

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	7
1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ	8
1.1 Введение	8
1.1.1 Наименование программы	8
1.1.2 Краткая характеристика области применения программы	8
1.2 Основание для разработки	8
1.3 Назначения разработки	9
1.4 Требования, предъявляемые к программе	9
1.4.1 Требования к функциональным характеристикам программы	9
1.4.2 Требования к техническим средствам, используемым при работе программы	9
1.4.3 Требования к языкам программы и среде разработки программы	9
1.4.4 Требования к информационным структурам на входе и выходе программы	10
1.5 Требования к программной документации	10
1.6 Этапы разработки	10
2 ОБЗОР СПОСОБОВ ОРГАНИЗАЦИИ ДАННЫХ И ОБОСНОВАНИЕ ВЫБОРА СТРУКТУРЫ ДАННЫХ ДЛЯ ЭФФЕКТИВНОГО ВЫПОЛНЕНИЯ ОПЕРАЦИЙ	11
2.1 Бинарное дерево поиска	11
2.2 Словарь	11
2.3 Хэш-таблица	12
2.4 Очередь	12
2.5 Массив	12

2.6 Выбор структуры данных.....	13
3 ОПИСАНИЕ ПРОГРАММЫ	14
3.1 Общие сведения.....	14
3.1.1 Наименование программы	14
3.1.2 Программное обеспечение, необходимое для функционирования программы.....	14
3.1.2 Язык программирования, на котором написана программа.....	14
3.2 Функциональное назначение программы (классы решаемых задач и функциональные ограничения на применения).....	15
3.3 Описание логической структуры программы	15
3.3.1 Алгоритмы, используемые в программе	15
3.3.1.1 Алгоритм запуска программы.	16
3.3.1.2 Алгоритм создания пустого словаря.....	18
3.3.1.3 Алгоритм добавления элемента в словарь и изменения элемента в словаре	18
3.3.1.4 Алгоритм исключения элемента из словаря	19
3.3.1.5 Алгоритм поиска элемента по ключу	19
3.3.1.6 Алгоритм вывода словаря в порядке возрастания значения ключей	20
3.3.1.7 Алгоритм чтения данных из текстового файла.....	20
3.3.1.8 Алгоритм записи данных в текстовый файл	21
3.3.2 Структура программы с описанием функций составных частей и связей между ними.....	22
Технические средства, которые используются при работе программы.....	23
3.5 Вызов программы.....	23
3.6 Входные данные (организация и предварительная подготовка входных данных).....	26

3.7 Выходные данне	26
ЗАКЛЮЧЕНИЕ	27
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	28
Приложение А	29

ВВЕДЕНИЕ

Словарь это одна из самых основных структур данных, имеющая обширную область применения. Они используются когда нужно быстро находить или изменять данные, имея “ключ”.

Словарь является одной из самых популярных структур данных благодаря своей простоте и скорости считывания/ перезаписи данных. Телефонные книги, базы данных пользователей, списки книг, фильмов, игр и т.п. реализуется на словарях.

Цель курсовой работы: получение практических навыков в иллюстрации области предусмотреть программирования в разработке приложения с интуитивно понятным интерфейсом по теме алгоритмов обработки данных.

Задачи, необходимые для достижения поставленной цели:

1. Рассмотреть методы и алгоритмы программирования, подходящие для разработки программы.
2. Реализовать приложение.
3. Протестировать и отладить программу.

1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1 Введение

Составленное техническое задание по дисциплине «Алгоритмические основы обработки данных» является документом к курсовой работе, который отражает все этапы разработки программного продукта, а также процесс проектирования и выявление требований, предъявляемых конечному продукту.

1.1.1 Наименование программы

Название данного приложения “Работа со словарем” будет напрямую связываться с темой курсовой работы “Словарь”. Словарь – это индексируемая структура данных, доступ к элементам которой выполняется только по индексу (ключу). Элемент словаря состоит из ключа и связанного с ключом значения. Пример словаря: русско-английский словарь, элементы которого содержат слово на русском языке (ключ) и перевод слова на английский язык, например, {[список], list}. Для хранения словаря в оперативной памяти использовать динамический массив.

