

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Отчет
по учебной (ознакомительной) практике

Студент:
гр. 250501 Лукьянов Е.О.

Руководитель:
старший преподаватель
Ковальчук А.М.

МИНСК 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 ПОСТАНОВКА ЗАДАЧИ.....	4
2 ОБЗОР ЛИТЕРАТУРЫ.....	5
3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ.....	6
3.1 Структура входных и выходных данных	6
4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ	8
4.1 Разработка схем алгоритмов.....	8
4.2 Разработка алгоритмов	8
5 РЕЗУЛЬТАТ РАБОТЫ	12
ЗАКЛЮЧЕНИЕ	15
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	16
ПРИЛОЖЕНИЕ А	17
ПРИЛОЖЕНИЕ Б.....	18
ПРИЛОЖЕНИЕ В	19
ПРИЛОЖЕНИЕ Г.....	20
ПРИЛОЖЕНИЕ Д	21

ВВЕДЕНИЕ

В настоящее время белорусская железная дорога (БЖД) является распространённым средством передвижения по регионам нашей страны. Целью данной работы является разработка программы управления расписанием поездов, а также применение ранее полученных знаний на практике.

Для реализации данной программы был выбран язык программирования Си. Язык Си является одним из самых быстрых и популярных языков во всём мире. Язык Си позволяет осуществлять большинство преобразований типов. Контроль за выполнением этих преобразований, а также проверка некоторых ошибок (например, выход за границы массива) возлагается на программиста. Реализованная в языке Си возможность напрямую манипулировать битами, байтами, словами и указателями необходима для программирования на системном уровне. Язык Си считается структурированным языком. Отличительной чертой структурированного языка является разделение кода и данных. Одним из способов решения этой проблемы является использование подпрограмм (функций), широко использующих локальные переменные. Необходимо отметить, что излишнее использование глобальных переменных может приводить к фатальным ошибкам. Как и ряд других структурированных языков, язык Си поддерживает ряд операторов цикла, условных операторов и операторов ветвления.

Язык Си содержит стандартные библиотеки, предоставляющие функции, выполняющие наиболее типичные задачи. Эти библиотеки легко могут быть подключены, а также дополнены. Язык Си позволяет разбивать программу на части и выполнять их отдельную компиляцию. Откомпилированные таким образом файлы объединяются для создания полного объектного кода. Преимущество отдельной компиляции в том, что при изменении одного файла не требуется перекомпиляции всей программы.

Важность данной работы обусловлена тем, что программа управления расписанием поездов может значительно облегчить жизнь части населения нашей страны, которая пользуется железнодорожным транспортом, обеспечивая удобный и быстрый доступ к достоверной информации.

1 ПОСТАНОВКА ЗАДАЧИ

Цель программы "Управление расписанием поездов" - создать эффективную систему управления расписанием поездов, которая позволит пользователям узнавать точную информацию об их поездках и добираться до пункта назначения по рациональным маршрутам.

Основными задачами данной программы являются выдача справки о маршруте, то есть время отправления и прибытия, расстояние между городами, поиск кратчайшего пути между ними, указание станций пересадок на маршруте (в случае их присутствия), поиск времени прибытия на станции пересадок и времени отбытия со станций пересадок (в случае их присутствия).

2 ОБЗОР ЛИТЕРАТУРЫ

2.1 Обзор методов решения поставленной задачи

Для решения задачи был выбран язык программирования Си.

Для хранения данных использовались текстовые файлы. Выбранный язык предоставляет набор функций для удобной работы с файловой системой.

Для реализации данной программы был использован модифицированный алгоритм Дейкстры для учёта станций пересадок. Алгоритм Дейкстры - это алгоритм нахождения кратчайшего пути во взвешенном графе с неотрицательными весами ребер. Алгоритм начинает работу с одной вершины графа и находит кратчайшие пути до всех остальных вершин.

