ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА

ИНСТИТУТ РАДИОЭЛЕКТРОНИКИ И ИНФОРМАЦИОННЫХ

ТЕХНОЛОГИЙ

Кафедра «Информационные радиосистемы»

КУРСОВАЯ РАБОТА

«Рассматривается расписание движения пригородных электропоездов и поездов дальнего следования. Каждая запись в расписании должна содержать следующую информацию: тип транспорта, направление движения, время отправления, режим отправления (ежедневно, по четным/нечетным дням, кроме праздничных дней, конкретный день недели), время прибытия. Определить поезд указанного типа, время отправления которого наиболее подходит к заданному времени »

Выполнил:

студент группы 24-Рз И. И. Иванов

Проверил:

доцент кафедры ИРС С. Б. Сидоров

Нижний Новгород

2025

Содержание

Введение

В работе рассматривается решение задачи разработки прикладной программы поиска поездов из общего списка, по ближайшему подходящему времени отправления и типу.

Одним из способов преодоления сложности данной задачи является использование отдельной функции для выполнения обработки. Этот подход позволяет свести задачу к последовательному решению более простых задач. Его использование позволяет уменьшить затраты на отладку и повысить надёжность программы. Также существенным моментом является использование структурных типов данных для адекватного отображения сущностей проблемной области в область программной реализации.

В начале основной части отчёта приводится точная формализованная постановка задачи с указанием полного набора операций, выполнение которых должна обеспечивать прикладная программа.

В руководстве пользователя раскрывается назначение программы, её возможности и выполняемые операции. Подробно объясняются правила пользования программой и приводятся конкретные примеры диалога с пользователем.

В руководстве программиста рассматриваются вопросы внутренней организации программы, в том числе перечень функций и их взаимодействие. Кроме этого описываются используемые структуры данных и наиболее важные, и интересные с точки зрения их реализации алгоритмы.

В заключении делаются выводы о полноте решения поставленной задачи, а также приводится ряд соображений по возможным направлениям доработки полученной прикладной программы. Также приведена техническая информация, включающая листинги программы.

1. Постановка задачи

Рассматривается модель информации о расписании поездов. Описание одного поезда представлено совокупностью свойств:

- Тип поезда;
- Направление движения;
- Время отправления;
- Режим отправления (ежедневно, по четным/нечетным дням, кроме праздничных дней, конкретный день недели);
- Время прибытия.

В области программной реализации модель поезда имеет вид структурного типа данных, таким образом, модель расписания будет являться массивом отдельных элементов – поездов.

Рассматривается расписание поездов с конечным количеством элементов. Требуется получить программную реализацию заданной обработки такого набора структурированных данных.

Все данные, необходимые для обработки, запрашиваются у пользователя: значения элементов, входящих в набор, и дополнительная информация, необходимая для выполнения конкретной обработки данных.

Обработка должна быть реализована отдельной функцией. При этом вся необходимая для выполнения обработки информация должна передаваться в функцию через список аргументов. Результат обработки набора структурированных данных должен передаваться из функции также через список аргументов.

Полученные результаты обработки должны быть выданы на монитор, то есть на стандартное устройство вывода.

Рассматривается расписание движения пригородных электропоездов и поездов дальнего следования. Каждая запись в расписании должна содержать следующую информацию: тип транспорта, направление движения, время отправления, режим отправления (ежедневно, по четным/нечетным дням, кроме праздничных дней, конкретный день недели), время прибытия. Определить поезд указанного типа, время отправления которого наиболее подходит к заданному времени.

2. Руководство пользователя

Программа предназначена для нахождения наиболее подходящего поезда, по заданным пользователем параметрам, в режиме диалога с пользователем.

Программа позволяет пользователю задать параметры поездов, и их количество. Результаты обработки выводятся на экран монитора, через консоль (терминал).

Запуск программы осуществляется либо через командную строку с вводом имени исполняемого файла, полученного в результате компиляции, с последующим нажатием клавиши *Enter*, либо иным способом в зависимости от операционной системы.

Программа является интерактивным консольным приложением. Весь диалог с пользователем осуществляется в текстовом режиме.