1.1.2 Краткая характеристика области применения программы

Программа предназначена для создания и изменения словаря, индексации по нему. Это приложение будет полезно для всех, кто работает с большим объемом информации.

1.2 Основание для разработки

Основанием для разработки является курсовая работа по дисциплине «Алгоритмические основы обработки данных», предусмотренная учебным

планом направления подготовки 09.03.01 «Информатика и вычислительная техника» профиля «Цифровые комплексы, системы и сети».

1.3 Назначения разработки

Приложение может помочь пользователям, использующим любой сервис чтения в Интернете, для быстрого поиска информации удобным способом.

1.4 Требования, предъявляемые к программе

1.4.1 Требования к функциональным характеристикам программы

В приложении должны быть реализованы следующие операции:

- создание пустого словаря;
- добавление элемента в словарь;
- исключение элемента из словаря;
- поиск элемента словаря по ключу;
- изменение значения элемента;
- вывод словаря в порядке возрастания ключей.

1.4.2 Требования к техническим средствам, используемым при работе программы

Персональный компьютер пользователя должен быть оснащён графическим адаптером, также должна быть установлена ОС Windows (не ниже Windows 7).

1.4.3 Требования к языкам программы и среде разработки программы

Для разработки используется язык программирования C++, в качестве среды разработки выступает Visual Studio. Для разработки интерфейса пользователя задействован механизм Visual Studio.

1.4.4 Требования к информационным структурам на входе и выходе программы

В качестве входных данных программа принимает файл, содержащий словарь вида “Номер телефона - ФИО”

Выходные данные представляют собой файл, содержащий словарь вида “Номер телефона - ФИО”

1.5 Требования к программной документации

1. Пояснительная записка оформляется в соответствии с ЛНА РТУ МИРЭА.

2. Проектная документация, составленная в соответствии с ГОСТ.

В процессе создания приложения вся проделанная работа документируется, должны быть сохранены все детали разработки, а также трудности, с которыми пришлось столкнуться. Всё вышеперечисленное должно быть отражено в пояснительной записке, которая прилагается к работе.

1.6 Этапы разработки

1. Обзор способов организации данных и обоснование выбора структуры данных для эффективного выполнения операций 02.09.2024-22.09.2024.

2. Разработка программы: 22.09.2024-30.11.2024.

3. Разработка программной документации: 01.12.2024-10.12.2024.

4. Оформление пояснительной записки: 11.12.2024-16.12.2024.

5. Защита курсовой работы: 23.12.2024-30.12.2024

2 ОБЗОР СПОСОБОВ ОРГАНИЗАЦИИ ДАННЫХ И ОБОСНОВАНИЕ ВЫБОРА СТРУКТУРЫ ДАННЫХ ДЛЯ ЭФФЕКТИВНОГО ВЫПОЛНЕНИЯ ОПЕРАЦИЙ

Данные могут быть организованы различными способами. Тип структуры данных в программе оказывает большое влияние на ее производительность. Для того чтобы выбрать наиболее простой и эффективный способ организации данных в программе рассмотрим несколько типов структур данных.

2.1 Бинарное дерево поиска

Бинарное дерево поиска - это бинарное дерево (бинарное дерево — это иерархическая структура данных, в которой каждый узел имеет значение, оно же является в данном случае и ключом, и ссылки на левого и правого потомка.), обладающее дополнительными свойствами: значение левого потомка меньше значения родителя, а значение правого потомка больше значения родителя для каждого узла дерева. То есть, данные в бинарном дереве поиска хранятся в отсортированном виде. При каждой операции вставки нового или удаления существующего узла отсортированный порядок дерева сохраняется. При поиске элемента сравнивается искомое значение с корнем. Если искомое больше корня, то поиск продолжается в правом потомке корня, если меньше, то в левом, если равно, то значение найдено и поиск прекращается.

