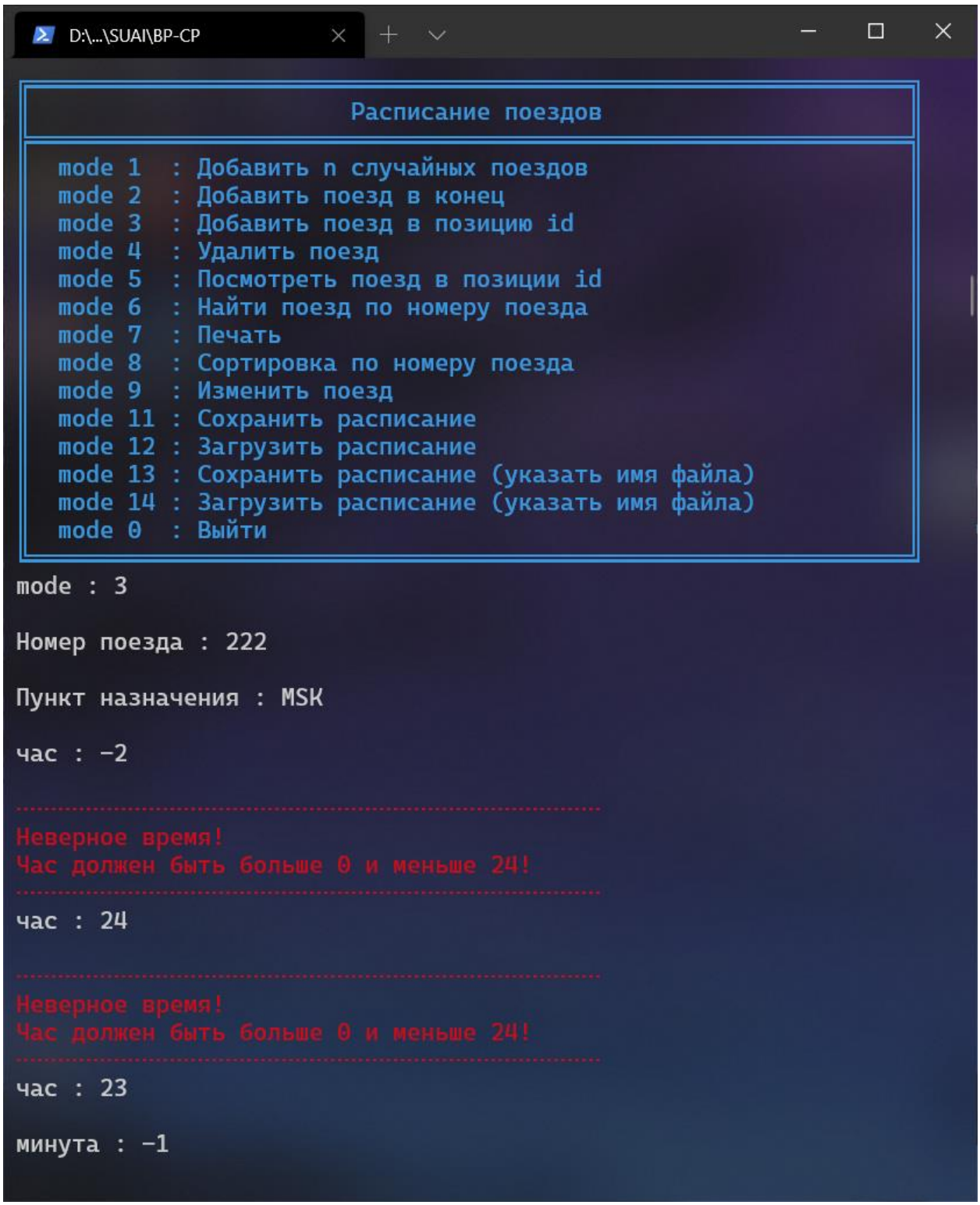
Создание нового элемента и попытка ввести неверные данные.



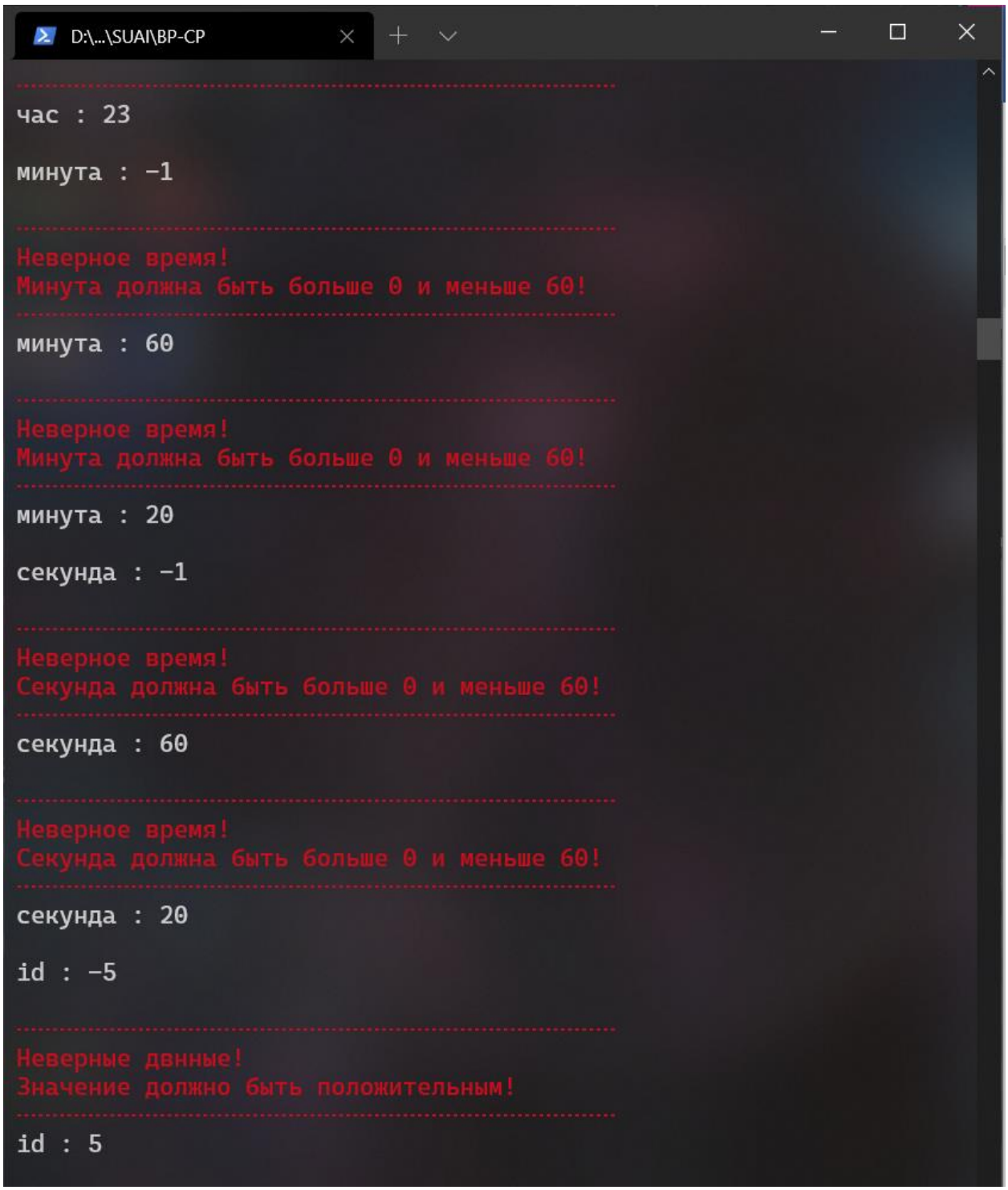
Создание нового элемента и попытка ввести неверные данные. (Продолжение)



Вывод на экран списка с новым элементом



Добавление элемента в список



Добавление элемента в список (продолжение)

```
mode : 7
```

id	number	destination	time
0	334	PKADqs	05:47:00
1	447	KJhpMb	02:35:00
2	333	NZerro	03:02:00
3	190	MmMePT	04:35:00
4	6	KzsGyX	10:50:00
5	222	MSK	23:20:20
6	439	pOIFDM	04:24:00
7	977	TiqkLB	00:30:00
8	111	SPB	10:10:10

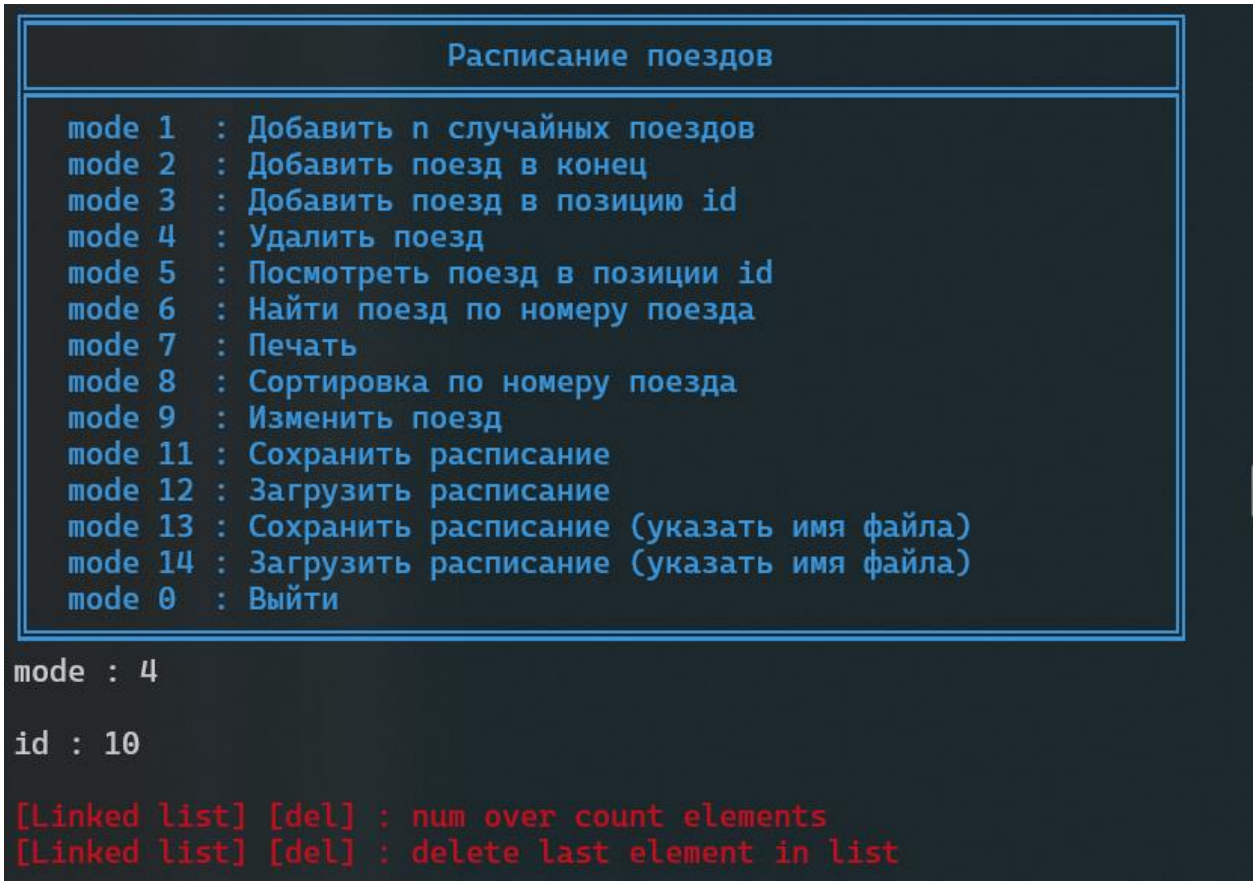
Список с добавленным элементом

```
                Расписание поездов

  mode 1  : Добавить n случайных поездов
  mode 2  : Добавить поезд в конец
  mode 3  : Добавить поезд в позицию id
  mode 4  : Удалить поезд
  mode 5  : Посмотреть поезд в позиции id
  mode 6  : Найти поезд по номеру поезда
  mode 7  : Печать
  mode 8  : Сортировка по номеру поезда
  mode 9  : Изменить поезд
  mode 11 : Сохранить расписание
  mode 12 : Загрузить расписание
  mode 13 : Сохранить расписание (указать имя файла)
  mode 14 : Загрузить расписание (указать имя файла)
  mode 0  : Выйти
mode : 4

id : 8
```

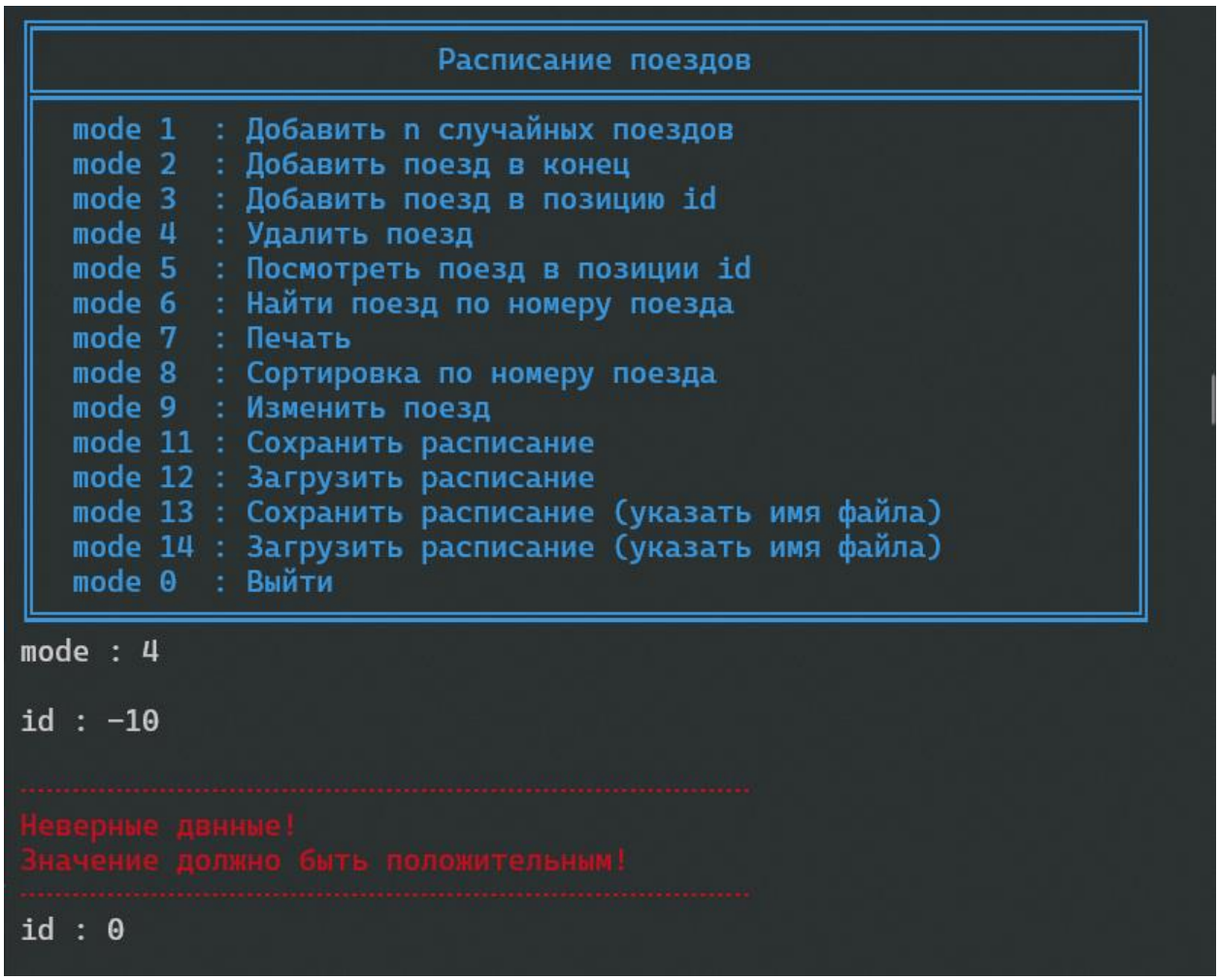
Удаление элемента по индексу

id	number	destination	time
0	334	PKADqs	05:47:00
1	447	KJhpMb	02:35:00
2	333	NZerro	03:02:00
3	190	MmMePT	04:35:00
4	6	KzsGyX	10:50:00
5	222	MSK	23:20:20
6	439	pOIFDM	04:24:00
7	977	TiqkLB	00:30:00