Сначала пользователю предлагается указать количество поездов в рассматриваемом расписании:

- `Enter number of trains:`

После этого у пользователя запрашиваются параметры каждого поезда из расписания:

- `Enter type for train # (0 – Suburban, 1 – LongDistance):`
- `Enter direction for train #:`
- `Enter departure time for train # (HH MM):`
- `Enter departure mode for train #:`
- `Enter arrival time for train # (HH MM):`

После ввода параметров каждого поезда, у пользователя запрашиваются критерии подбора – тип поезда (Дальнего следования или пригородный), время отправления.

- Enter train type to search (0 – Suburban, 1 – LongDistance):
- Enter desired departure time (HH MM):

По завершении ввода поездов, составляющих расписание, и параметров подбора, будет произведен поиск наиболее подходящего поезда. По завершении обработки выводится результат следующего вида:

- Most suitable train:
  - *(#. (Тип поезда) (Направление) (Время отправления) (Режим отправления) (Время прибытия))*

Если подходящих поездов не найдно, выводится соответствующее сообщение:

- No trains are applicable.

С логической точки зрения, даже при наличии всего одного поезда в расписании, программа, в данной конфигурации, укажет именно на него, при условии соблюдения типа поезда (Дальнего следования / Пригородный).

По завершении выдачи результатов программа завершает свою работу.

3. Руководство программиста

3.1 Структура программы

Прикладная программа разработана с использованием принципов императивного программирования. Она является совокупностью взаимодействующих функций. Структура программы представлена на рис. 1.

Рисунок 1. Структура программы

Программа состоит из двух функций, назначение каждой из которых приведено ниже:

1. `main` – основная функция приложения
2. `select_trains` – функция поиска поездов в расписании
3. `print_trains` – функция вывода результатов подбора

3.2 Структура данных

Для описания поезда определяем структурный тип данных `Train`. По условию задачи, для описания одного поезда, используется:

- Тип поезда;
- Направление движения;
- Время отправления;
- Режим отправления (ежедневно, по четным/нечетным дням, кроме праздничных дней, конкретный день недели);

- Время прибытия.

Так же, добавим порядковый номер поезда. Таким образом получаем следующее определение структурного типа данных, для описания понятия “Поезд”:

```
typedef struct {
    int index;
    bool is_long_distance;
    char direction[DIR_LEN];
    int departure_time;
    char departure_mode[MODE_LEN];
    int arrival_time;
} Train;
```

Для предоставления функциональности ввода количества поездов, во время непосредственно выполнения программы пользователем, используется выделение памяти во время работы программы, посредством `malloc()`. Для выделения памяти для *n* поездов типа `Train`, необходимо выделять память посредством умножения размера структуры `Train` на количество поездов, введенных пользователем. Таким образом мы получаем адрес выделенной памяти, и указываем программе, как с ним необходимо обращаться, определяя его как массив элементов типа `Train`.

Для представления расписания поездов используем массив из элементов `Train *trains_depot = malloc(sizeof(Train) * train_count);` Максимально допустимое количество элементов в массиве ограничим целочисленной переменной `train_count`, ее значение запрашивается у пользователя.

Для представления наиболее подходящего поезда, так же, используем тип `Train`. По условию задачи, необходимо выдать информацию только о самом подходящем поезде. Исходя из этого наиболее целесообразно хранить информацию только об одном поезде, представив как `Train trains_selected;`. Это наиболее логичная и прямая запись, для переменной, массива из одного элемента. Но в таком случае изменится способ передачи этой переменной через аргумент, придется передавать адрес переменной в памяти через & и “ловить” адрес как указатель, разъименовывая его через *. В то время как запись `Train *trains_selected` и `Train trains_selected[]` являются равнозначными, для упрощения кода, было принято решение объявить массив из одного элемента. В таком случае запись кода выглядит более лаконично и органично. Выбранный поезд представляем как массив из одного элемента – `Train trains_selected[1]`.

3.3 Алгоритм поиска наиболее подходящего поезда

Поставленная, в условии, задача отбора наиболее подходящего поезда по параметрам - тип поезда и время отправления сводится к проверке этих условий у каждого поезда и сохранению наиболее подходящего, в последствии сравнивая уже с ним, структура алгоритма представлена на рисунке 2.