2.2 Словарь

Словарь (ассоциативный массив) — это индексируемая структура данных, доступ к элементам которой выполняется только по индексу (ключу). Элемент словаря состоит из ключа и связанного с ключом значения. Словарь поддерживает добавление пар ключ-значение, получение значения по ключу, удаление пары ключ-значение, получение размера словаря, проверка словаря на

пустоту. Имеет быстрый доступ к данным по ключу, а также гибкость хранения данных разного типа. Но занимает больше памяти, чем простые списки, и не гарантирует упорядоченность элементов.

2.3 Хэш-таблица

Хэш-таблица - структура данных, реализующая интерфейс ассоциативного массива, а именно, она позволяет хранить пары (ключ, значение) и выполнять три операции: операцию добавления новой пары, операцию удаления и операцию поиска пары по ключу. Она реализуется с использованием хэш-функции, которая преобразует ключ в индекс массива (или другой структуры), где хранится значение. Позволяет быстро искать, добавлять и удалять данные, так как операции выполняются за $O(1)$. Но при большом количестве данных могут возникать коллизии (ситуация, когда разные ключи дают одинаковый хэш). Также требует больше памяти

2.4 Очередь

Очередь – динамическая структура данных, работающая по принципу FIFO – first in, first out (первый вошел, первый вышел). Элементы добавляются в конец очереди, и оттуда же извлекаются. Позволяет добавить элемент в конец очереди, удалить элемент в конце очереди, получить доступ к концу и началу очереди, получить размер массива. Очередь часто используется в алгоритмах для организации порядка выполнения задач. Полезна в алгоритмах, где требуется упорядоченная обработка данных. Неэффективна для поиска и случайного доступа, поскольку доступ к элементам идет строго в порядке добавления.

2.5 Массив

Массив — одна из самых простых и часто применяемых структур данных. Другие структуры данных, к примеру, стеки и очереди, являются производными

от массивов. Сами массивы используются для обработки большого количества однотипных данных.

Элементы в массиве имеют свой индекс — номер элемента, по которому можно производить поиск. Существует два типа массивов: одномерные и многомерные. Первые представляют собой простейшие линейные структуры, которые называются вложенными и включают другие массивы. Чаще всего в программировании используются одномерные и двумерные массивы.

Также существуют динамические массивы, для выделения работы с ними используются отдельные формы операторов `new` и `delete`: `new[]` и `delete[]`. Динамическое выделение массива позволяет устанавливать его длину во время выполнения программы.

2.6 Выбор структуры данных

В ходе использования программы пользователь вводит в консоль текст, либо вводит название текстовых файлов, поэтому нужно воспользоваться типом данных `string` (строковый).

Чтобы реализовать хранение данных в файле, необходимо использовать файловый поток `fstream` для чтения, записи и хранения файла.

Для хранения паспортных данных и ФИО, напомним структуру данных, используя типы данных `int` и `string`. Так будет удобнее хранить и выводить данные, сортировать и записывать.

Для хранения структур будет использоваться структура `map` (словарь). Это позволит удобно хранить данные, а также обеспечит простые операции поиска по ключу и сортировке данных.

3 ОПИСАНИЕ ПРОГРАММЫ

3.1 Общие сведения

В ходе выполнения курсовой работы была создана программа с консольным интуитивно понятным интерфейсом для работы с словарем для операционной системы Windows. В ней выполняются все условия, обозначенные в техническом задании, и содержатся все необходимые компоненты, инструменты для корректной работы

3.1.1 Наименование программы

Название программы: «Работа с словарем» или на английском языке «Working with dictionary». Оно отражает предназначение и главную функцию созданного приложения.