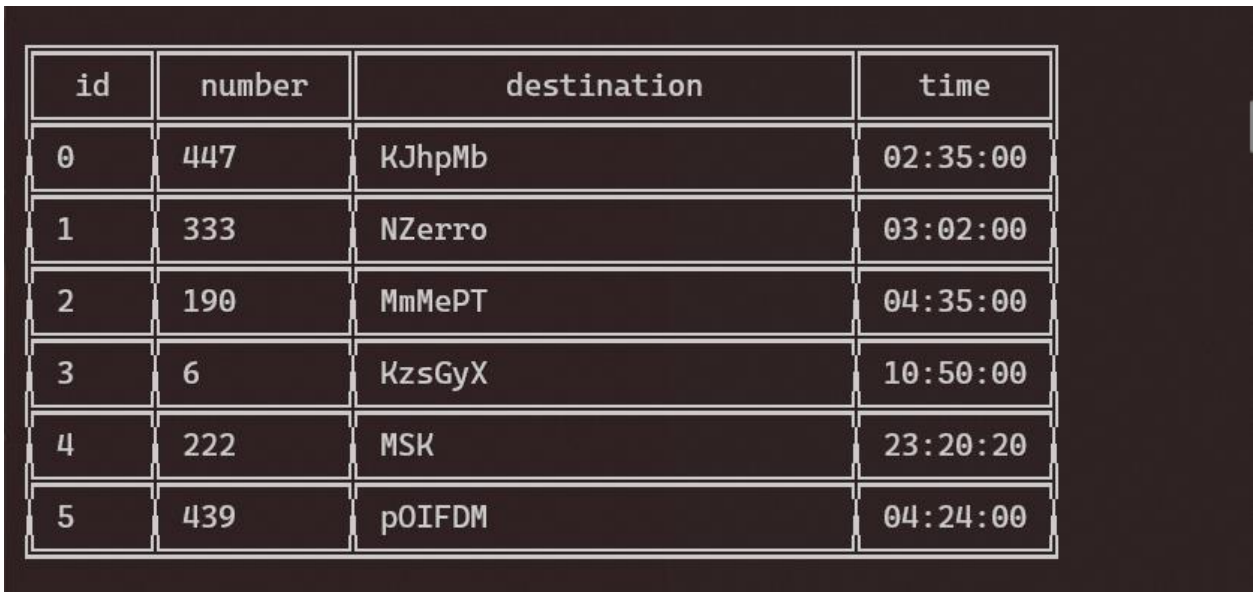
Демонстрация списка после удаления



Попытка удалить элемент, индекс которого больше максимального



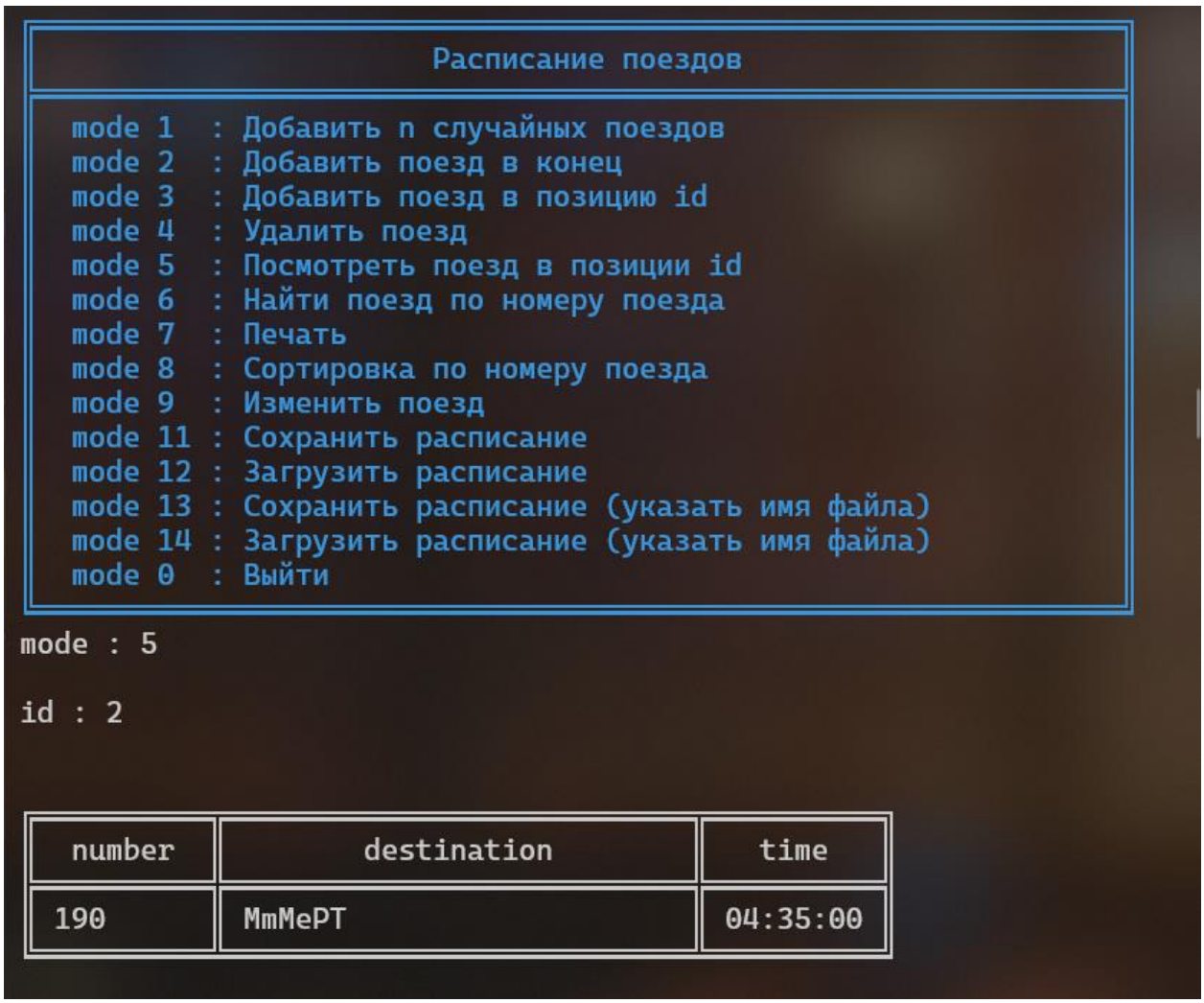
Попытка удалить элемент по отрицательному индексу и удаление первого элемента



id	number	destination	time
0	447	KJhpMb	02:35:00
1	333	NZerro	03:02:00
2	190	MmMePT	04:35:00
3	6	KzsGyX	10:50:00
4	222	MSK	23:20:20
5	439	pOIFDM	04:24:00

Распечатка списка после попыток удаления

Тестирование проверок пользовательского ввода



Получение элемента из списка

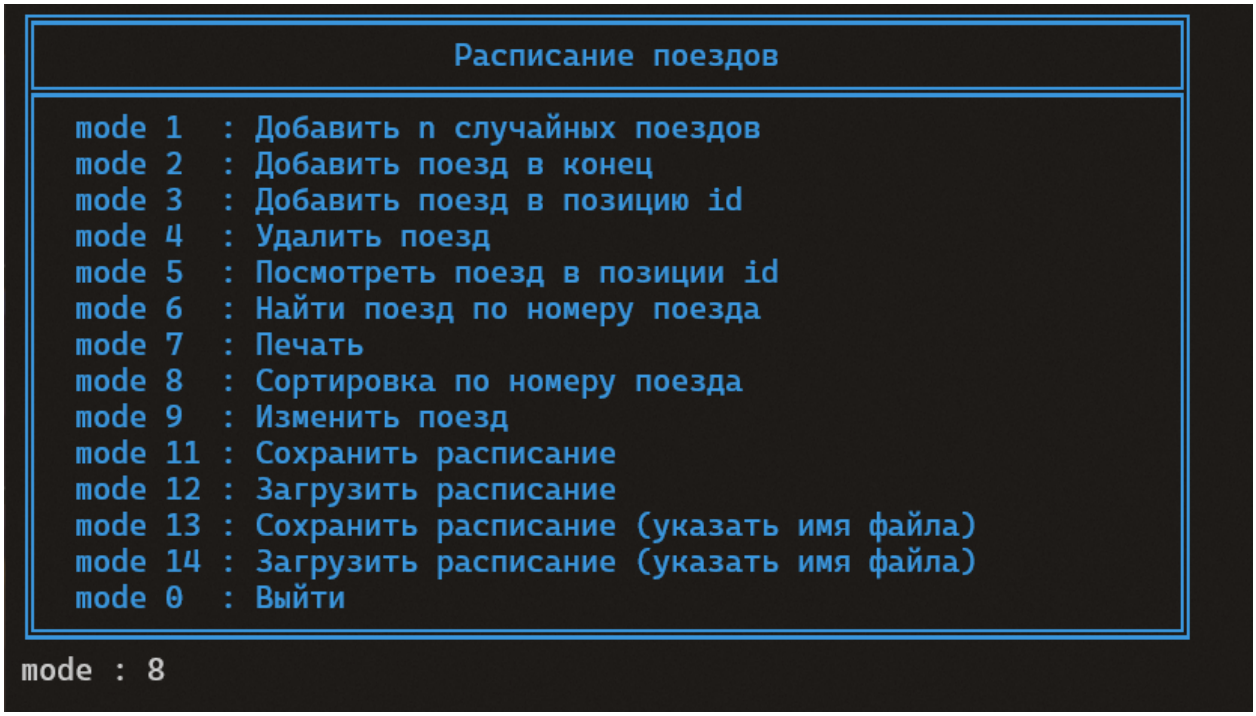
```
                    Расписание поездов

  mode 1  : Добавить n случайных поездов
  mode 2  : Добавить поезд в конец
  mode 3  : Добавить поезд в позицию id
  mode 4  : Удалить поезд
  mode 5  : Посмотреть поезд в позиции id
  mode 6  : Найти поезд по номеру поезда
  mode 7  : Печать
  mode 8  : Сортировка по номеру поезда
  mode 9  : Изменить поезд
  mode 11 : Сохранить расписание
  mode 12 : Загрузить расписание
  mode 13 : Сохранить расписание (указать имя файла)
  mode 14 : Загрузить расписание (указать имя файла)
  mode 0  : Выйти

mode : 6

Номер поезда : 190
```

number	destination	time
190	MmMePT	04:35:00

Поиск поезда

```
                    Расписание поездов

  mode 1  : Добавить n случайных поездов
  mode 2  : Добавить поезд в конец
  mode 3  : Добавить поезд в позицию id
  mode 4  : Удалить поезд
  mode 5  : Посмотреть поезд в позиции id
  mode 6  : Найти поезд по номеру поезда
  mode 7  : Печать
  mode 8  : Сортировка по номеру поезда
  mode 9  : Изменить поезд
  mode 11 : Сохранить расписание
  mode 12 : Загрузить расписание
  mode 13 : Сохранить расписание (указать имя файла)
  mode 14 : Загрузить расписание (указать имя файла)
  mode 0  : Выйти

mode : 6

Номер поезда : 1212

Не найден
```

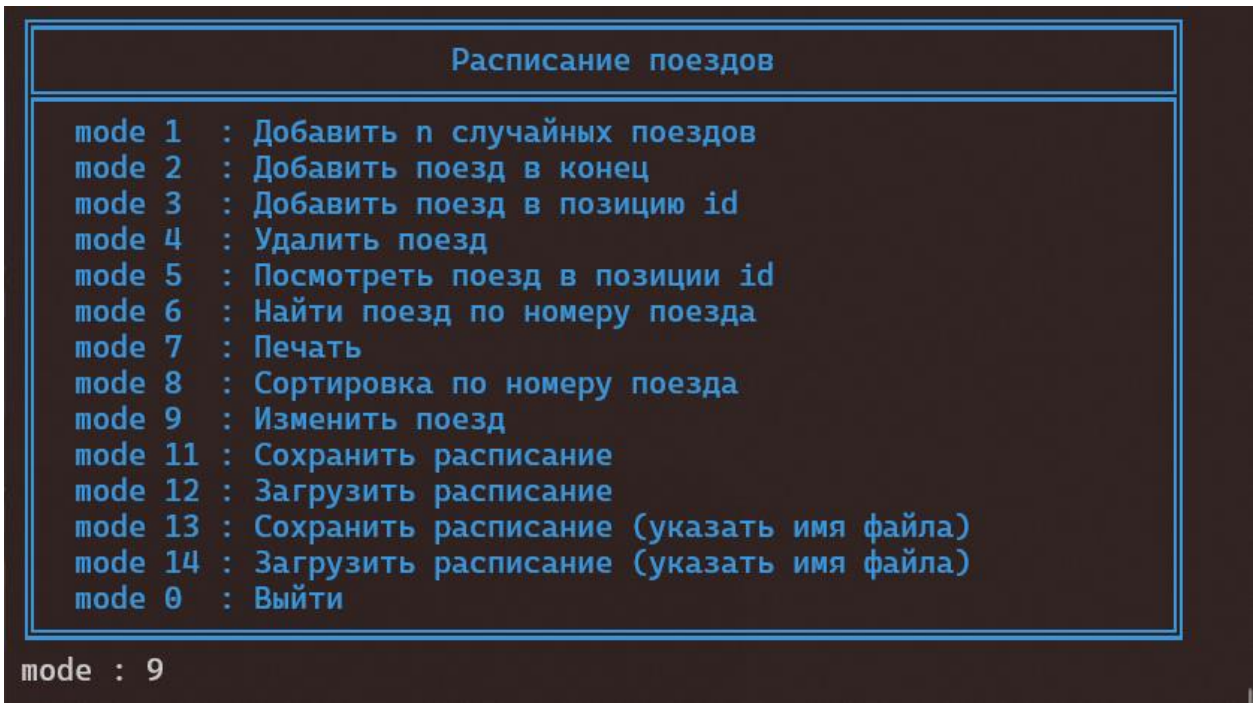
Поиск несуществующего поезда



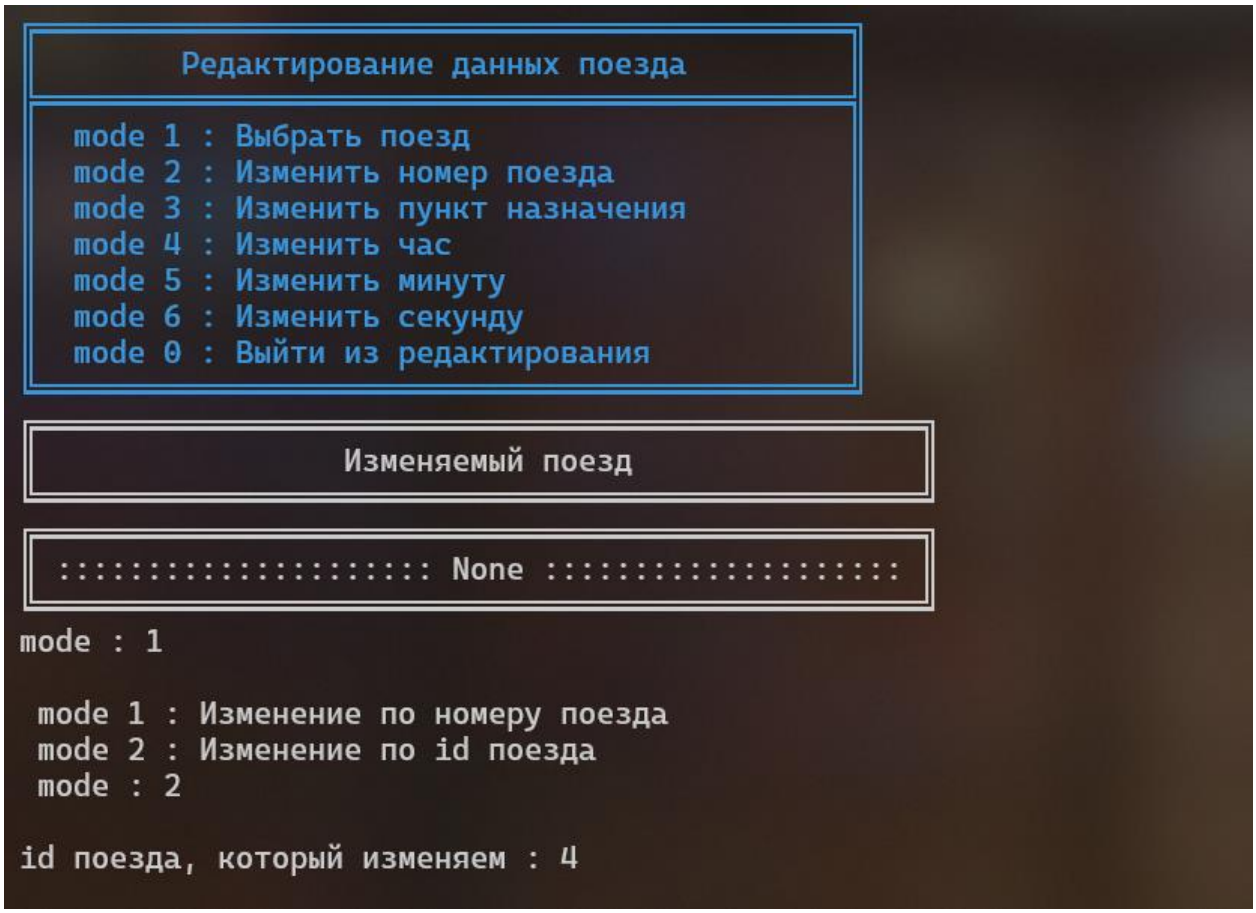
Сортировка списка

id	number	destination	time
0	6	KzsGyX	10:50:00
1	190	MmMePT	04:35:00
2	222	MSK	23:20:20
3	333	NZerro	03:02:00
4	439	pOIFDM	04:24:00
5	447	KJhpMb	02:35:00