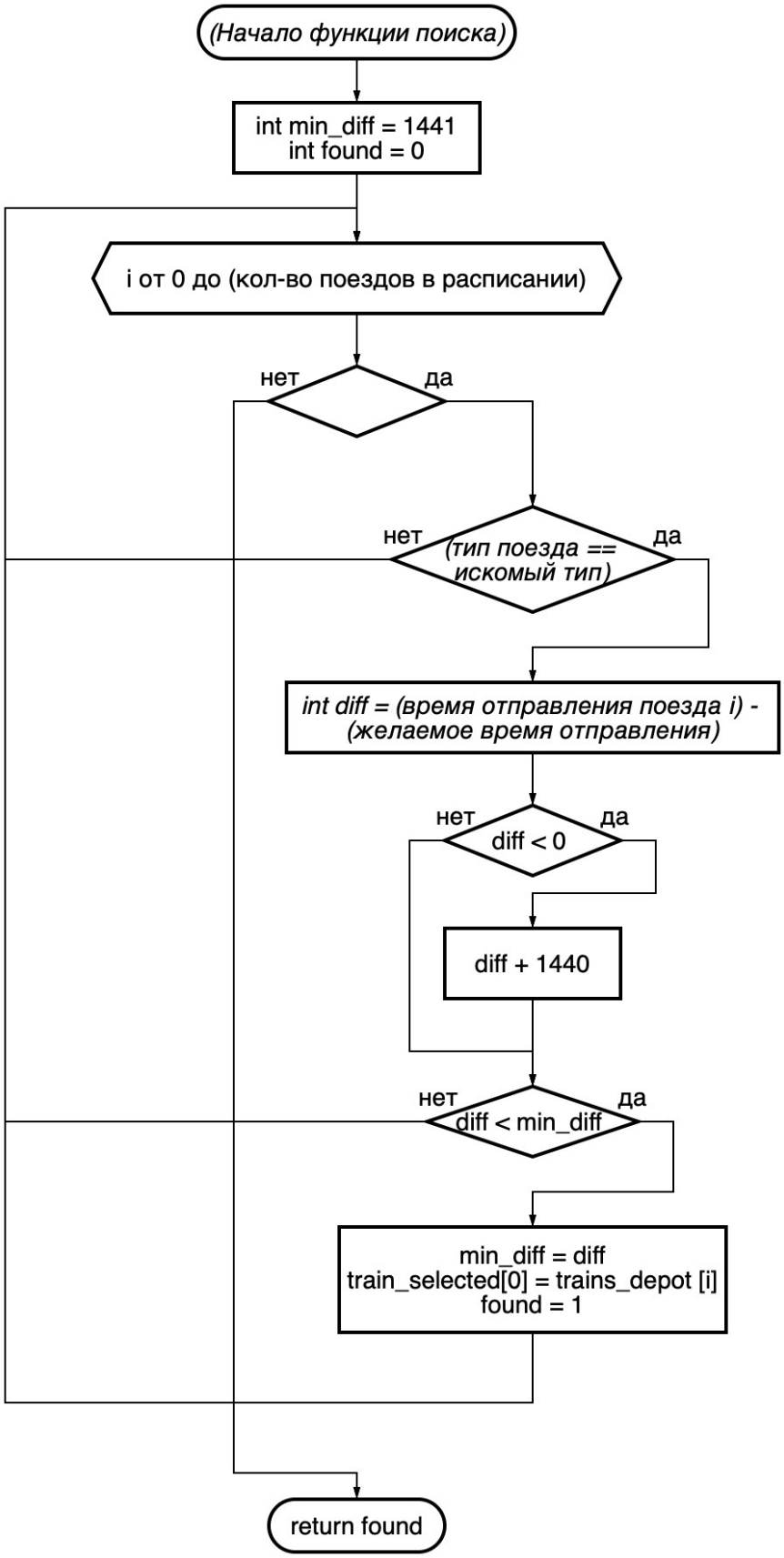

Рисунок 2. Алгоритм функции поиска

Заключение

В данной работе задача разработки прикладной программы поиска наиболее подходящего поезда из расписания, была решена с использованием императивного программирования. На основании проведённой отладки и испытаний с помощью контрольных примеров можно сделать вывод, что полученная прикладная программа решает поставленную задачу правильно и в полном объёме.