Алгоритм Дейкстры работает следующим образом:

1. Изначально все вершины помечаются как не посещенные, а расстояние до всех вершин, кроме стартовой, считается бесконечностью.
2. Устанавливаем расстояние от стартовой вершины до нее самой равным 0.
3. Выбираем вершину с наименьшим расстоянием из еще не посещенных вершин и помечаем ее как посещенную.
4. Обновляем расстояния до смежных вершин через выбранную вершину. Если расстояние до смежной вершины короче через текущую вершину, то обновляем его.
5. Повторяем шаги 3 и 4, пока все вершины не будут помечены как посещенные.
6. После завершения работы алгоритма расстояния до всех вершин графа будут определены и мы можем найти кратчайший путь до любой вершины.

Алгоритм Дейкстры с модификацией для учета станций пересадок работает следующим образом:

1. Создать дополнительную вершину-пересадку для каждой пары станций, между которыми нет прямого пути.
2. Соединить каждую из этих дополнительных вершин со всеми станциями, на которые можно пересест с линии каждой из оригинальных станций.
3. Запустить алгоритм Дейкстры для поиска кратчайшего пути между стартовой и конечной станциями, учитывая при этом вершины-пересадки.
4. При построении кратчайшего пути учитывать, что на пути может быть несколько вершин-пересадок. В таком случае необходимо вывести информацию о пересадке и продолжить построение пути.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

3.1 Структура входных и выходных данных

Рисунок 3.1.1 описывает структуру данных s1. Рисунки 3.1.2 и 3.1.3 иллюстрируют структуру текстовых файлов f.txt и result.txt.

```
struct s1 // объявление структуры, хранящей информацию о городе (населённого пункта)
{
    char City[N]; // название города (населённого пункта)
    char Time1[t][N]; // массив времён отбытия на t элементов
    char Time2[t][N]; // массив времён прибытия на t элементов
    int Line; // номер железно-дорожной линии, на которой расположен город (населённый пункт)
    int Dist; // расстояние от данного города (населённого пункта) до Минска
};
```