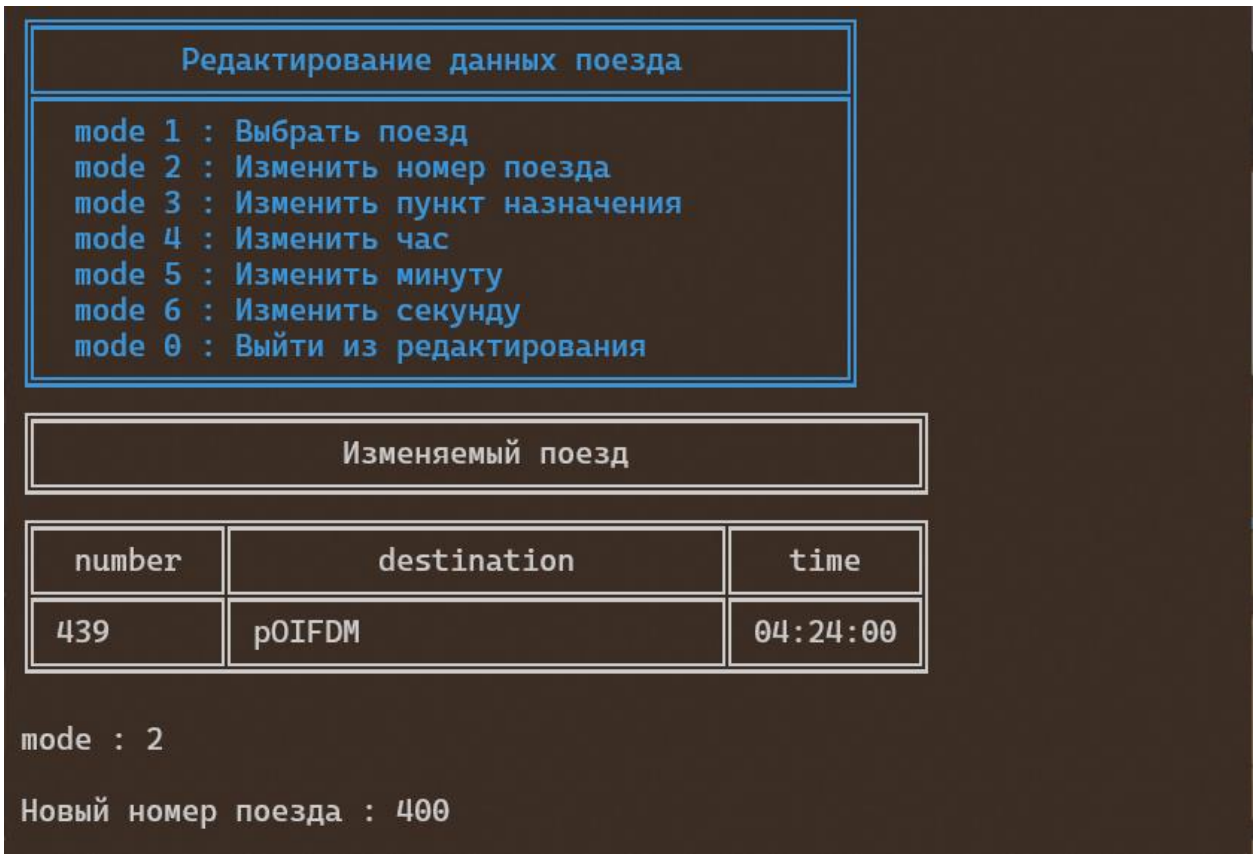
Распечатка отсортированного списка



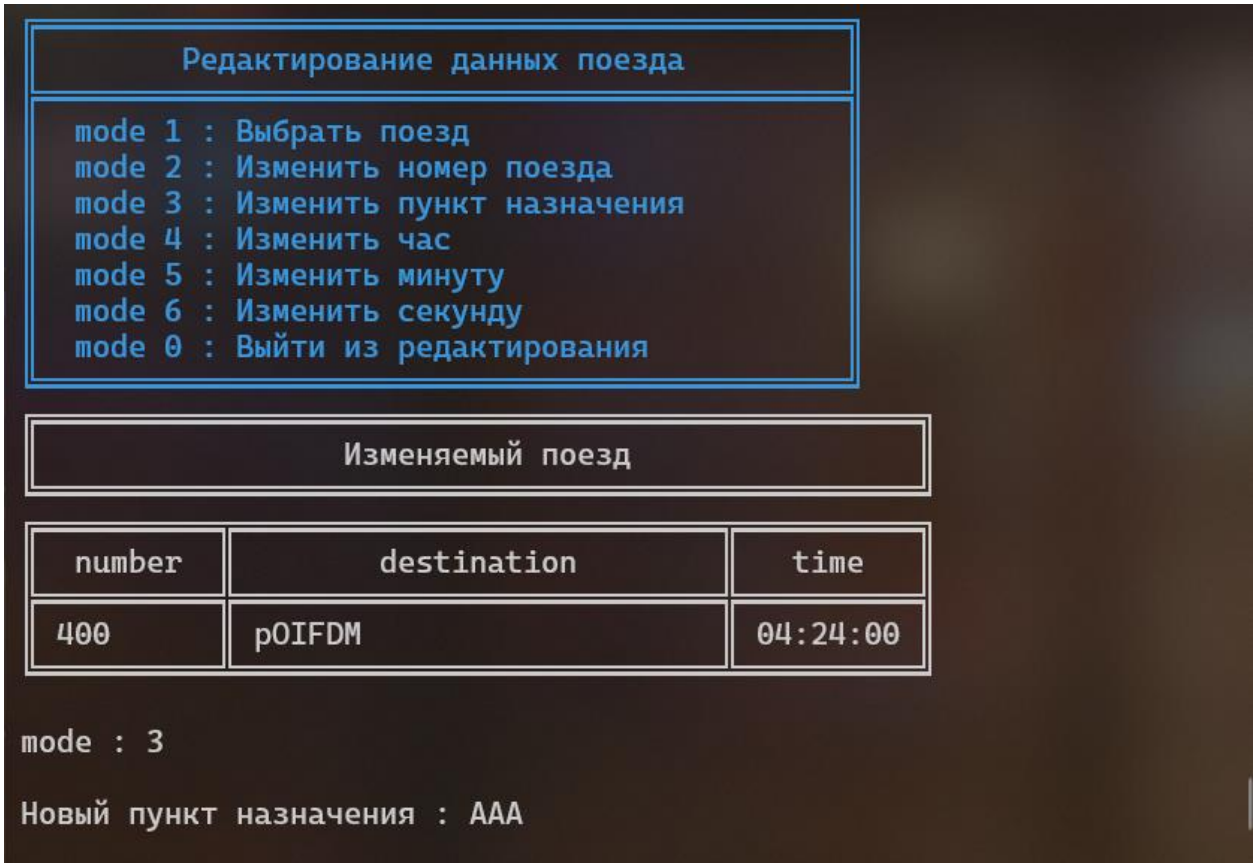
Вход в режим редактирования поезда



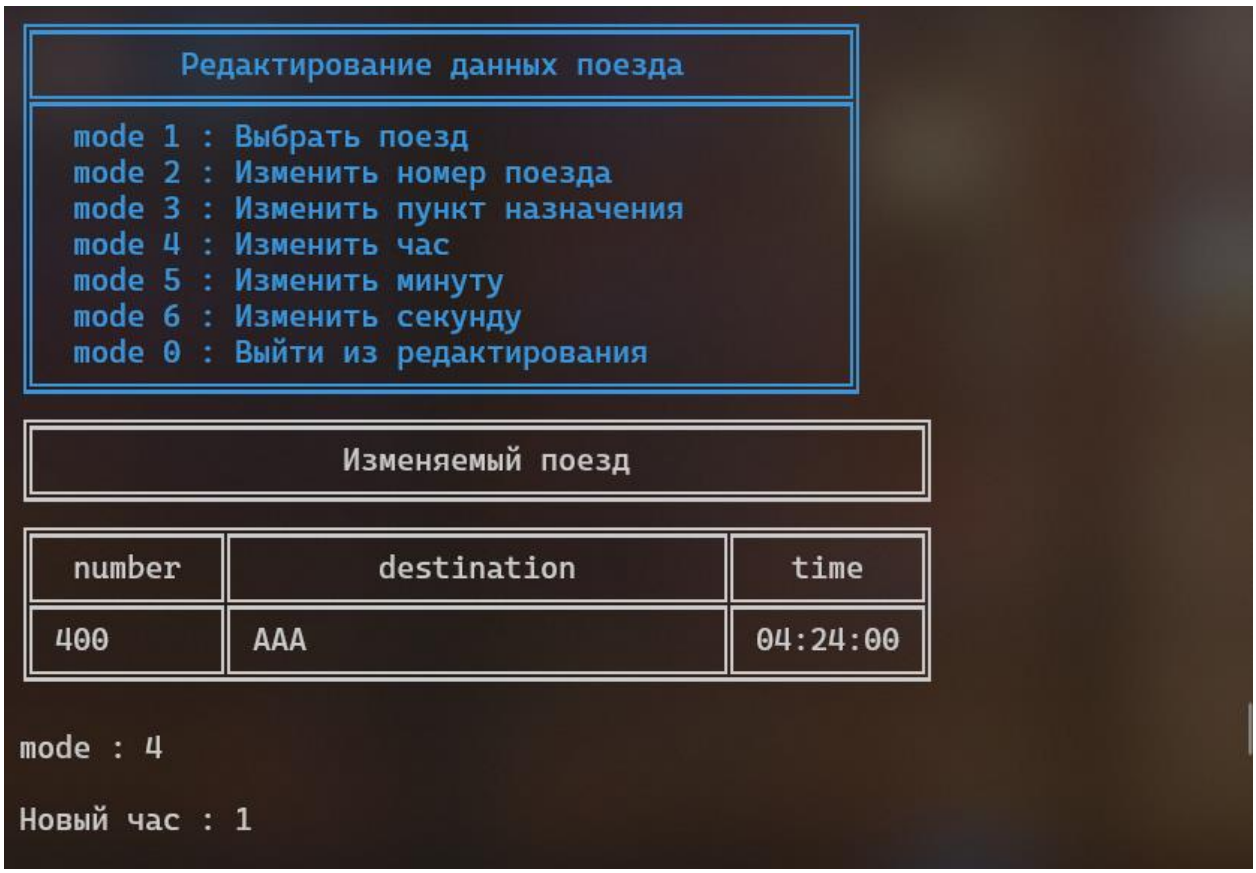
Выбор поезда для редактирования



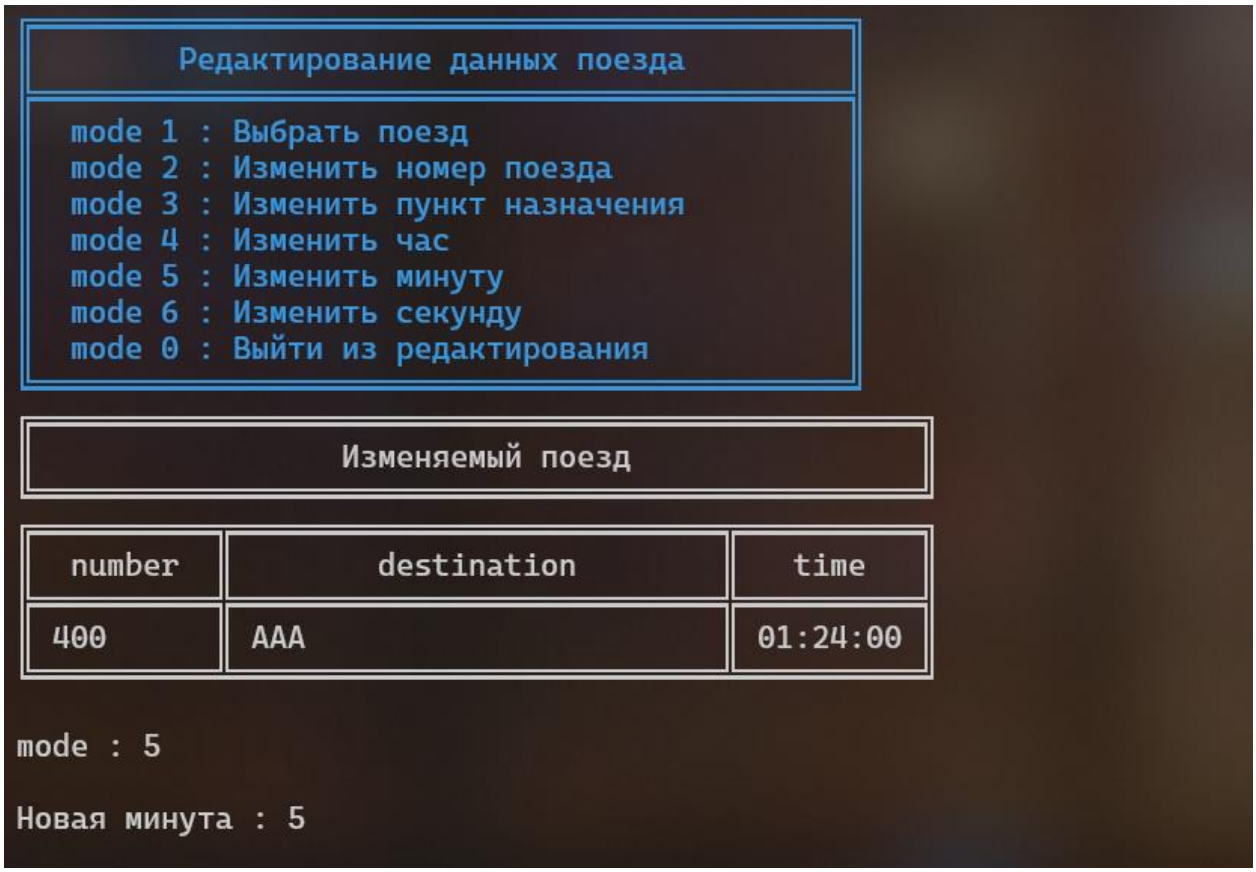
Изменение номера поезда



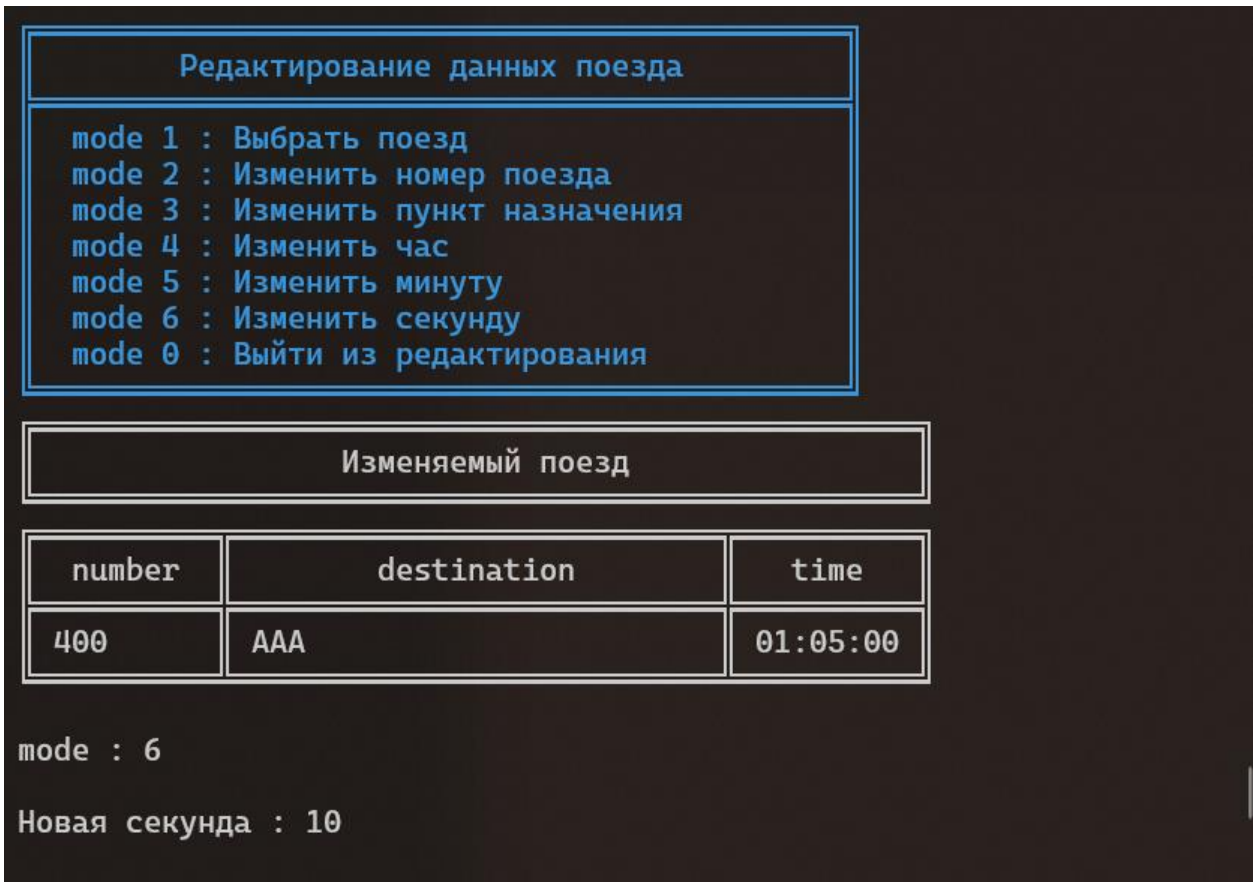
Изменение Пункта назначения для поезда



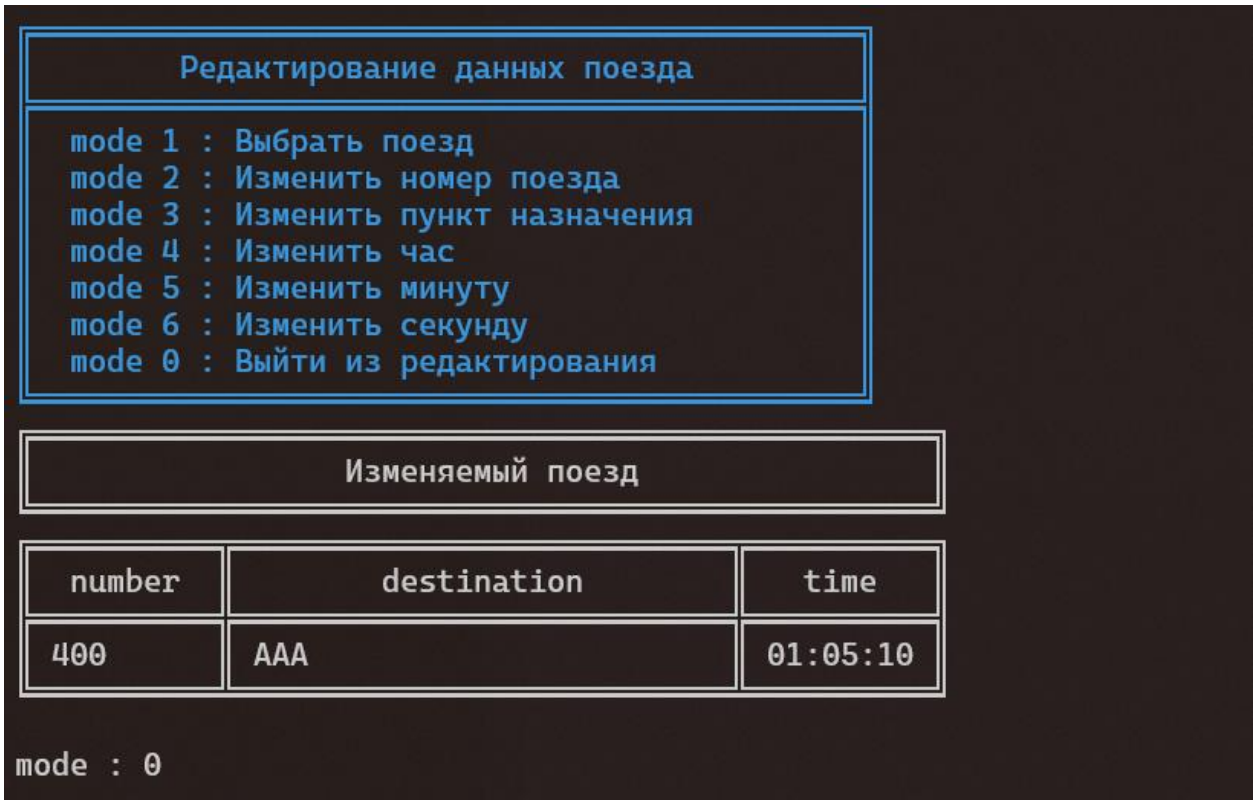
Изменение времени для поезда



Изменение времени для поезда (Продолжение)



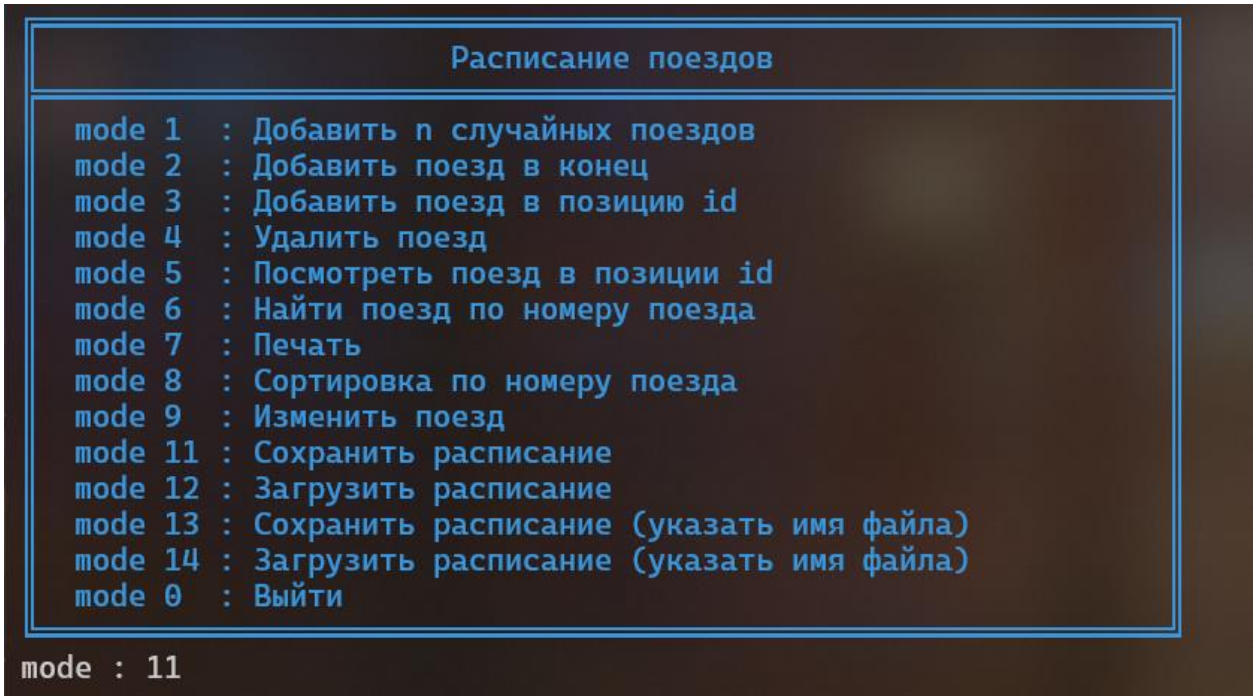
Изменение времени для поезда (Продолжение)



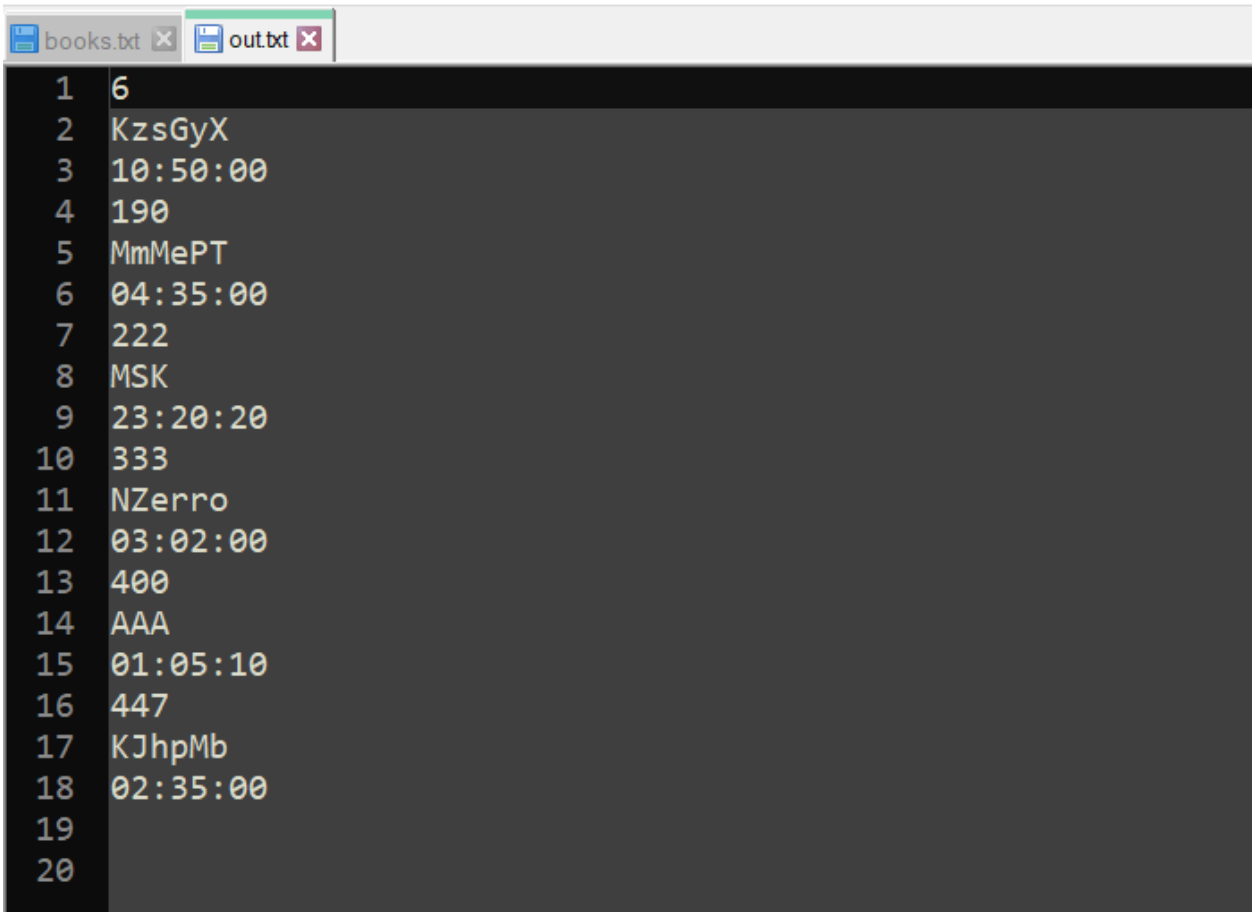
Выход из режима редактирования

id	number	destination	time
0	6	KzsGyX	10:50:00
1	190	MmMePT	04:35:00
2	222	MSK	23:20:20
3	333	NZerro	03:02:00
4	400	AAA	01:05:10
5	447	KJhpMb	02:35:00

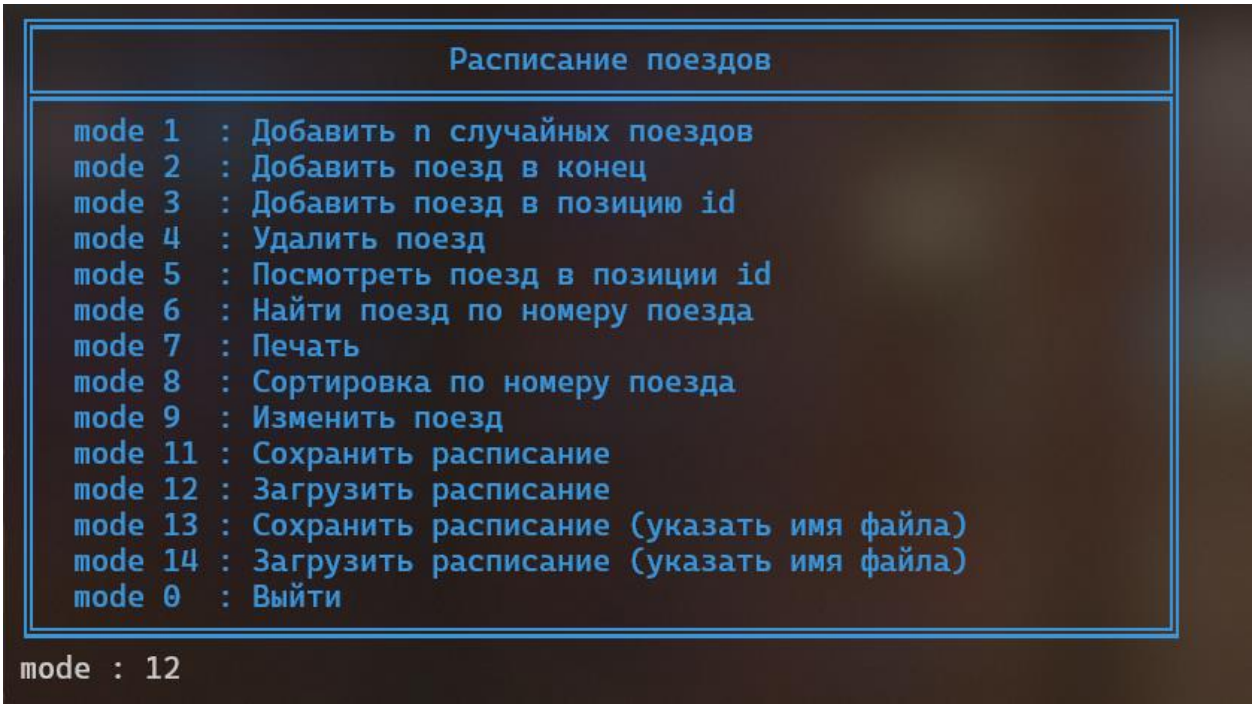
Распечатка списка с отредактированным элементом



Сохранение списка в файл



Демонстрация сохраненного файла



Загрузка списка из файла

id	number	destination	time
0	6	KzsGyX	10:50:00
1	190	MmMePT	04:35:00
2	222	MSK	23:20:20
3	333	NZerro	03:02:00
4	400	AAA	01:05:10
5	447	KJhpMb	02:35:00
6	1	1	01:01:01
7	2	2	02:02:02
8	5	5	05:05:05
9	31	WuIOzN	03:34:00
10	190	MmMePT	04:35:00
11	209	myKoRP	05:29:00
12	333	NZerro	03:02:00
13	439	pOIFDM	04:24:00
14	447	KJhpMb	02:35:00
15	977	TiqkLB	00:30:00

Распечатка загруженных из файла значений

Заключение

Подводя итоги, можно сказать, что программа имеет

- Недостатки
    - Неудачно выбран тип данных для number (Отрицательные значения не задействованы)
    - Можно было бы написать сортировку эффективнее, чем пузырьковая
    - Прихотливый интерфейс (не все консоли/терминалы поддерживают управляющие последовательности (здесь используются для цвета))
    - Возникли проблемы с кириллицей при использовании юникода (Хоть проблема и была решена)
- Достоинства

- Хорошая обработка вводимых пользователем данных
- Удобный режим редактирования поезда