Список литературы

1. Керниган, Брайан У., Ритчи, Деннис М. Язык программирования C, 2-е издание. [пер. с анг.] / Б.У. Керниган, Д.М. Ритчи – М.: Вильямс, 2007.
2. Павловская, Т.А. C/C++. Программирование на языке высокого уровня: учебник для ВУЗов / Т.А. Павловская. – СПб.: Питер, 2009.
3. Орлов, С.А. Технологии разработки программного обеспечения. учеб. пособие. 2-е изд./ С.А. Орлов, – СПб.: Питер, 2003. – 480 с.: ил.
4. Борисенко, В.В. Основы программирования / В.В.Борисенко, – Интернет-университет информационных технологий – ИНТУИТ.ру, 328 стр. – 2005 г.
5. Шилдт, Г. Полный справочник по C: учеб. пособие / Г. Шилдт. – 4-е изд. – М.: Изд. дом "Вильямс", 2008.
6. Костюкова, Н.И. Язык Си и особенности работы с ним / Н.И. Костюкова, Н.А. Калинина – Интернет-университет информационных технологий – ИНТУИТ.ру, 208 стр. – 2006 г.

Приложение А

Заголовочный файл

```
#ifndef TRAIN_H
#define TRAIN_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define MODE_LEN 32
#define DIR_LEN 64

typedef struct {
    int index;
    bool is_long_distance; // false: suburban, true: long-distance
    char direction[DIR_LEN];
    int departure_time; // minutes since midnight
    char departure_mode[MODE_LEN];
    int arrival_time; // minutes since midnight
} Train;

int select_trains(Train trains_depot[], Train trains_selected[], int
train_count, bool is_long_distance, int desired_time);
void print_trains(Train trains_selected[], int selected_num);

#endif // TRAIN_H
```

Основная программа

```
#include "train.h"

int select_trains(Train trains_depot[], Train trains_selected[], int
train_count, bool is_long_distance, int desired_time) {
    int min_diff = 1441; // More than a day in minutes
    int found = 0;
    for (int i = 0; i < train_count; ++i) {
        if (trains_depot[i].is_long_distance == is_long_distance) {
            int dep_time = trains_depot[i].departure_time;
            int diff = dep_time - desired_time;
            if (diff < 0) diff += 1440;  // Wrap around midnight
            if (diff < min_diff) {
                min_diff = diff;
                trains_selected[0] = trains_depot[i];
                found = 1;
            }
        }
    }
    return found;
}

void print_trains(Train trains_selected[], int selected_num) {
    if (selected_num == 0) {
        printf("No trains are applicable.\n");
        return;
    }
    printf("\nMost suitable train:\n");
    printf("%d. %s %s Dep: %02d:%02d Mode: %s Arr: %02d:%02d\n",
        trains_selected[0].index,
        trains_selected[0].is_long_distance ? "LongDistance" : "Suburban",
        trains_selected[0].direction,
        trains_selected[0].departure_time / 60,
trains_selected[0].departure_time % 60,
        trains_selected[0].departure_mode,
        trains_selected[0].arrival_time / 60, trains_selected[0].arrival_time %
60
    );
}

int main() {
    int train_count;
    printf("Enter number of trains: ");
    scanf("%d", &train_count);

    Train *trains_depot = malloc(sizeof(Train) * train_count);
    if (!trains_depot) {
```

```
        printf("Memory allocation failed.\n");
        return 1;
    }

    for (int i = 0; i < train_count; ++i) {
        trains_depot[i].index = i + 1;
        int type_input;
        printf("Enter type for train %d (0 - Suburban, 1 - LongDistance): ", i +
1);
        scanf("%d", &type_input);
        trains_depot[i].is_long_distance = (type_input == 1);
        printf("Enter direction for train %d: ", i + 1);
        scanf("%s", trains_depot[i].direction);
        int h, m;
        printf("Enter departure time for train %d (HH MM): ", i + 1);
        scanf("%d %d", &h, &m);
        trains_depot[i].departure_time = h * 60 + m;
        printf("Enter departure mode for train %d: ", i + 1);
        scanf("%s", trains_depot[i].departure_mode);
        printf("Enter arrival time for train %d (HH MM): ", i + 1);
        scanf("%d %d", &h, &m);
        trains_depot[i].arrival_time = h * 60 + m;
    }

    int type_input, h, m;
    printf("Enter train type to search (0 - Suburban, 1 - LongDistance): ");
    scanf("%d", &type_input);
    bool search_is_long_distance = (type_input == 1);
    printf("Enter desired departure time (HH MM): ");
    scanf("%d %d", &h, &m);
    int desired_time = h * 60 + m;

    Train trains_selected[1];
    int selected_num = select_trains(trains_depot, trains_selected, train_count,
search_is_long_distance, desired_time);
    print_trains(trains_selected, selected_num);

    free(trains_depot);

    return 0;
}
```