Рис. 3.1.1 – Структура данных s1

f.txt – Блокнот

Файл Правка Формат Вид Справка

Minsk 8 00:10 05:30 06:53 07:48 09:55 12:55 14:27 16:11 18:06 20:16 06:59 07:58 09:16 10:59 12:43 15:40 16:39 19:28 21:33 22:08 00
Kolodischi 1 00:32 05:52 07:20 08:16 10:17 13:14 14:53 16:30 18:28 20:42 06:37 07:36 08:42 10:38 12:17 15:18 16:13 19:00 21:08 21:43 18
Sloboda 1 00:45 06:05 07:34 08:31 10:30 13:29 15:08 16:45 18:41 20:57 06:24 07:23 08:26 10:23 12:02 15:05 15:58 18:37 20:52 21:26 29
Smolevichi 1 00:57 06:17 07:48 08:44 10:42 13:42 15:21 16:58 18:53 21:10 06:13 07:12 08:14 10:10 11:49 14:54 15:45 18:25 20:40 21:13 40
Ost'yabr'skiy 1 01:09 06:29 07:58 08:54 10:54 13:52 15:31 17:08 19:05 21:20 06:02 07:01 08:03 09:59 11:38 14:43 15:34 18:14 20:29 21:02 50
Zhodino 1 01:27 06:58 08:07 09:03 11:05 14:01 15:41 17:17 19:16 21:29 05:46 07:03 07:53 09:49 11:28 14:30 15:24 18:04 20:19 20:52 62
Borisov 1 01:48 07:17 08:32 09:28 11:28 14:25 16:13 17:41 19:40 21:54 05:23 06:48 07:32 09:27 11:07 14:10 15:03 17:43 19:58 20:30 81
Krupki 1 02:40 08:01 09:22 10:21 12:35 15:06 17:04 18:23 20:46 22:46 04:38 05:35 06:54 08:34 10:28 13:18 14:25 17:04 19:19 19:55 120
Orsha 7 04:16 09:35 10:58 11:52 14:00 16:59 18:33 20:15 22:11 - 02:55 03:59 05:20 06:44 08:52 11:30 12:50 15:27 17:33 18:13 213
Puhovichi 2 01:25 06:45 08:08 09:03 11:10 14:10 15:42 17:26 19:21 21:31 05:45 06:44 08:00 09:45 11:30 14:25 15:24 18:15 20:18 20:53 63
Osipovich 2 02:26 07:49 09:11 10:07 12:14 15:13 16:45 18:30 20:24 22:30 04:55 05:54 07:10 08:55 10:40 13:35 14:34 17:25 19:28 20:03 107
Bobruisk 2 03:56 09:19 10:41 11:39 13:44 16:44 18:15 20:00 21:54 23:59 04:01 05:00 06:16 08:01 09:46 12:41 13:40 16:31 18:34 19:09 149
Zhlobin 2 05:16 10:30 11:52 12:48 14:55 17:55 19:24 21:09 22:05 - 02:40 03:40 04:56 06:40 08:25 11:23 12:22 15:13 17:14 17:49 214
Tihinichi 2 06:28 11:42 13:04 14:00 16:07 19:07 20:36 22:21 23:17 - 01:28 02:28 03:44 05:28 07:13 10:11 11:10 14:01 16:02 16:37 274
Gomel 2 07:16 12:30 13:52 14:49 16:55 19:55 21:24 23:09 23:59 - 00:40 01:40 02:56 04:40 06:25 09:23 10:22 13:13 15:14 15:47 300
Ratomka 3 00:32 05:54 07:17 08:00 10:18 13:20 14:50 16:33 18:29 20:39 06:34 07:34 08:51 10:35 12:10 15:15 16:14 19:03 21:08 21:43 17
Radoshkovichi 3 01:00 06:21 07:45 08:29 10:46 13:50 15:18 17:02 19:00 21:06 06:09 07:09 08:26 10:10 11:45 14:50 15:49 18:38 20:43 21:21 35
Molodechno 3 01:55 07:18 08:40 09:25 11:40 14:46 16:15 18:00 19:57 22:02 05:12 06:12 07:29 09:13 10:47 13:54 14:53 17:42 19:47 20:25 77
Smorgon 3 02:45 08:08 09:30 10:15 12:30 15:36 17:04 18:52 20:37 22:53 04:08 05:09 06:23 08:10 09:44 12:50 13:49 16:38 18:41 19:22 113
Oshmyany 3 03:20 08:43 10:05 10:50 13:05 16:11 17:40 19:37 21:12 23:34 03:38 04:37 06:00 07:48 09:23 12:30 13:28 16:18 18:20 19:00 133
Fanipol 4 00:40 06:00 07:23 08:18 10:25 13:25 14:57 16:41 18:36 20:46 06:29 07:28 08:46 10:29 12:13 15:10 16:09 18:58 21:03 21:38 22
Stolbtsy 4 01:50 07:10 08:33 09:28 11:35 14:35 16:07 17:51 19:46 21:56 05:19 06:18 07:36 09:19 11:03 14:00 14:59 17:48 19:53 20:28 74
Baranovich 4 03:00 08:20 09:43 10:38 12:45 15:45 17:17 19:01 20:56 23:06 04:09 05:08 06:26 08:09 09:53 12:50 13:49 16:38 18:43 19:18 140
Beryoz 4 04:30 09:50 11:13 12:08 14:15 17:15 18:47 20:31 22:26 - 02:39 03:38 04:56 06:39 08:23 11:20 12:19 15:08 17:13 17:48 243
Zhabinka 4 05:40 11:00 12:23 13:18 15:25 18:25 19:57 21:41 23:30 - 01:29 02:28 03:46 05:29 07:13 10:10 11:09 13:58 16:03 16:38 316
Brest 4 06:15 11:35 12:58 13:53 16:00 19:00 20:32 22:16 23:59 - 00:54 01:53 03:11 04:54 06:38 09:35 10:34 13:23 15:28 16:03 342
Shklov 5 05:10 10:30 11:52 12:46 14:54 17:53 19:27 21:09 23:05 - 02:01 03:05 04:26 05:50 07:58 10:36 11:56 14:33 16:39 17:19 285
Mogilyov 5 06:01 11:21 12:42 13:36 15:44 16:43 20:17 22:00 23:56 - 01:10 02:14 03:35 04:59 07:07 09:45 11:05 13:42 15:48 16:28 320
Bulinichi 5 06:17 11:37 12:58 13:52 16:00 16:59 20:33 22:16 - - 00:54 01:58 03:19 04:43 06:51 09:29 10:49 13:26 15:32 16:12 330
Byhov 5 07:07 12:27 13:48 14:42 16:50 17:49 21:23 23:06 - - 00:04 01:08 02:29 03:53 06:00 08:39 09:59 12:36 14:42 15:22 371
Zamostochie 6 05:26 10:45 12:08 13:02 15:10 18:09 19:43 21:25 23:22 - 01:45 02:49 04:10 05:34 07:42 10:20 11:40 14:17 16:23 17:03 273
Vitebsk 6 05:59 11:15 12:38 13:32 15:41 18:40 20:13 21:55 23:52 - 01:15 02:19 03:40 05:04 07:12 09:50 11:10 13:47 15:53 16:33 295