Приложение 1 Блок-схемы и интерфейс

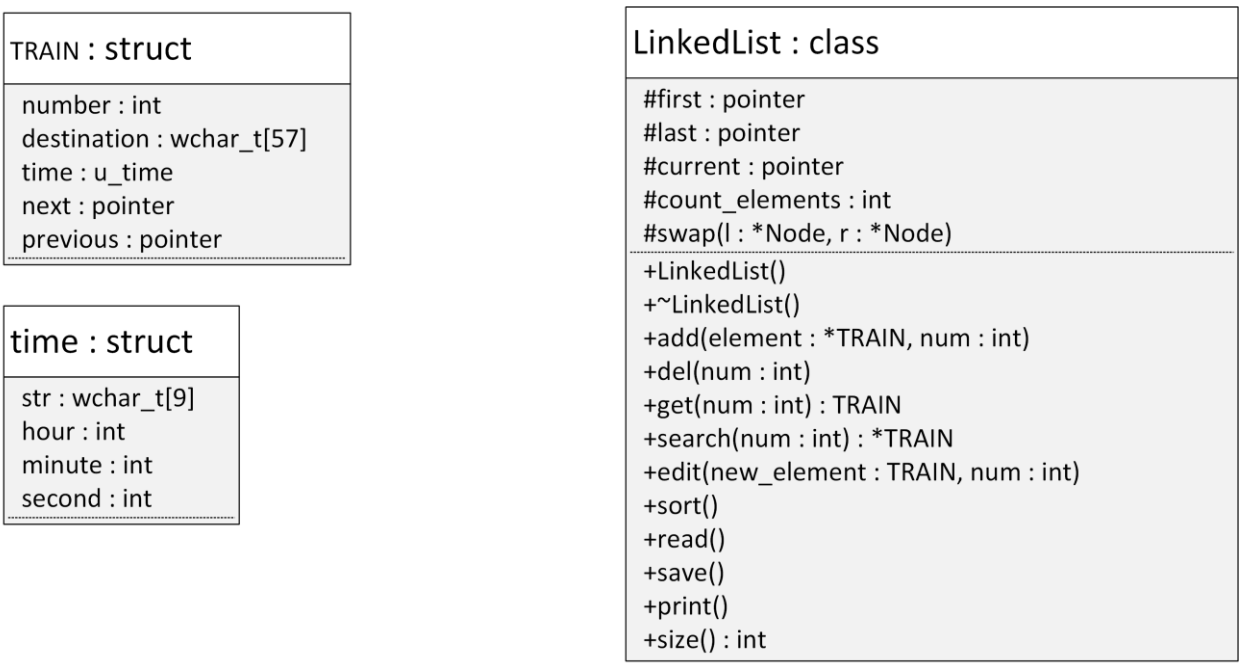


Рис. 1 — Диаграмма классов

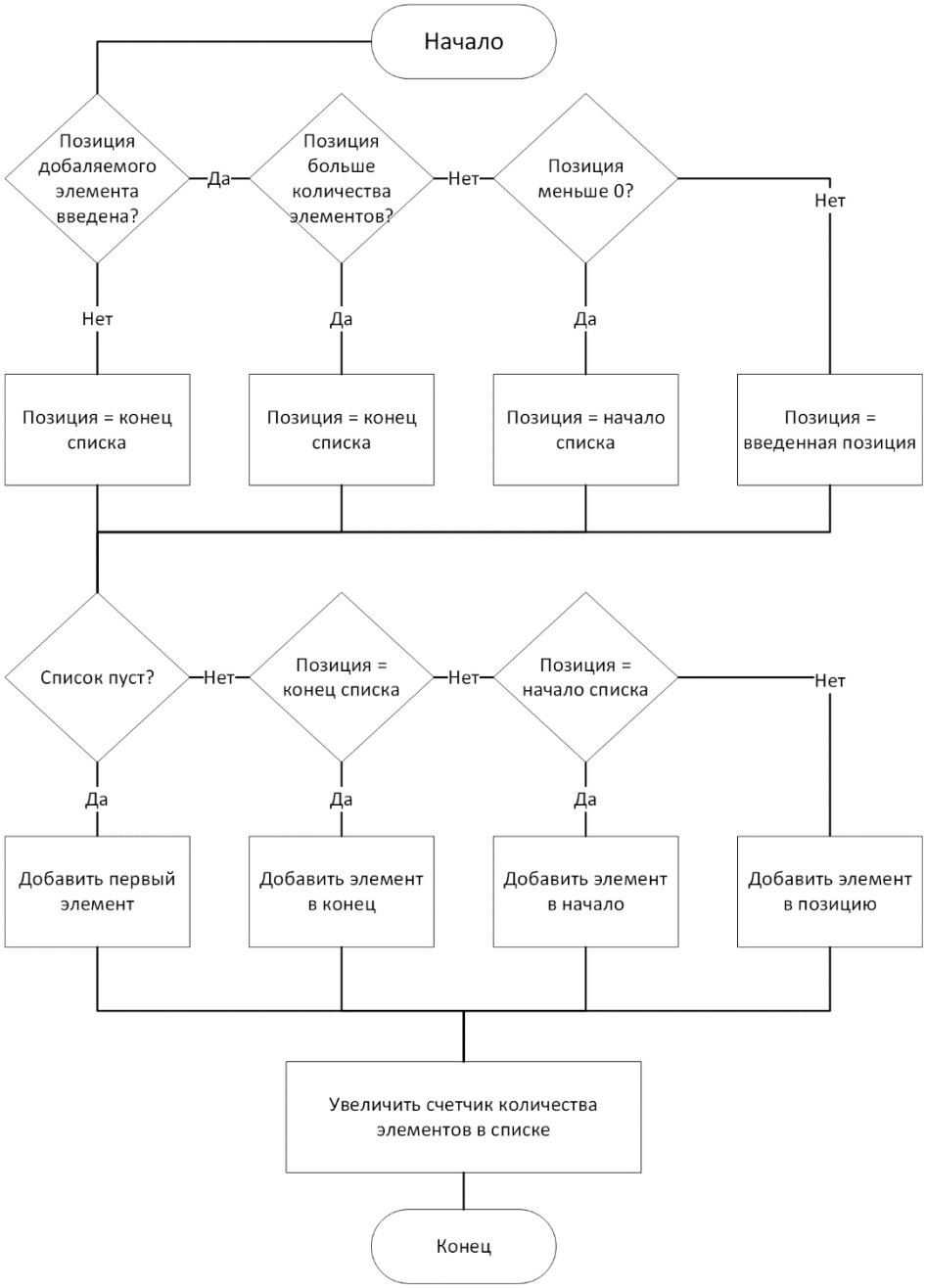


Рис. 2 — Блок-схема алгоритма Добавление

Рис. 3— Блок-схема алгоритма Удаление

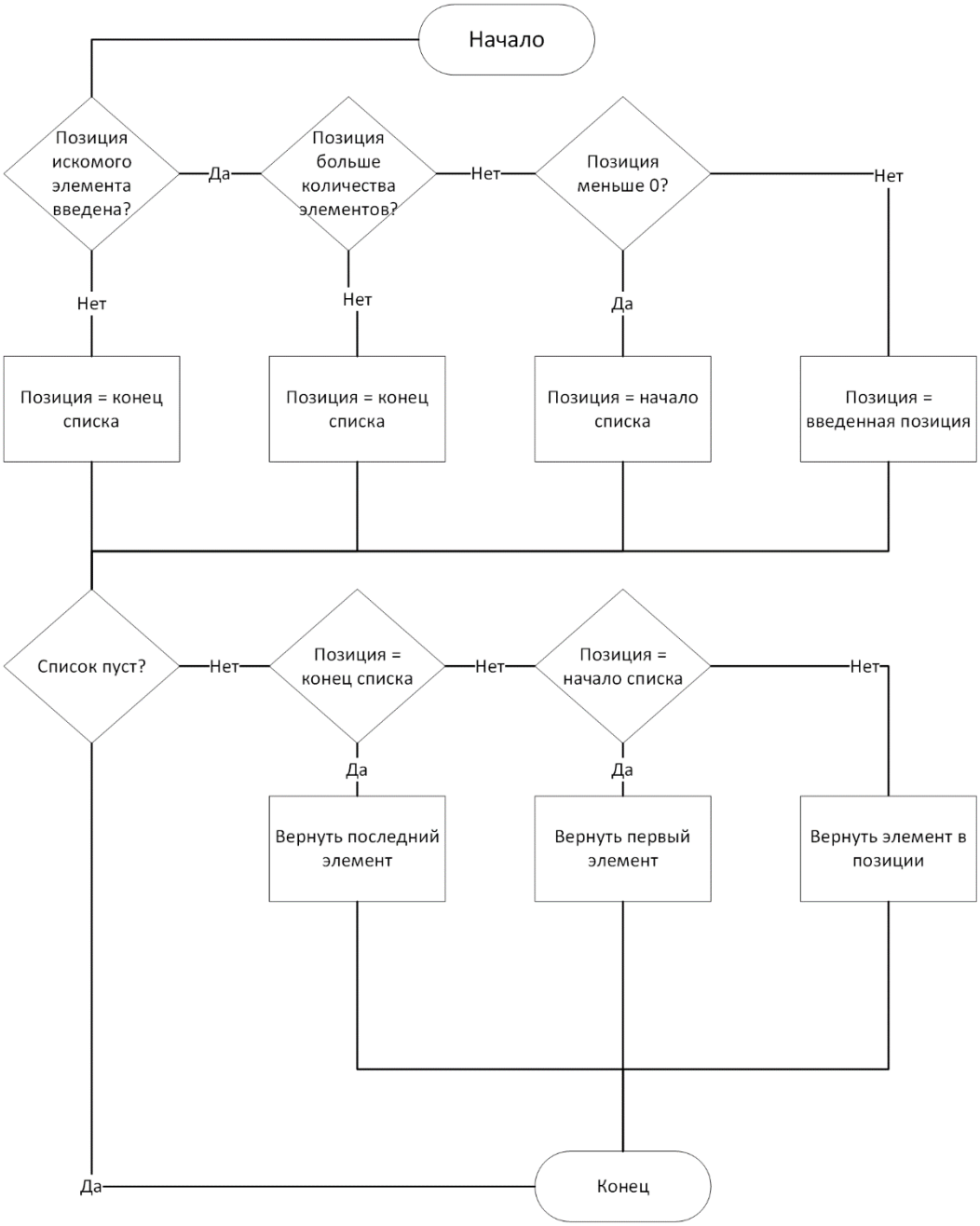


Рис 4 — Блок-схема алгоритма Получить

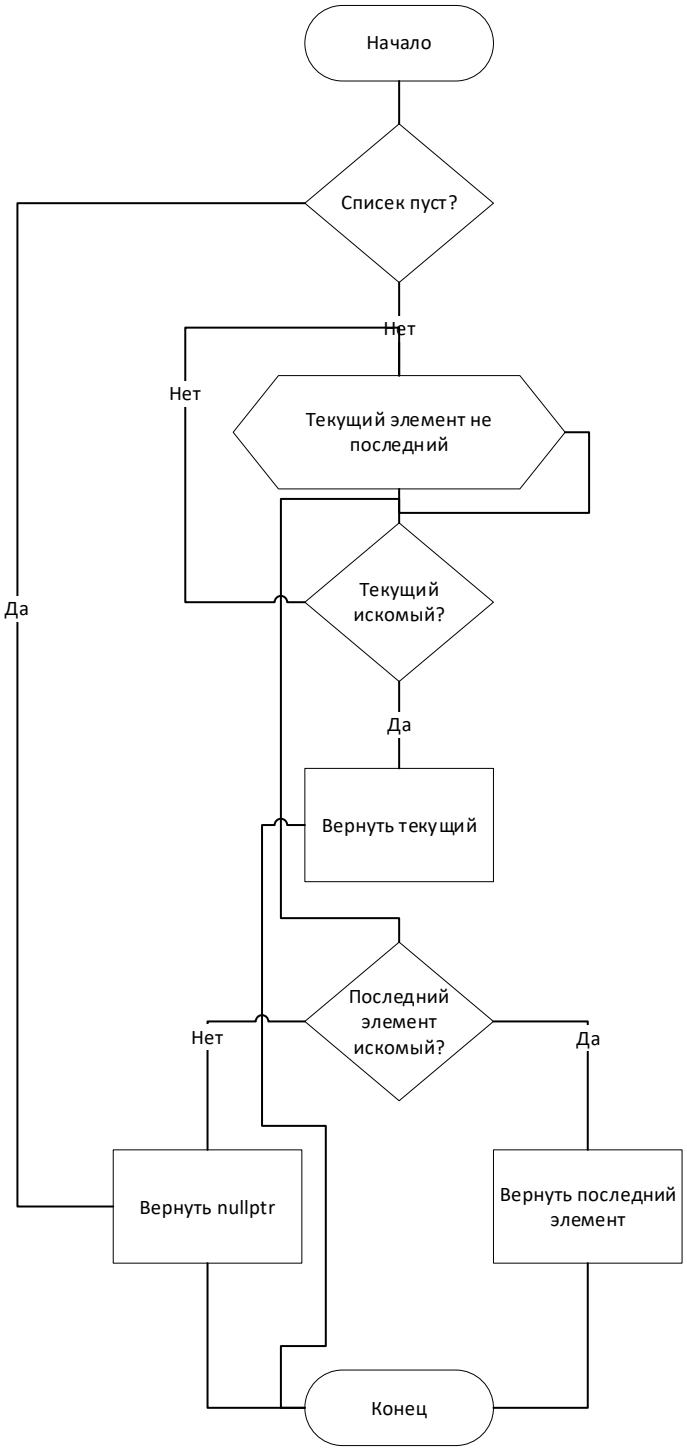


Рис 5 — Блок-схема алгоритма Поиск

Рис 6 — Блок-схема алгоритма Изменение

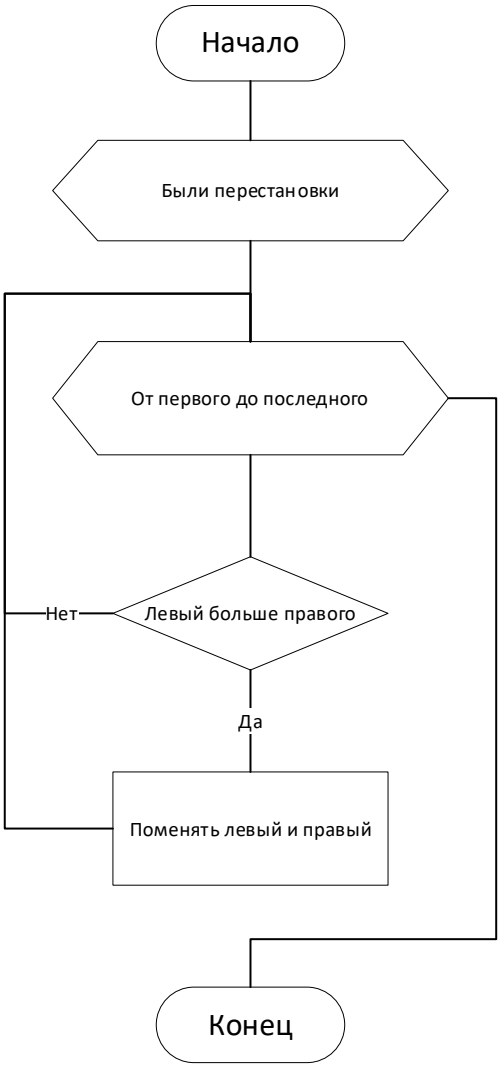


Рис 7 — Блок-схема алгоритма Сортировка

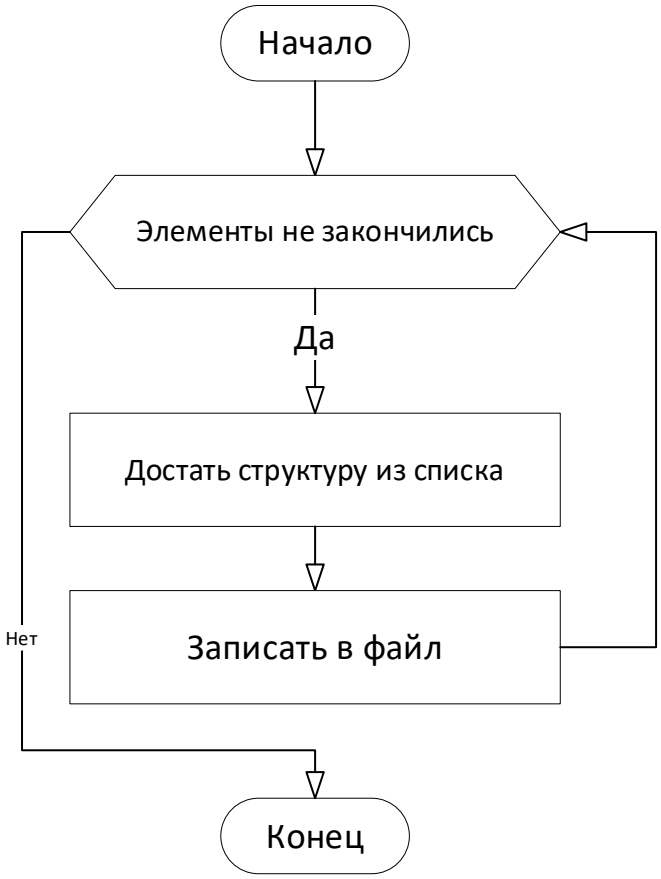


Рис 8 — Блок-схема алгоритма Сохранение

Рис 9 — Блок-схема алгоритма Чтение

Рис 10 — Блок-схема алгоритма Печать

id	number	destination	time
0	334	PKADqs	05:47:00
1	447	KJhpMb	02:35:00
2	333	NZerro	03:02:00
3	190	MmMePT	04:35:00