3.1.2 Программное обеспечение, необходимое для функционирования программы

Для корректного функционирования данного программного продукта необходимо, чтобы на персональном компьютере или ноутбуке пользователя была установлена ОС от компании Microsoft, а именно Windows (Windows 10/11). Также требуется наличие графического адаптера, чтобы устройство могло справляться с обработкой отображения консоли приложения. Другие требования к устройству пользователя не предусмотрены.

3.1.2 Язык программирования, на котором написана программа

Для написания программы был выбран язык программирования C++, за его доступность, понятность и высокую производительность

3.2 Функциональное назначение программы (классы решаемых задач и функциональные ограничения на применения)

Данная программа написана для упрощения понимания словарей и для удобной работы с ними. Функциональные цели приложения включают операции с текстовым файлом для работы с словарем: создание пустого словаря для ввода в консоль, вывод элементов словаря в порядке возрастания ключей, поиск элемента словаря по ключу. Также предусмотрено чтение словаря из текстового файла и запись словаря в текстовый файл.

3.3 Описание логической структуры программы

В программе используется процедурный подход реализации алгоритмов для упрощения данного приложения. Для хранения данных о пользователях используется структура с единственным методом о выводе информации. Исходный код программы представлен в Приложении А.

3.3.1 Алгоритмы, используемые в программе

Для написания программы необходимо подключить библиотеку «fstream» для работы с файлами, работа со строками осуществляется средствами библиотеки «string», для вызова консоли необходимо подключить библиотеку «iostream». Библиотека «map» для использования словаря.

Основными алгоритмами для работы данного приложения являются алгоритм создания словаря и добавление в него элементов, алгоритм поиска элемента по ключу, алгоритм удаления элемента по ключу, изменение значения по ключу, алгоритм записи информации в файл, алгоритм вывода информации на консоль, алгоритм чтения данных из файла и алгоритм запуска программы.

3.3.1.1 Алгоритм запуска программы.

В начале программы пользователю предлагается текстовое меню с 9 пунктами на выбор, который осуществляется с помощью конструкции switch-case:

1. Создание пустого словаря.
2. Добавление элемента в словарь
3. Исключение элемента из словаря
4. Нахождение элемента в словаре по ключу
5. Изменение значения элемента
6. Вывод словаря в порядке возрастания ключей
7. Чтение данных словаря из текстового файла
8. Запись словаря в файл
9. Запись словаря в файл и завершение работы

В первом случае создается словарь (или очищается старый) в программе.

Во втором случае у пользователя запрашиваются данные и ключ, по которому данные сохраняются в словаре.

В третьем случае из словаря удаляется элемент, ключ которого пользователь ввел.

В четвертом случае находятся и выводятся данные о элементе под введенным ключом.

В пятом случае данные, ключ которых введен, меняются на новые, введенные с клавиатуры.

В шестом случае словарь выводится в консоль в порядке возрастания ключей.

В седьмом данные из текстового файла записываются в словарь.

В восьмом случае данные из словаря записываются в текстовый файл(адрес которого вводится с клавиатуры).

В девятом случае данные из словаря записываются в текстовый файл и программа завершается.