Рис. 3.1.2 – Файл «f.txt», содержащий информацию о всех городах

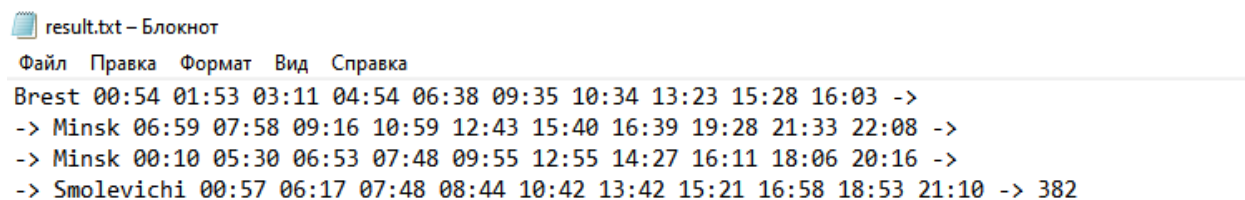


Рис. 3.1.3 – Файл «result.txt», содержащий информацию о поездке пользователя

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1 Разработка схем алгоритмов

Схема алгоритма функции main() представлена в приложении А.

Функция napravl() определяет направление поездки. Схема алгоритма представлена в приложении Б.

4.2 Разработка алгоритмов

Алгоритм по шагам функции dijkstra():

1. Начало алгоритма.
2. Объявление и инициализация переменных:

Вх:

int graph[V][V] – граф, хранящий расстояния между всеми вершинами.

int start – индекс города, из которого едет пользователь.

int end – индекс города, в который едет пользователь.

struct s1* info – массив структур с информацией о всех городах.

struct s2* infoWay – массив структур для хранения информации о поездке пользователя.

int* num – указатель на переменную, хранящую информацию о количестве пересадок на маршруте.

Промеж:

int dist[V] – массив расстояний от стартовой вершины.

int visited[V] – массив посещенных вершин.

int prev[V] – массив предыдущих вершин на пути.

int number = -1 – переменная для хранения кол-ва пересадок на маршруте.

int numb = 0 – переменная для записи нужных времён отбытия со станций пересадок и прибытия на станции пересадок.

int current = end – текущее положение начинается с конечного города поездки для прохождения по всем городам на маршруте.

int IndexTransfer – индекс города, где будет осуществлена ближайшая пересадка.

int nap – переменная для хранения направления поездки.

int ind = prev[current] – сохранение последовательности предыдущих вершин на пути.

int flag = 0 – город не является пересадкой.

Вых:

struct s2* infoWay – массив структур с информацией о поездке пользователя.