Рис 11 — Распечатка списка

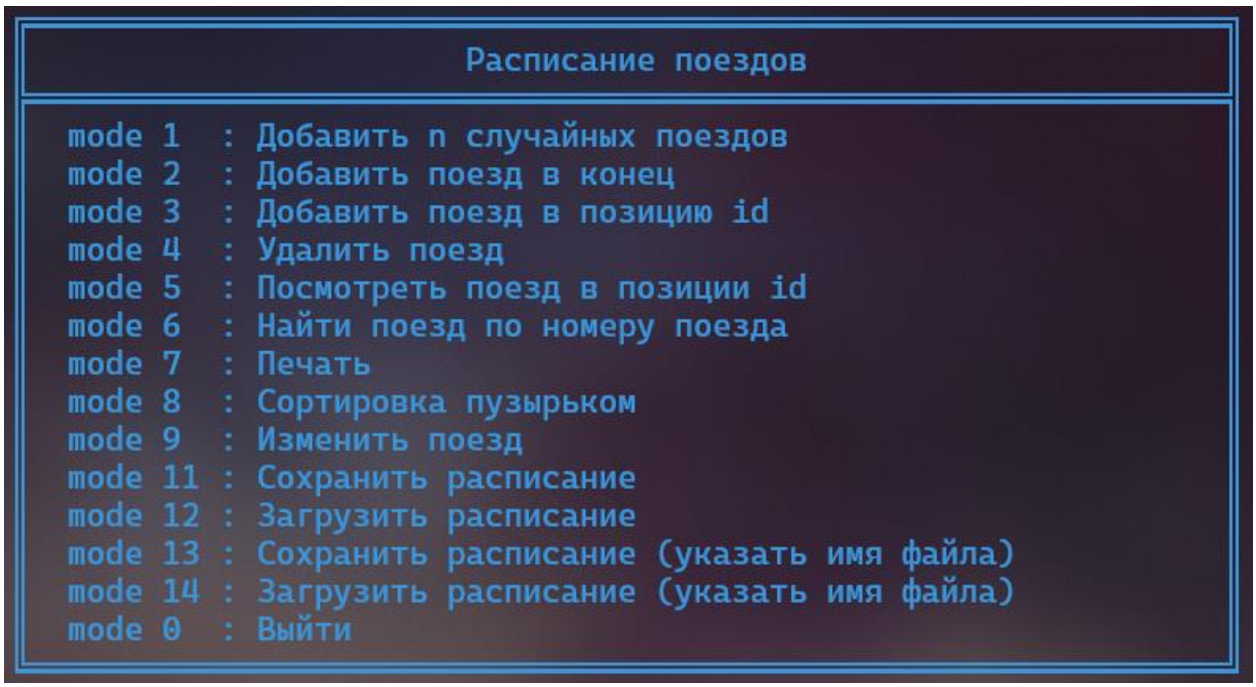


Рис 12 — Меню пользователя

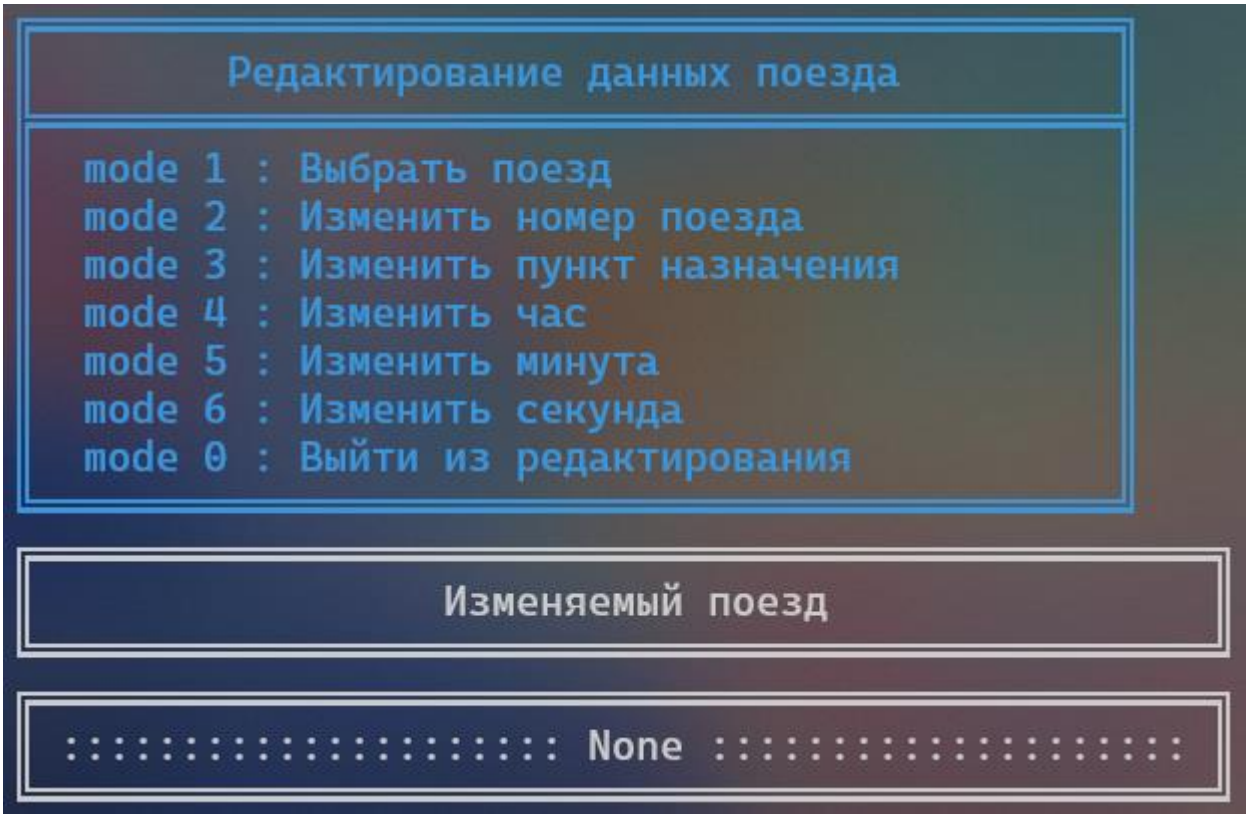


Рис 13 — Меню редактирования поезда

Приложение 2 Код программы

Код также можно найти на GitHub https://github.com/limenitis/SUAI/tree/master/BP-CP

linkedlist.h

```
#pragma once
#include <iostream>
#include <fstream>
```

```
namespace train_namespace
{
    using std::cerr;
    using std::cin;
    using std::clog;
    using std::cout;
    using std::wcerr;
    using std::wclog;
    using std::wcout;
    using std::wcin;
    using std::endl;
    using std::istream;
    using std::ostream;
    using std::ifstream;
    using std::ofstream;
    using std::wistream;
    using std::wostream;
    using std::wifstream;
    using std::wofstream;
}

struct u_time
{
    wchar_t str[9] = {'0','0',':','0','0',':','0','0','\0'};
    int hour = 0;
    int minute = 0;
    int second = 0;
};

/*
    Самое длинное название города в мире (57):
    Лланвайрпуллгуингиллг
    огерихуирндробуллллан
    тисилиогогогóх

    Самое длинное название города в России (25):
    Александровск-Сахалинский
*/

struct TRAIN
{
    int number;
    wchar_t destination[57]{'\0'};
    u_time time;
    TRAIN* next;
    TRAIN* prev;
};

void convert_time(u_time &t);

wchar_t  gen_rand_key(bool);
TRAIN*   gen_rand_train();

bool is_num_c (char);
bool is_num_i (int );
bool is_char  (int );

wchar_t  num2char(int    );
int      char2int(wchar_t);

bool is_russ(wchar_t c);
void correct_to_rus_symbol(wchar_t &symbol);
void correct_to_rus_symbol_str(wchar_t str[]);
void correct_from_rus_symbol(wchar_t &symbol);
void correct_from_rus_symbol_str(wchar_t str[]);
```

```
class LinkedList {
protected:
    void  swap(TRAIN *l, TRAIN *r);

    TRAIN* FIRST;
    TRAIN* CURRENT;
    TRAIN* LAST;
    int count_elements;

public:
    LinkedList();
    ~LinkedList();

    void add(TRAIN *element, int num = -1);
    void del(int num = -1);

    TRAIN  get(int num);
    TRAIN* search(int number);

    void edit(TRAIN new_element, int num);
    void bubble_sort();

    void read();
    void read(wchar_t str[]);
    void save();
    void save(wchar_t str[]);
    void print();

    int size();

    friend std::wostream&  operator<< (std::wostream&  out,  LinkedList& obj);
    friend std::wofstream& operator<< (std::wofstream& fout, LinkedList& obj);
    friend std::wifstream& operator>> (std::wifstream& fin,  LinkedList& obj);
};

bool operator< (u_time t1, u_time t2);
bool operator> (u_time t1, u_time t2);
bool operator<=(u_time t1, u_time t2);
bool operator>=(u_time t1, u_time t2);

std::wostream&  operator<< (std::wostream&  out,  TRAIN &n);

std::wostream&  operator<< (std::wostream&  out,  u_time &t);
std::wistream&  operator>> (std::wistream&  in,   u_time &t);
std::wofstream& operator<< (std::wofstream& fout, u_time& obj);
std::wifstream& operator>> (std::wifstream& fin,  u_time& obj);
```