Блок-схема алгоритма представлена на рисунках 3.1 и 3.2

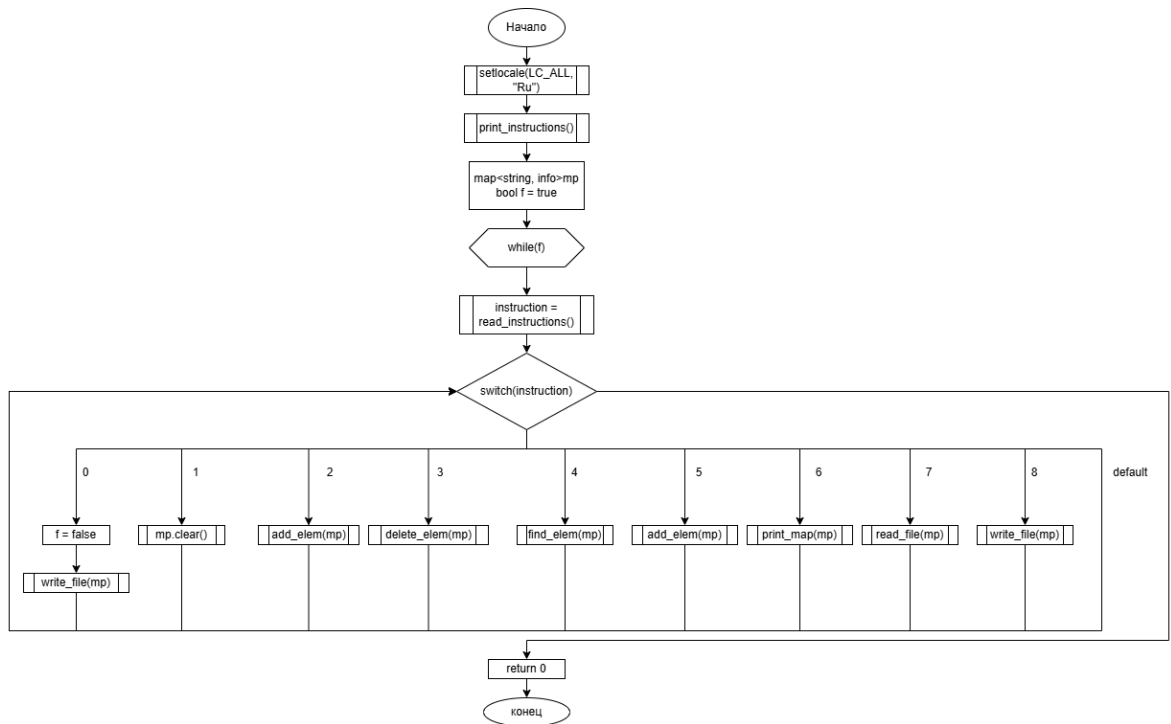


Рисунок 3.1 – Алгоритм запуска программы main()

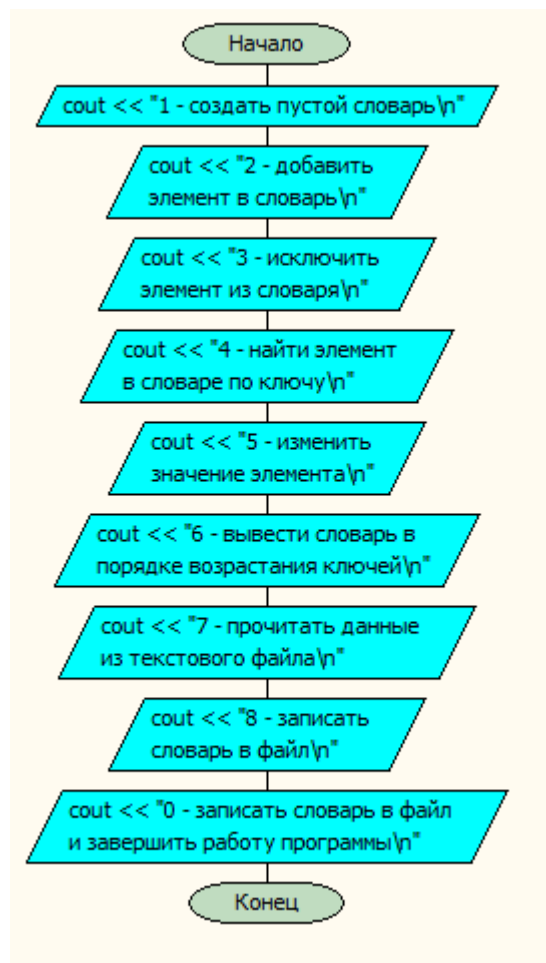


Рисунок 3.2 – Алгоритм вывода инструкции (print_instruction)

3.3.1.2 Алгоритм создания пустого словаря

Этот алгоритм очищает память, которую занимал словарь при работе программы. Все данные из предыдущего словаря будут соответственно удалены. Для этого используем метод `.clear()` ,библиотеки `map`.