3. Цикл от $i = 0$ до V инициализации переменных для каждого города.
4. $\text{Dist}[i] = \text{INT_MAX}$ – инициализация всех расстояний как “бесконечность”. $\text{Visited}[i] = 0$ – отметка всех вершин как не посещённых. $\text{Prev}[i] = -1$ – отсутствие предыдущих вершин на пути.
5. Конец цикла i .
6. $\text{Dist}[\text{start}] = 0$ – расстояние от начального города до самого себя равняется 0.
7. Цикл от $i = 0$ до $V - 1$ для поиска кратчайшего пути для всех вершин графа.
8. $\text{int } u = \text{min_distance}(\text{dist}, \text{visited})$ – вызов функции для поиска вершины с наименьшей стоимостью среди вершин, ещё не включённых в кратчайший путь. $\text{Visited}[u] = 1$ – отметка выбранной вершины как посещённой.
9. Цикл от $v = 0$ до V для обновления расстояний до смежных вершин, учитывая станции пересадок.
10. Если вершина ещё не включена в кратчайший путь, есть ребро из выбранной вершины до смежной вершины и расстояние от источника до выбранной вершины плюс стоимость ребра меньше текущего расстояния до смежной вершины, то обновляем расстояние до смежной вершины: $\text{dist}[v] = \text{dist}[u] + \text{graph}[u][v]$; $\text{prev}[v] = u$;
11. Конец цикла v .
12. Конец цикла i .
13. Если расстояние до конечного города поездки равно “бесконечности”, то путь не найден.
14. Если начальный или конечный город является городом-пересадкой, то $\text{flag} = 1$.
15. Пока не пройдём всю последовательность предыдущих вершин на маршруте, выполняем пункт 16.
16. Если линии данной станции и предыдущей на маршруте разные, то $\text{number} = \text{number} + 1$ и становимся на предыдущую вершину на маршруте ($\text{current} = \text{prev}[\text{current}]$).
17. $\text{Numb} = \text{number}$ – сохраняем кол-во пересадок на маршруте для чтения времён прибытия на станцию пересадки и отбытия со станции пересадки.
18. Если $\text{number} > 0$, то выполняем пункты 19.
19. Пока не пройдём всю последовательность предыдущих вершин на маршруте, выполняем пункты 20-22.

20. Если линии данной станции и предыдущей на маршруте разные, то с помощью функции `strcpy(infoWay[0].Peresadki[numb-1], info[prev[current]].City)` копируем название города-пересадки с конца. `Numb = numb - 1` – уменьшаем кол-во оставшихся пересадок.
21. Если `numb == 0`, то выходим из цикла, так как названия всех городов-пересадок уже скопированы.
22. Становимся на предыдущую вершину на маршруте.
23. С помощью функции `IndexTransfer = find_city_index(info, infoWay[0].Peresadki[0])` получаем индекс ближайшей пересадки на маршруте.
24. С помощью функции `nap = napravl(info, start, IndexTransfer, end, number)` определяем направление поездки (если `nap = 1`, то поездка от центра, если `nap = 0`, то к центру).
25. С помощью функции `infoWay = GetTime(info, infoWay, start, end, nap, number)` считываем нужные времена отбытия из начального города и прибытия в конечный город.
26. С помощью функции `infoWay = GetTimeOfPeresadki(info, infoWay, nap, number)` считываем нужные времена прибытия в город-пересадку и отбытия из него.
27. `infoWay[0].Distance = dist[end]` – сохраняем кратчайшее расстояние между начальным и конечным городами.
28. `*num = number` – сохраняем кол-во пересадок для дальнейшего использования.
29. Возвращаем `infoWay`.
30. Конец алгоритма.

Алгоритм по шагам функции `min_distance()`:

1. Начало алгоритма.
2. Объявление и инициализация переменных:
Вх:
`int dist[V]` – массив расстояний от стартовой вершины.
`int visited[V]` – массив посещенных вершин.
`int min = INT_MAX` – изначальное расстояние от источника принимается за бесконечно большое число.
Промеж:
`int min` – кратчайшее расстояние до вершины.
`int min_index` – индекса вершины с кратчайшим расстоянием
Вых:

- `int min_index` – переменная направления поездки (если `napravl = 1`, то поездка от центра, если `napravl = 0`, то поездка к центру).
3. Если `num == 0`, то второе расстояние принимаем равным расстоянию до города, в который едет пользователь (`dist2 = info[end].Dist`).
 4. Иначе второе расстояние принимаем равным расстоянию до города, в который едет пользователь (`dist2 = info[end].Dist`).
 5. Если $(dist1 - dist2) < 0$, то `napravl = 1`, то есть поездка от центра, иначе `napravl = 0`, то есть поездка к центру.
 6. Возвращаем `napravl`.
 7. Конец алгоритма.

Алгоритм по шагам функции `napravl()`:

1. Начало алгоритма.
2. Объявление и инициализация переменных:

Вх:

`int index1` – индекс города, из которого едет пользователь.

`int index2` – индекс ближайшего города-пересадки (если существует).

`int end` – индекс города, в который едет пользователь.

`int num` – количество пересадок на маршруте.

`struct s1* info` – массив структур с информацией о всех городах.