linkedlist.cpp

```
#pragma once
#include ".\..\include\linked-list.h"
#include ".\..\include\color_out.h"
#include <iostream>
#include <iomanip>
#include <fstream>

using namespace train_namespace;

LinkedList::LinkedList()
{
    FIRST = nullptr;
    CURRENT = nullptr;
    LAST = nullptr;
    count_elements = 0;
```

```
}

LinkedList::~LinkedList()
{
    if (count_elements != 0)
    {
        while (count_elements != 0 && FIRST->next != nullptr)
        {
            CURRENT = FIRST;
            FIRST = FIRST->next;
            delete CURRENT;
            count_elements -= 1;
        }
        if (count_elements != 0)
        {
            delete FIRST;
        }
    }
}

void LinkedList::add(TRAIN *new_element, int num)
{
    if (num == -1)
    {
        num = count_elements;
    }
    else if (num > count_elements)
    {
        num = count_elements;
        wcolor_out(L"[Linked list] [add] : num over count elements \n", "red");
        wcolor_out(L"[Linked list] [add] : add last element in list \n", "red");
    }
    else if(num < -1)
    {
        num = 0;
        wcolor_out(L"[Linked list] [add] : num less 0 \n", "red");
        wcolor_out(L"[Linked list] [add] : add first element in list \n", "red");
    }

    if(FIRST == nullptr)
    {
        FIRST = new_element;
        LAST = new_element;
        new_element->next = nullptr;
        new_element->prev = nullptr;
    }
    else
    {
        if (num == 0)
        {
            new_element->prev = nullptr;
            FIRST->prev = new_element;
            new_element->next = FIRST;

            FIRST = new_element;
        }
        else if (num == count_elements)
        {
            new_element->next = nullptr;
            LAST->next = new_element;
            new_element->prev = LAST;

            LAST = new_element;
        }
        else
```

```
        {
            CURRENT = FIRST;
            while (CURRENT->next != nullptr && num > 0)
            {
                CURRENT = CURRENT->next;
                num -= 1;
            }

            //          paste before CURRENT
            CURRENT->prev->next = new_element;
            new_element->prev = CURRENT->prev;
            CURRENT->prev = new_element;
            new_element->next = CURRENT;
        }
    }

    count_elements += 1;
}

void LinkedList::del(int num)
{
    if (num == -1)
    {
        num = count_elements - 1;
    }
    else if (num > count_elements - 1 && count_elements != 0)
    {
        num = count_elements - 1;
        wcolor_out(L"[Linked list] [del] : num over count elements \n", "red");
        wcolor_out(L"[Linked list] [del] : delete last element in list \n", "red");
    }
    else if(num < -1)
    {
        num = 0;
        wcolor_out(L"[Linked list] [del] : num less 0 \n", "red");
        wcolor_out(L"[Linked list] [del] : command were ignored \n", "red");
        return;
    }

    if(FIRST == nullptr || count_elements == 0)
    {
        // list empty
        wcolor_out(L"[Linked list] [del] : list empty \n", "red");
        return;
    }
    else if(FIRST == LAST)
    {
        // only 1 element
        delete FIRST;
    }
    else
    {
        if (num == 0)
        {
            FIRST = FIRST->next;
            delete FIRST->prev;
            FIRST->prev = nullptr;
        }
        else if (num == count_elements - 1)
        {
            LAST = LAST->prev;
            delete LAST->next;
            LAST->next = nullptr;
        }
        else
```

```
        {
            CURRENT = FIRST;
            while (CURRENT->next != nullptr && num > 0)
            {
                CURRENT = CURRENT->next;
                num -= 1;
            }

            //          paste before CURRENT        // it's ok
            CURRENT->prev->next = CURRENT->next;    // becose first and last
            CURRENT->next->prev = CURRENT->prev;    // processing apart
            delete CURRENT;
        }
    }

    count_elements -= 1;
}

TRAIN LinkedList::get(int num)
{
    if (num == -1)
    {
        num = count_elements - 1;
    }
    else if (num > count_elements - 1 && count_elements != 0)
    {
        num = count_elements - 1;
        wcolor_out(L"[Linked list] [get] : num over count elements \n", "red");
        wcolor_out(L"[Linked list] [get] : return last element in list \n", "red");
    }
    else if(num < -1)
    {
        num = 0;
        wcolor_out(L"[Linked list] [get] : num less 0 \n", "red");
        wcolor_out(L"[Linked list] [get] : return first element in list \n", "red");
    }


    if(FIRST == nullptr || count_elements == 0)
    {
        // list empty
        wcolor_out(L"[Linked list] [get] : list empty \n", "red");
        TRAIN error_TRAIN;
        error_TRAIN.destination[0] = '\0';
        error_TRAIN.number = 0;
        error_TRAIN.time.str[0] =  {'\0'};
        return error_TRAIN;
    }
    else if(FIRST == LAST)
    {
        // only 1 element
        return *FIRST;
    }
    else
    {
        if (num == 0)
        {
            return *FIRST;
        }
        else if (num == count_elements - 1)
        {
            return *LAST;
        }
        else
        {
```

```
            CURRENT = FIRST;
            while (CURRENT->next != nullptr && num > 0)
            {
                CURRENT = CURRENT->next;
                num -= 1;
            }
            return *CURRENT;
        }
    }
}

TRAIN *LinkedList::search(int number)
{
    // find train for nuber
    CURRENT = FIRST;
    if(CURRENT == nullptr || count_elements == 0) { return nullptr; }

    while (CURRENT->next != nullptr)
    {
        if (CURRENT->number == number)
        {
            return CURRENT;
        }
        CURRENT = CURRENT->next;
    }

    if(CURRENT->number == number){ return CURRENT; }
    else { return nullptr; }
}

void LinkedList::edit(TRAIN new_element, int num)
{
    TRAIN *element;
    element = search(num);

    if (element == nullptr)
    {
        wcolor_out(L"[LinkedList::edit] : element not found \n", "red");
        return;
    }

    element->number = new_element.number;
    element->time = new_element.time;
    int i = 0;
    for(auto e : element->destination)
    {
        element->destination[i] = new_element.destination[i];
        i++;
    }
}

void LinkedList::swap(TRAIN *l, TRAIN *r)
{
    TRAIN *nex = r->next;
    TRAIN *last = l->prev;

    if (l == FIRST)
    {
        if (last != nullptr) { last->next = r; }
        l->next = nex;
        l->prev = r;
        r->next = l;
        r->prev = last;
        if (nex != nullptr) { nex->prev = l; }
        FIRST = r;
```

```
    }
    else if(r == LAST)
    {
        if (last != nullptr) { last->next = r; }
        l->next = nex;
        l->prev = r;
        r->next = l;
        r->prev = last;
        if (nex != nullptr) { nex->prev = l; }
        LAST = l;
    }
    else
    {
        if (last != nullptr) { last->next = r; }
        l->next = nex;
        l->prev = r;
        r->next = l;
        r->prev = last;
        if (nex != nullptr) { nex->prev = l; }
    }
}

void LinkedList::bubble_sort()
{
    bool swaps = 1;
    while (swaps){
        swaps = false;
        for (TRAIN *ptr = FIRST; ptr != nullptr && ptr->next != nullptr; ptr = ptr->next)
{
            if ( ptr->number > ptr->next->number )
            {
                swap(ptr, ptr->next);
                swaps = true;
            }
        }
    }
}

void LinkedList::read()
{
    wifstream fin("in.txt");   // входной поток
    fin >> *this;
    fin.close();
}

void LinkedList::read(wchar_t str[])
{
    wifstream fin(str);   // входной поток
    fin >> *this;
    fin.close();
}

void LinkedList::save()
{
    wofstream fout("out.txt");  // выходной поток
    fout << *this;
    fout.close();
}

void LinkedList::save(wchar_t str[])
{
    wofstream fout(str);  // выходной поток
    fout << *this;
    fout.close();
}
```

```
void LinkedList::print()
{
    CURRENT = FIRST;
    int num = count_elements;
    wcout << L"\n";
    wcout << L"╔══════╦══════════╦═══════════════════════════╦══════════╗ \n";
    wcout << L"║  id  ║  number  ║        destination        ║   time   ║ \n";
    while (CURRENT != nullptr && num != 0)
    {
        wcout << L"╠══════╬══════════╬═══════════════════════════╬══════════╣ \n";
        wcout.setf(std::ios::left); wcout << L"┃ "  /* ┃ ┆ ┃ ┆ */ << std::setw(5)  <<
count_elements - num;
        wcout.setf(std::ios::left); wcout << L"┃ "  /* ┃ ┆ ┃ ┆ */ << std::setw(9)  <<
CURRENT->number;
        correct_to_rus_symbol_str(CURRENT->destination);
        wcout.setf(std::ios::left); wcout << L"┃ "  /* ┃ ┆ ┃ ┆ */ << std::setw(26) <<
CURRENT->destination;
        correct_from_rus_symbol_str(CURRENT->destination);
        wcout.setf(std::ios::left); wcout << L"┃ "  /* ┃ ┆ ┃ ┆ */ << std::setw(9)  <<
CURRENT->time;
                                    wcout << L"┃\n" /* ┃ ┆ ┃ ┆ */   ;
        CURRENT = CURRENT->next;
        num--;
    }
    wcout << L"╚══════╩══════════╩═══════════════════════════╩══════════╝ \n";
    wcout << L"\n";
}

int LinkedList::size() { return count_elements; }

std::wostream  &operator<<(std::wostream &out,   u_time &t)
{
    convert_time(t);
    out << t.str;

    return out;
}

std::wistream  &operator>>(std::wistream &in,    u_time &t)
{
    wcout << L"hour   : "; in >> t.hour;
    wcout << L"minute : "; in >> t.minute;
    wcout << L"second : "; in >> t.second;
    convert_time(t);
    return in;
}

std::wofstream &operator<<(std::wofstream &fout, u_time &t)
{
    fout << t.str;

    return fout;
}

std::wifstream &operator>>(std::wifstream &fin,  u_time &t)
{
    wchar_t symbol = ' ';
    wchar_t tmp[2] = {' ', ' '};

    fin >> tmp[0];
    fin >> tmp[1];
    t.hour   = char2int(tmp[0])*10 + char2int(tmp[1]);
    fin >> symbol; // get ':'
```

```
    fin >> tmp[0];
    fin >> tmp[1];
    t.minute = char2int(tmp[0])*10 + char2int(tmp[1]);
    fin >> symbol; // get ':'
    fin >> tmp[0];
    fin >> tmp[1];
    t.second = char2int(tmp[0])*10 + char2int(tmp[1]);
    convert_time(t);
    return fin;
}

std::wostream  &operator<<(std::wostream &out,   LinkedList &obj)
{
    out << L"\n";
    obj.print();
    out << L"\n";

    return out;
}

std::wofstream &operator<<(std::wofstream &fout, LinkedList &obj)
{
    obj.CURRENT = obj.FIRST;
    int num = obj.count_elements;
    while (obj.CURRENT != nullptr && num > 0)
    {
        fout << obj.CURRENT->number << endl;
        fout << obj.CURRENT->destination << endl;
        fout << obj.CURRENT->time << endl;
        obj.CURRENT = obj.CURRENT->next;
        num--;
    }
    fout << L"\n";

    return fout;
}

std::wifstream &operator>>(std::wifstream &fin,  LinkedList &obj)
{
    while (!fin.eof())
    {
        TRAIN* new_element = new TRAIN;
        fin >> new_element->number;
        fin >> new_element->destination;
        fin >> new_element->time;
        obj.add(new_element);
    }
    obj.del();

    return fin;
}

std::wostream  &operator<<(std::wostream  &out,  TRAIN &n)
{
    out << L"╔═══════════╦═══════════════════════════╦═══════════╗ \n";
    out << L"║  number   ║        destination        ║   time    ║ \n";
    out << L"╠═══════════╬═══════════════════════════╬═══════════╣ \n";
    out.setf(std::ios::left); out << L"║ " << std::setw(9)  << n.number;
    correct_to_rus_symbol_str(n.destination);
    out.setf(std::ios::left); out << L"║ " << std::setw(26) << n.destination;
    correct_from_rus_symbol_str(n.destination);
    out.setf(std::ios::left); out << L"║ " << std::setw(9)  << n.time;
    out << L"║\n";
    out << L"╚═══════════╩═══════════════════════════╩═══════════╝ \n";
```

```
    return out;
}

bool operator< (u_time t1, u_time t2)
{
    if (t1.hour < t2.hour)
    {
        return true;
    }
    else if (t1.hour == t2.hour)
    {
        if (t1.minute < t2.minute)
        {
            return true;
        }
        else if (t1.minute == t2.minute)
        {
            if (t1.second < t2.second)
            {
                return true;
            }
            else if (t1.second == t2.second)
            {
                return false;
            }
            else
            {
                return false;
            }
        }
        else
        {
            return false;
        }
    }
    else
    {
        return false;
    }
}
bool operator> (u_time t1, u_time t2)
{
    if (t1.hour > t2.hour)
    {
        return true;
    }
    else if (t1.hour == t2.hour)
    {
        if (t1.minute > t2.minute)
        {
            return true;
        }
        else if (t1.minute == t2.minute)
        {
            if (t1.second > t2.second)
            {
                return true;
            }
            else if (t1.second == t2.second)
            {
                return false;
            }
            else
            {
                return false;
```

```
            }
        }
        else
        {
            return false;
        }
    }
    else
    {
        return false;
    }
}
bool operator<=(u_time t1, u_time t2)
{
    if(t1.hour == t2.hour && t1.minute == t2.minute && t1.second == t2.second)
    {
        return true;
    }

    if (t1.hour <= t2.hour)
    {
        return true;
    }
    else if (t1.hour == t2.hour)
    {
        if (t1.minute <= t2.minute)
        {
            return true;
        }
        else if (t1.minute == t2.minute)
        {
            if (t1.second <= t2.second)
            {
                return true;
            }
            else if (t1.second == t2.second)
            {
                return false;
            }
            else
            {
                return false;
            }
        }
        else
        {
            return false;
        }
    }
    else
    {
        return false;
    }
}
bool operator>=(u_time t1, u_time t2)
{
    if (t1.hour == t2.hour && t1.minute == t2.minute && t1.second == t2.second)
    {
        return true;
    }

    if (t1.hour >= t2.hour)
    {
        return true;
    }
```

```
    else if (t1.hour == t2.hour)
    {
        if (t1.minute >= t2.minute)
        {
            return true;
        }
        else if (t1.minute == t2.minute)
        {
            if (t1.second >= t2.second)
            {
                return true;
            }
            else if (t1.second == t2.second)
            {
                return false;
            }
            else
            {
                return false;
            }
        }
        else
        {
            return false;
        }
    }
    else
    {
        return false;
    }
}

TRAIN* gen_rand_train()
{
    TRAIN* train = new TRAIN;
    train->time.hour   = 0 + rand() % 12;
    train->time.minute = 0 + rand() % 60;
    train->time.second = 0;
    train->number      = 0 + rand() % 1000;

    int i = 0;
    bool format[] = { 1, 1, 1, 1, 1, 1 };  // char -> 1             int -> 0
    for (bool f : format)
    {
        train->destination[i] = gen_rand_key(f);
        i += 1;
    }

    train->next = nullptr;
    train->prev = nullptr;

    return train;
}

wchar_t gen_rand_key(bool f)
{
    wchar_t key;
    int MIN_VALUE_C = 65;
    int MAX_VALUE_C = 122;
    int MIN_VALUE_I = 0;
    int MAX_VALUE_I = 9;

    int tmp;
    if (f) // char
    {
```

```
        tmp = MIN_VALUE_C + rand() % MAX_VALUE_C;
        while (!is_char(tmp))
        {
            tmp = MIN_VALUE_C + rand() % MAX_VALUE_C;
        }
        key = tmp;
    }
    else // int
    {
        tmp = MIN_VALUE_I + rand() % MAX_VALUE_I;
        while (!is_num_i(tmp))
        {
            tmp = MIN_VALUE_I + rand() % MAX_VALUE_I;
        }
        key = num2char(tmp);
    }
    return key;
}

bool is_num_c(char num)
{
    char nums[] = {'1', '2', '3', '4', '5', '6', '7', '8', '9', '0'};
    for (auto i : nums)
    {
        if (num == i)
        {
            return true;
        }
    }
    return false;
}

bool is_num_i(int num)
{
    int nums[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
    for (auto i : nums)
    {
        if (num == i)
        {
            return true;
        }
    }
    return false;
}

bool is_char(int c)
{
    if ((64 < c) && (c < 91) || (96 < c) && (c < 123))
    {
        return true;
    }
    else
    {
        return false;
    }
}

wchar_t num2char(int i) // 48 char == 0 int
{
    return i + 48;
}

int char2int(wchar_t c) // 48 char == 0 int
{
    return c - 48;
```

```
}

void convert_time(u_time &t)
{
    // ints to str
    wchar_t temp[2] = {'0','0'};
    if(t.hour / 10 == 0) // one-digit number
    {
        temp[0] = '0';
        temp[1] = num2char(t.hour);
    }
    else // two-digit number
    {
        temp[0] = num2char(t.hour / 10);
        temp[1] = num2char(t.hour % 10);
    }

    t.str[0] = temp[0];
    t.str[1] = temp[1];

    if(t.minute / 10 == 0)
    {
        temp[0] = '0';
        temp[1] = num2char(t.minute);
    }
    else
    {
        temp[0] = num2char(t.minute / 10);
        temp[1] = num2char(t.minute % 10);
    }

    t.str[3] = temp[0];
    t.str[4] = temp[1];

    if(t.second / 10 == 0)
    {
        temp[0] = '0';
        temp[1] = num2char(t.second);
    }
    else
    {
        temp[0] = num2char(t.second / 10);
        temp[1] = num2char(t.second % 10);
    }

    t.str[6] = temp[0];
    t.str[7] = temp[1];
}


bool is_russ(wchar_t c)
{
       return c >= 192 && c<= 255;
}

// А == 1040 // print
// А == 192 // input
// Я == 223 // input
// а == 224 // input
// я == 255 // input
void correct_to_rus_symbol(wchar_t &symbol)
{
    if (is_russ(symbol))
    {
        symbol = wchar_t( int(symbol) + 848);
```

```
    }
}

void correct_to_rus_symbol_str(wchar_t str[])
{
    int i = 0;
    while (str[i] != '\0')
    {
        correct_to_rus_symbol(str[i]);
        i++;
    }
}

void correct_from_rus_symbol(wchar_t &symbol)
{
    if ( is_russ( wchar_t( int(symbol) - 848) ) )
    {
        symbol = wchar_t( int(symbol) - 848);
    }
}

void correct_from_rus_symbol_str(wchar_t str[])
{
    int i = 0;
    while(str[i] != '\0')
    {
        correct_from_rus_symbol(str[i]);
        i++;
    }
}
```

color_out.h

```
#pragma once
#include <iostream>
#include <string>
#include <cassert>
#include <fcntl.h>
#include <io.h>
#include ".\..\include\linked-list.h"

namespace colors
{
    using std::cout;
    using std::wcout;
    using std::wcerr;
    using std::cerr;
    using std::endl;
    using std::string;
    using std::to_string;
    using std::wstring;
    using std::to_wstring;
}
using namespace colors;

int front2int(string front_color);

int back2int(string back_color);

// -------------
// front colors:
// black, red, green, yellow, blue, magenda, cyan, white
// -------------
// back colors:
// on_black, on_red, on_green, on_yellow, on_blue, on_magenda, on_cyan, on_white
```

```cpp
// -------------
template <typename T>
inline void color_out(const T &str, string front_color, string back_color = "default")
{
    int front = front2int(front_color);
    int back = back2int(back_color);

    if(front != -1) { cout  << "\x1b[3" << front << "m"; }
    if(back  != -1) { cout  << "\x1b[4" << back  << "m"; }
    cout << str << "\x1b[0m";
}

// -------------
// front colors:
// black, red, green, yellow, blue, magenda, cyan, white
// -------------
// back colors:
// on_black, on_red, on_green, on_yellow, on_blue, on_magenda, on_cyan, on_white
// -------------
template <typename T>
inline void wcolor_out(const T &str, string front_color, string back_color = "default")
{
    int front = front2int(front_color);
    int back = back2int(back_color);

    if(front != -1) wcout << L"\x1b[3" << front << L"m";
    if(back  != -1) wcout << L"\x1b[4" << back  << L"m";
    wcout << str << L"\x1b[0m";
}

// const string BLACK   = "\x1b[30m";
// const string RED     = "\x1b[31m";
// const string GREEN   = "\x1b[32m";
// const string YELLOW  = "\x1b[33m";
// const string BLUE    = "\x1b[34m";
// const string MAGENDA = "\x1b[35m";
// const string CYAN    = "\x1b[36m";
// const string WHITE   = "\x1b[37m";
// const string ON_BLACK   = "\x1b[40m";  // background
// const string ON_RED     = "\x1b[41m";  // background
// const string ON_GREEN   = "\x1b[42m";  // background
// const string ON_YELLOW  = "\x1b[43m";  // background
// const string ON_BLUE    = "\x1b[44m";  // background
// const string ON_MAGENDA = "\x1b[45m";  // background
// const string ON_CYAN    = "\x1b[46m";  // background
// const string ON_WHITE   = "\x1b[47m";  // background
// const string RESET_COLOR = "\x1b[0m";

// const wstring wBLACK   = L"\x1b[30m";
// const wstring wRED     = L"\x1b[31m";
// const wstring wGREEN   = L"\x1b[32m";
// const wstring wYELLOW  = L"\x1b[33m";
// const wstring wBLUE    = L"\x1b[34m";
// const wstring wMAGENDA = L"\x1b[35m";
// const wstring wCYAN    = L"\x1b[36m";
// const wstring wWHITE   = L"\x1b[37m";
// const wstring wON_BLACK   = L"\x1b[40m";  // background
// const wstring wON_RED     = L"\x1b[41m";  // background
// const wstring wON_GREEN   = L"\x1b[42m";  // background
// const wstring wON_YELLOW  = L"\x1b[43m";  // background
// const wstring wON_BLUE    = L"\x1b[44m";  // background
// const wstring wON_MAGENDA = L"\x1b[45m";  // background
// const wstring wON_CYAN    = L"\x1b[46m";  // background
// const wstring wON_WHITE   = L"\x1b[47m";  // background
// const wstring wRESET_COLOR = L"\x1b[0m";
```

```
// static const char* chBLACK   = "\x1b[30m";
// static const char* chRED     = "\x1b[31m";
// static const char* chGREEN   = "\x1b[32m";
// static const char* chYELLOW  = "\x1b[33m";
// static const char* chBLUE    = "\x1b[34m";
// static const char* chMAGENDA = "\x1b[35m";
// static const char* chCYAN    = "\x1b[36m";
// static const char* chWHITE   = "\x1b[37m";
// static const char* chON_BLACK   = "\x1b[40m";  // background
// static const char* chON_RED     = "\x1b[41m";  // background
// static const char* chON_GREEN   = "\x1b[42m";  // background
// static const char* chON_YELLOW  = "\x1b[43m";  // background
// static const char* chON_BLUE    = "\x1b[44m";  // background
// static const char* chON_MAGENDA = "\x1b[45m";  // background
// static const char* chON_CYAN    = "\x1b[46m";  // background
// static const char* chON_WHITE   = "\x1b[47m";  // background
// static const char* chRESET_COLOR = "\x1b[0m";
```