Алгоритм представлен на рисунке 3.3

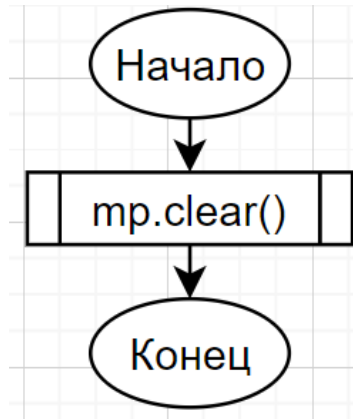


Рисунок 3.3 - Алгоритм создания словаря

3.3.1.3 Алгоритм добавления элемента в словарь и изменения элемента в словаре

Этот алгоритм используется для добавления или изменения информации о человеке в словаре `mp`. Для этого программа запрашивает ключ, а затем данные, которые надо записать под этим ключом. Обе операции выполняются по одному и тому же алгоритму.

Блок-схема алгоритма представлена на рисунке 3.4

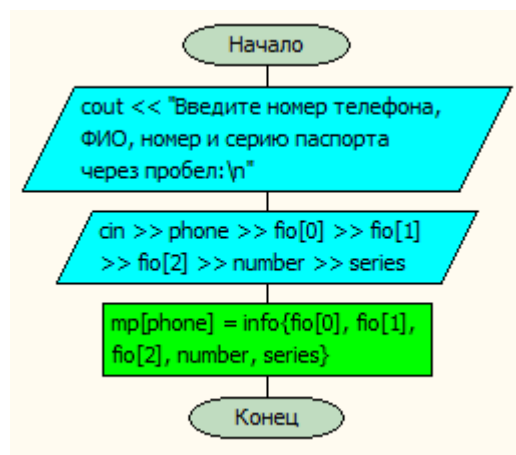


Рисунок 3.4 – Алгоритм добавления пары ключ-значение в словарь

3.3.1.4 Алгоритм исключения элемента из словаря

Алгоритм исключения из словаря удаляет данные о человеке по ключу, который введет пользователь. В случае, если пары “введенный ключ-значение” нет, то алгоритм ничего не поменяет в словаре

Блок-схема алгоритма представлена на рисунке 3.5

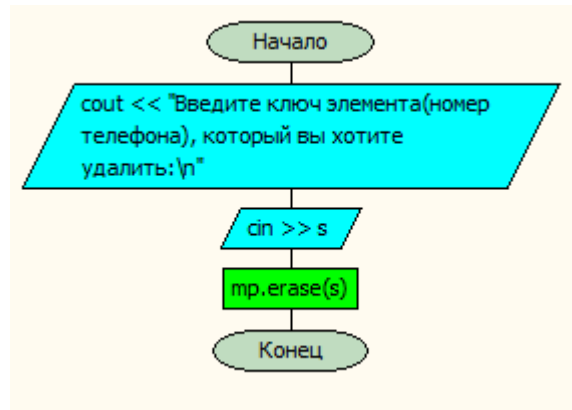


Рисунок 3.5 – Алгоритм исключения элемента из словаря

3.3.1.5 Алгоритм поиска элемента по ключу

Алгоритм поиска элемента по ключу считывает ключ и выводит информацию, которая хранится под этим ключом. Если же пара ключ-значение не была найдена, программа уведомляет об этом пользователя