Промеж:

`int dist1 = info[index1].Dist` – расстояние до начального города.

`int dist2` – расстояние до конечного города или до ближайшего города-пересадки.

`int napravl` – переменная для хранения направления поездки (если `napravl = 1`, то поездка от центра, если `napravl = 0`, то поездка к центру).

Вых:

`int napravl` – переменная направления поездки (если `napravl = 1`, то поездка от центра, если `napravl = 0`, то поездка к центру).

3. Если `num == 0`, то второе расстояние принимаем равным расстоянию до города, в который едет пользователь (`dist2 = info[end].Dist`).
4. Иначе второе расстояние принимаем равным расстоянию до города, в который едет пользователь (`dist2 = info[end].Dist`).
5. Если $(dist1 - dist2) < 0$, то `napravl = 1`, то есть поездка от центра, иначе `napravl = 0`, то есть поездка к центру.
6. Возвращаем `napravl`.
7. Конец алгоритма.

5 РЕЗУЛЬТАТ РАБОТЫ

Допустим пользователь вёл, что собирается поехать из Бреста в Смолевичи.

На рисунках 5.1-5.6 представлены пример работы программы по исходным данным.

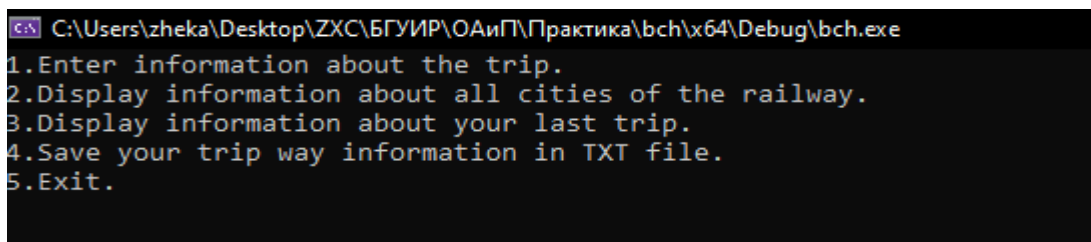
A screenshot of a Windows command prompt window. The title bar at the top shows the file path: C:\Users\zheka\Desktop\ZXC\БГУИР\ОАиП\Практика\bch\x64\Debug\bch.exe. The window contains a list of five numbered options in a monospaced font:
1.Enter information about the trip.
2.Display information about all cities of the railway.
3.Display information about your last trip.
4.Save your trip way information in TXT file.
5.Exit.

Рис. 5.1 – Меню пользователя

C:\Users\zheka\Desktop\ZXC\БГУИР\ОАиП\Практика\bch\x64\Debug\bch.exe

The information about all cities:

City	Time1	Time2	Distance
Minsk	00:10	06:59	0
	05:30	07:58	
	06:53	09:16	
	07:48	10:59	
	09:55	12:43	
	12:55	15:40	
	14:27	16:39	
	16:11	19:28	
	18:06	21:33	
	20:16	22:08	
Kolodischi	00:32	06:37	18
	05:52	07:36	
	07:20	08:42	
	08:16	10:38	
	10:17	12:17	
	13:14	15:18	
	14:53	16:13	
	16:30	19:00	
	18:28	21:08	
	20:42	21:43	
Sloboda	00:45	06:24	29
	06:05	07:23	
	07:34	08:26	
	08:31	10:23	
	10:30	12:02	
	13:29	15:05	
	15:08	15:58	
	16:45	18:37	
	18:41	20:52	
	20:57	21:26	
Smolevichi	00:57	06:13	40
	06:17	07:12	
	07:48	08:14	
	08:44	10:10	
	10:42	11:49	
	13:42	14:54	
	15:21	15:45	
	16:58	18:25	
	18:53	20:40	
	21:10	21:13	

Рис. 5.2 – Вывод информации о всех городах

```
C:\Users\zheka\Desktop\ZXC\БГУИР\ОАиП\Практика\bch\x64\Debug\bch.exe

Enter the city, where you are coming from: Brest

Enter the city, where you are going: Smolevichi

Enter the any button to return the menu:
```