color_out.cpp

```
#pragma once
#include ".\..\include\color_out.h"
#include <iostream>
#include <string>
#include <fcntl.h>
#include <io.h>

// using namespace colors;
using namespace std;

int front2int (string front_color)
{
    if (front_color[0] == 'b')
    {
        if (front_color[2] == 'a') // black
        {
            return 0;
        }
        else if (front_color[2] == 'u') // blue
        {
            return 4;
        }
        else
        {
            return -1;
        }
    }
    else if (front_color[0] == 'r') // red
    {
        return 1;
    }
    else if (front_color[0] == 'g') // green
    {
        return 2;
    }
    else if (front_color[0] == 'y') // yellow
    {
        return 3;
    }
    else if (front_color[0] == 'm') // magenda
    {
        return 5;
    }
```

```
    else if (front_color[0] == 'c') // cyan
    {
        return 6;
    }
    else if (front_color[0] == 'w') // white
    {
        return 7;
    }
    else
    {
        return -1;
    }
}

int back2int (string back_color)
{
    if (back_color[3] == 'b')
    {
        if (back_color[5] == 'a') // on_black
        {
            return 0;
        }
        else if (back_color[5] == 'u') // on_blue
        {
            return 4;
        }
        else
        {
            return -1;
        }
    }
    else if (back_color[3] == 'r') // on_red
    {
        return 1;
    }
    else if (back_color[3] == 'g') // on_green
    {
        return 2;
    }
    else if (back_color[3] == 'y') // on_yellow
    {
        return 3;
    }
    else if (back_color[3] == 'm') // on_magenda
    {
        return 5;
    }
    else if (back_color[3] == 'c') // on_cyan
    {
        return 6;
    }
    else if (back_color[3] == 'w') // on_white
    {
        return 7;
    }
    else
    {
        return -1;
    }
}
```

main.cpp

```
#pragma once
#include ".\..\include\color_out.h"
```

```
#include ".\..\include\linked-list.h"
#include <string>
#include <iostream>
#include <fcntl.h>
#include <io.h>
#include <vector>
#include <Windows.h>
#include <iomanip>

// using namespace train_namespace;
using namespace std;

// Для обнаружения утечек памяти
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>
#ifdef _DEBUG
#ifndef DBG_NEW
#define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
#define newDBG_NEW
#endif
#endif

#define DumpMemoryLeaks                                 \
_CrtSetReportMode(_CRT_WARN, _CRTDBG_MODE_FILE);        \
_CrtSetReportFile(_CRT_WARN, _CRTDBG_FILE_STDOUT);      \
_CrtSetReportMode(_CRT_ERROR, _CRTDBG_MODE_FILE);       \
_CrtSetReportFile(_CRT_ERROR, _CRTDBG_FILE_STDOUT);     \
_CrtSetReportMode(_CRT_ASSERT, _CRTDBG_MODE_FILE);      \
_CrtSetReportFile(_CRT_ASSERT, _CRTDBG_FILE_STDOUT);    \
_CrtDumpMemoryLeaks();

vector <int> v; // история команд

/**
 * @brief               Функция проверяет не упал ли поток ввода
 *                      при попытке ввести не тот тип данных и проверяет
 *                      дополнительные условия checks
 * @tparam T            Тип запрашиваемых данных
 * @param message       Сообщение, какие данные нужны на ввод
 * @param var           Переменная куда записать входные данные
 * @param checks        Дополнительные проверки - функция принимающая var и выводящая со-
общение
 *                      при неудачной проверке
 */
template <typename T>
void winput(const wchar_t* message, T &var, bool checks(T) = nullptr)
{
    while(true)
    {
        try
        {
            wcout << message;
            wcin >> var;
            wcout << endl;
            v.push_back(int(var)); // история команд
            if (wcin.fail()) { throw 1; }

            // Проверка дополнительных условий
            if(checks)
            {
                if (checks(var) == false)
                {
                    throw 2;
                }
```

```
            }
            break;
        }
        catch (int error_id)
        {
            switch (error_id)
            {
            case 1:
                wcin.clear();
                wcin.ignore();
                wcolor_out(L"---------------------------------------- \n", "red");
                wcolor_out(L"Неправильный тип данных \n", "red");
                wcolor_out(L"---------------------------------------- \n", "red");
                break;
            case 2:
                wcin.clear();
                wcin.ignore();
                // function checks() output message of error
                break;
            default:
                assert(0 && L"Undefined error_id");
            }
        }
        catch (...)
        {
            wcerr << L"Unknown error" << endl;
        }
    }
}

bool check_hour      ( int hour   );
bool check_minute    ( int minute );
bool check_second    ( int second );
bool check_positive  ( int var    );
void test_rus_symb(); // для проверки конвертации символов

int main()
{
    LinkedList* list = new LinkedList;
    TRAIN* element;
    TRAIN  element2;
    int input_int;
    wchar_t input_str[10];

    SetConsoleCP(1251);         // установка кодовой страницы win-cp 1251 в поток ввода
    SetConsoleOutputCP(1251);   // установка кодовой страницы win-cp 1251 в поток вывода
    input_int = _setmode(_fileno(stdout), _O_U16TEXT);

    while (true)
    {
        // test_rus_symb();
        // return 0;

        wcolor_out( L"╔═══════════════════════════════════════════════════════════════╗
\n", "cyan");
        wcolor_out( L"║                      Расписание поездов                       ║
\n", "cyan");
        wcolor_out( L"╠═══════════════════════════════════════════════════════════════╣
\n", "cyan");
        wcolor_out( L"║  mode 1  : Добавить n случайных поездов                       ║
\n", "cyan");
        wcolor_out( L"║  mode 2  : Добавить поезд в конец                             ║
\n", "cyan");
        wcolor_out( L"║  mode 3  : Добавить поезд в позицию id                        ║
\n", "cyan");
```

```
        wcolor_out( L"║  mode 4  : Удалить поезд                                    ║
\n", "cyan");
        wcolor_out( L"║  mode 5  : Посмотреть поезд в позиции id                    ║
\n", "cyan");
        wcolor_out( L"║  mode 6  : Найти поезд по номеру поезда                     ║
\n", "cyan");
        wcolor_out( L"║  mode 7  : Печать                                           ║
\n", "cyan");
        wcolor_out( L"║  mode 8  : Сортировка по номеру поезда                      ║
\n", "cyan");
        wcolor_out( L"║  mode 9  : Изменить поезд                                   ║
\n", "cyan");
        wcolor_out( L"║  mode 11 : Сохранить расписание                             ║
\n", "cyan");
        wcolor_out( L"║  mode 12 : Загрузить расписание                             ║
\n", "cyan");
        wcolor_out( L"║  mode 13 : Сохранить расписание (указать имя файла)         ║
\n", "cyan");
        wcolor_out( L"║  mode 14 : Загрузить расписание (указать имя файла)         ║
\n", "cyan");
        wcolor_out( L"║  mode 0  : Выйти                                            ║
\n", "cyan");
        wcolor_out( L"╚═════════════════════════════════════════════════════════════╝
\n", "cyan");

        winput(L"mode : ", input_int);

        switch (input_int)
        {
        case 1:
            winput(L"Количество поездов : ", input_int, check_positive);
            while (input_int--)
            {
                list->add(gen_rand_train());
            }
            break;
        case 2:
            element = new TRAIN;
            winput(L"Номер поезда : ", element->number, check_positive);
            winput(L"Пункт назначения : ", element->destination);
            winput(L"час : ", element->time.hour, check_hour);
            winput(L"минута : ", element->time.minute, check_minute);
            winput(L"секунда : ", element->time.second, check_second);

            list->add(element);
            break;
        case 3:
            element = new TRAIN;
            winput(L"Номер поезда : ", element->number, check_positive);
            winput(L"Пункт назначения : ", element->destination);
            winput(L"час : ", element->time.hour, check_hour);
            winput(L"минута : ", element->time.minute, check_minute);
            winput(L"секунда : ", element->time.second, check_second);

            winput(L"id : ", input_int, check_positive);
            list->add(element, input_int);
            break;
        case 4:
            winput(L"id : ", input_int, check_positive);
            list->del(input_int);
            break;
        case 5:
            winput(L"id : ", input_int, check_positive);
            element2 = list->get(input_int);
            wcout << endl << element2 << endl;
```

```
            break;
        case 6:
            winput(L"Номер поезда : ", input_int, check_positive);
            element = list->search(input_int);
            if(element != nullptr)
            {
                wcout << *element;
            }
            else
            {
                wcolor_out(L"Не найден \n", "red");
            }
            break;
        case 7:
            list->print();
            break;
        case 8:
            list->bubble_sort();
            break;
        case 9:
            element = nullptr;
            while (true)
            {
                wcolor_out( L"╔═══════════════════════════════════════════╗  \n",
"cyan");
                wcolor_out( L"║        Редактирование данных поезда       ║  \n",
"cyan");
                wcolor_out( L"╠═══════════════════════════════════════════╣  \n",
"cyan");
                wcolor_out( L"║  mode 1 : Выбрать поезд                   ║  \n",
"cyan");
                wcolor_out( L"║  mode 2 : Изменить номер поезда           ║  \n",
"cyan");
                wcolor_out( L"║  mode 3 : Изменить пункт назначения       ║  \n",
"cyan");
                wcolor_out( L"║  mode 4 : Изменить час                    ║  \n",
"cyan");
                wcolor_out( L"║  mode 5 : Изменить минуту                 ║  \n",
"cyan");
                wcolor_out( L"║  mode 6 : Изменить секунду                ║  \n",
"cyan");
                wcolor_out( L"║  mode 0 : Выйти из редактирования         ║  \n",
"cyan");
                wcolor_out( L"╚═══════════════════════════════════════════╝  \n",
"cyan");

                wcout << L"╔═══════════════════════════════════════════╗  \n";
                wcout << L"║              Изменяемый поезд             ║  \n";
                wcout << L"╚═══════════════════════════════════════════╝  \n";
                if (element) { wcout << *element << endl; }
                else
                {
                    wcout << L"╔═══════════════════════════════════════════╗  \n";
                    wcout << L"║ :::::::::::::::::::: None ::::::::::::::::::: ║  \n";
                    wcout << L"╚═══════════════════════════════════════════╝  \n";
                }

                winput(L"mode : ", input_int);
                if(input_int == 0) { break; }

                switch (input_int)
                {
                case 1:
                    winput(L" mode 1 : Изменение по номеру поезда \n mode 2 : Изменение
по id поезда \n mode : ", input_int, check_positive);
```

```
                    if(input_int == 1)
                    {
                        winput(L"Номер поезда, который изменяем : ", input_int,
check_positive);
                        element = list->search(input_int);
                    }
                    else if(input_int == 2)
                    {
                        winput(L"id поезда, который изменяем : ", input_int, check_posi-
tive);
                        element2 = list->get(input_int);
                        element = list->search(element2.number);
                    }
                    break;
                case 2:
                    winput(L"Новый номер поезда : ", element->number, check_positive);
                    break;
                case 3:
                    winput(L"Новый пункт назначения : ", element->destination);
                    break;
                case 4:
                    winput(L"Новый час : ", element->time.hour, check_hour);
                    break;
                case 5:
                    winput(L"Новая минута : ", element->time.minute, check_minute);
                    break;
                case 6:
                    winput(L"Новая секунда : ", element->time.second, check_second);
                    break;
                default:
                    wcolor_out(L"Такой опции нет \n", "red");
                    break;
                }
            }
            break;
        case 11:
            list->save();
            break;
        case 12:
            list->read();
            break;
        case 13:
            winput(L"Имя фала в который сохранить : ", input_str);
            list->save(input_str);
            break;
        case 14:
            winput(L"Имя фала из которого загрузить : ", input_str);
            list->read(input_str);
            break;
        case 0:
            delete list;

            wcout << L"--------------" << endl;
            wcout << L"История комманд" << endl;
            for(int i : v)
            {
                wcout << i << endl;
            }
            wcout << L"--------------" << endl;

            // из-за хранения команд существует утечка памяти
            // стоит заметить, что эта история существует только для удобства отладки
            // (или например составления тестов)
            // так что в Release версии ее не будет, и следовательно это не ошибка
```

```
            DumpMemoryLeaks
            return 0;
        default:
            wcolor_out(L"Такой опции нет \n", "red");
            break;
        }
    }
}

bool check_hour     (int hour  )
{
    if (hour > -1 && hour < 24)
    {
        return true;
    }
    else
    {
        wcolor_out(L"-----------------------------------------------\n", "red");
        wcolor_out(L"Неверное время! \n", "red");
        wcolor_out(L"Час должен быть больше 0 и меньше 24! \n", "red");
        wcolor_out(L"-----------------------------------------------\n", "red");
        return false;
    }
}
bool check_minute   (int minute)
{
    if (minute > -1 && minute < 60)
    {
        return true;
    }
    else
    {
        wcolor_out(L"-----------------------------------------------\n", "red");
        wcolor_out(L"Неверное время! \n", "red");
        wcolor_out(L"Минута должна быть больше 0 и меньше 60! \n", "red");
        wcolor_out(L"-----------------------------------------------\n", "red");
        return false;
    }
}
bool check_second   (int second)
{
    if (second > -1 && second < 60)
    {
        return true;
    }
    else
    {
        wcolor_out(L"-----------------------------------------------\n", "red");
        wcolor_out(L"Неверное время! \n", "red");
        wcolor_out(L"Секунда должна быть больше 0 и меньше 60! \n", "red");
        wcolor_out(L"-----------------------------------------------\n", "red");
        return false;
    }
}
bool check_positive (int var )
{
    if (var >= 0)
    {
        return true;
    }
    else
    {
        wcolor_out(L"-----------------------------------------------\n", "red");
        wcolor_out(L"Неверные двнные! \n", "red");
        wcolor_out(L"Значение должно быть положительным! \n", "red");
```

```
        wcolor_out(L"-----------------------------------------\n", "red");
        return false;
    }
}


void test_rus_symb()
{
    wchar_t t = ' ';
    while (t != '0' && t != 0)
    {
        wcin >> t;
        wcout << L"in = " << wchar_t(t) << L"| out = " << wchar_t(int(t) + 848) << L"||"
<< endl; // A == 192
        wcout << "is rus(s5) = " << is_russ(t) << " " << int(t) << endl;
        correct_to_rus_symbol(t);
        wcout << "correct_to_rus_symbol_str(t) = " << t << endl;
        wcout << "is rus(s5) = " << is_russ(t-848) << " " << int(t-848) << endl;
        correct_from_rus_symbol(t);
        wcout << "correct_from_rus_symbol_str(t) = " << t << endl;
    }

    // for (int i = 192; i < 255; i++)
    // {
    //     wcout << wchar_t(i+848) << " = " << int(i) << endl;
    // }

    return;

    // for (wchar_t i = L'А'; i != L'я'; i++)
    // {
    //     wcout << L"[1] " << wchar_t(i) << " = " << int(i) << endl;
    // }

    // for (wchar_t i = 1040; i < 1040+32; i++)
    // {
    //     wcin >> t;
    //     wcout << L"[2] " << wchar_t(t) << " = " << int(t) << endl;
    // }

    // wchar_t s1, s2, s3, s4, s5;
    // wcout << L"А = " <<  "is_russ(L'А') = " << is_russ(L'А') << endl;
    // wcout << L"Я = " <<  "is_russ(L'Я') = " << is_russ(L'Я') << endl;
    // wcout << L"а = " <<  "is_russ(L'а') = " << is_russ(L'а') << endl;
    // wcout << L"я = " <<  "is_russ(L'я') = " << is_russ(L'я') << endl;

    // s1 = L'А';
    // correct_to_rus_symbol(s1);
    // wcout << "correct_to_rus_symbol_str(L'А') = " << s1 << endl;
    // wcout << "is rus(L'А') = " << is_russ( int(s1)+848) << endl;
    // s2 = L'Я';
    // correct_to_rus_symbol(s2);
    // wcout << "correct_to_rus_symbol_str(L'Я') = " << s2 << endl;
    // wcout << "is rus(L'Я') = " << is_russ( int(s2)+848) << endl;
    // s3 = L'а';
    // correct_to_rus_symbol(s3);
    // wcout << "correct_to_rus_symbol_str(L'а') = " << s3 << endl;
    // wcout << "is rus(L'а') = " << is_russ( int(s3)+848) << endl;
    // s4 = L'я';
    // correct_to_rus_symbol(s4);
    // wcout << "correct_to_rus_symbol_str(L'я') = " << s4 << endl;
    // wcout << "is rus(L'я') = " << is_russ( int(s4)+848) << endl;

    // correct_from_rus_symbol(s1);
    // wcout << "correct_from_rus_symbol_str(L'А') = " << s1 << endl;
```

```
    // correct_from_rus_symbol(s2);
    // wcout << "correct_from_rus_symbol_str(L'Я') = " << s2 << endl;
    // correct_from_rus_symbol(s3);
    // wcout << "correct_from_rus_symbol_str(L'a') = " << s3 << endl;
    // correct_from_rus_symbol(s4);
    // wcout << "correct_from_rus_symbol_str(L'я') = " << s4 << endl;
}
```

Источники

[1] https://ru.wikipedia.org/wiki/История_железнодорожного_транспорта