Блок-схема алгоритма представлена на рисунке 3.6

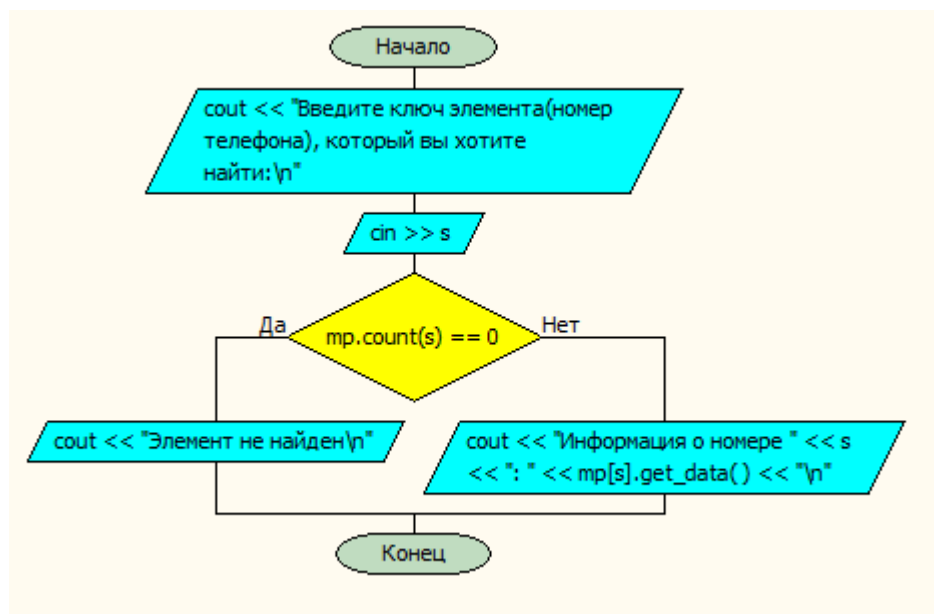


Рисунок 3.6 – Алгоритм поиска элемента по ключу

3.3.1.6 Алгоритм вывода словаря в порядке возрастания значения ключей

Этот алгоритм выводит все пары ключ-значение из словаря. Так как в словаре данные автоматически упорядочены, сортировать ничего не надо и программе просто остается вывести все данные с помощью цикла `for`.

Блок-схема алгоритма представлена на рисунке 3.7

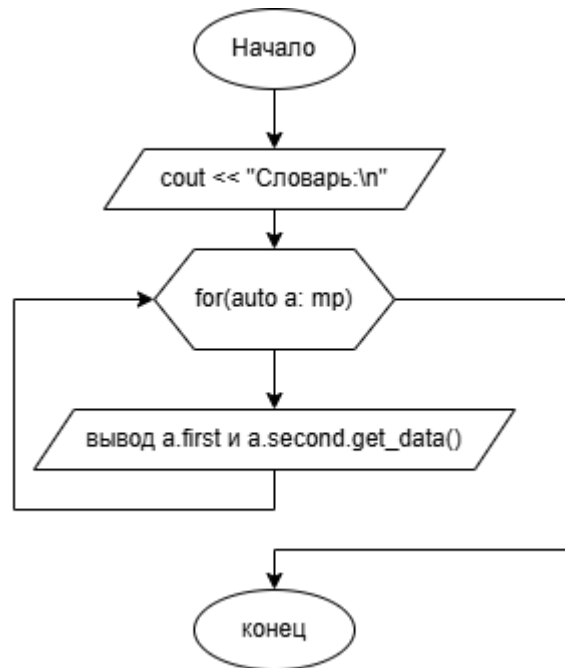


Рисунок 3.7 – Алгоритм вывода словаря в порядке возрастания ключей

3.3.1.7 Алгоритм чтения данных из текстового файла.

Алгоритм запрашивает название файла, данные которого будут считываться. Если файл не найден или его не удалось открыть, программа сообщает об этом пользователю. После чего алгоритм построчно считывает данные из текстового файла и записывает в словарь. Для этого используется библиотека `fstream` и цикл `while`.