Рис. 5.3 – Ввод названия городов, из которого и в который едет пользователь

```
C:\Users\zheka\Desktop\ZXC\БГУИР\ОАиП\Практика\bch\x64\Debug\bch.exe
The information about your trip:
+-----+-----+-----+-----+-----+-----+-----+-----+
| CityFrom | TimeFrom | Peresadka | Time1 | | Time2 | CityTo | TimeTo | Distance |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Brest    | 00:54    | Minsk     | 06:59  | | 00:10  | Smolevichi | 00:57  | 382      |
|          | 01:53    |           | 07:58  | | 05:30  |           | 06:17  |          |
|          | 03:11    |           | 09:16  | | 06:53  |           | 07:48  |          |
|          | 04:54    |           | 10:59  | | 07:48  |           | 08:44  |          |
|          | 06:38    |           | 12:43  | | 09:55  |           | 10:42  |          |
|          | 09:35    |           | 15:40  | | 12:55  |           | 13:42  |          |
|          | 10:34    |           | 16:39  | | 14:27  |           | 15:21  |          |
|          | 13:23    |           | 19:28  | | 16:11  |           | 16:58  |          |
|          | 15:28    |           | 21:33  | | 18:06  |           | 18:53  |          |
|          | 16:03    |           | 22:08  | | 20:16  |           | 21:10  |          |
+-----+-----+-----+-----+-----+-----+-----+-----+

Enter the any button to return the menu:
```

Рис. 5.4 – Вывод информации о поездке пользователя

```
C:\Users\zheka\Desktop\ZXC\БГУИР\ОАиП\Практика\bch\x64\Debug\bch.exe
TXT file is successfully created. Name of this file is result.txt.

Enter the any button to return the menu: _
```

Рис. 5.5 – Сохранение информации о поездке в текстовый файл result.txt

```
Консоль отладки Microsoft Visual Studio
Thank you fo using our service. Goodbye!
C:\Users\zheka\Desktop\ZXC\БГУИР\ОАиП\Практика\bch\x64\Debug\bch.exe (процесс 9244) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно: _
```

Рис. 5.6 – Конец программы

ЗАКЛЮЧЕНИЕ

В результате выполнения учебной практики были выполнены первоначально заданные цели, закреплены теоретические и практические знания в использовании структур данных, работы с файлами и т.д.

На сегодняшний день существует огромное количество программ, которые анализируют введённую информацию, а также позволяют пользователям взаимодействовать с ней. Данная программа предназначена для таких же целей. Пример приложения с такими же задачами: БЧ (беларуская чыгунка).

Дальнейшее развитие программы может включать в себя:

1. Расширение функционала программы, включая возможность расчёта и вывода стоимости поездки, бронирования билетов, выбор определённого вагона и места и т.д.
2. Добавление большего количества железнодорожных линий и разных видов поездов (линии эконом-класс, линии бизнес-класс, региональные линии, городские линии и т.д.).
3. Графический интерфейс.

Для успешного запуска данного программного средства и работы с ним требуется следующие системные требования:

- ОС Windows 10
- Среда разработки Microsoft Visual Studio 2022
- 64 – разрядная архитектура (x64)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Луцик Ю.А., Ковальчук А.М., Сасин Е.А. – Учебное пособие по курсу «Основы алгоритмизации и программирования: язык С». – Минск: БГУИР, 2015 г.
2. Луцик Ю.А., Ковальчук А.М., Лукьянова И.В., Бушкевич А.В. – Лабораторный практикум по курсу «Основы алгоритмизации и программирования» (в 2-х частях). – Минск: БГУИР, 2008 г.
3. Демидович, Е. М. Основы алгоритмизации и программирования. Язык Си / Е. М. Демидович. – СПб. : БХВ-Петербург, 2008. – 440 с.
4. Белорусская железная дорога [Электронный ресурс]. – Режим доступа: <https://railwayz.info/photolines/rw/13> Дата доступа: 25.04.2023.

ПРИЛОЖЕНИЕ А
(обязательное)

Схема алгоритма функции main()

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема алгоритма функции `pravl()`

ПРИЛОЖЕНИЕ В
(обязательное)

Код файла bch.cpp

ПРИЛОЖЕНИЕ Г
(обязательное)

Код файла functions.h

ПРИЛОЖЕНИЕ Д
(обязательное)

Код файла functions.cpp