Блок-схема алгоритма представлена на рисунке 3.8

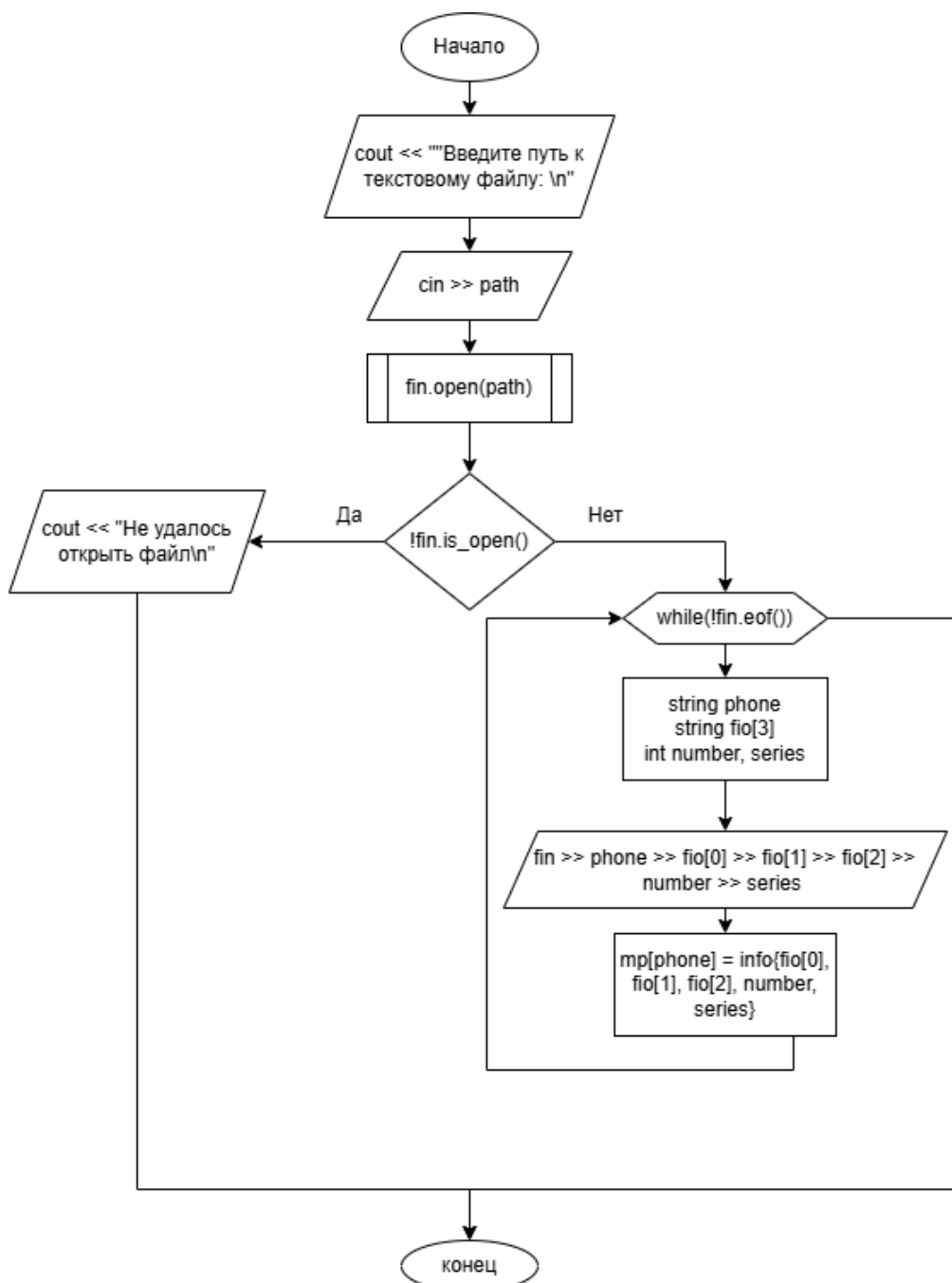


Рисунок 3.8 – Алгоритм чтения данных из текстового файла

3.3.1.8 Алгоритм записи данных в текстовый файл

Алгоритм записывает все данные из словаря в текстовый файл. Для этого используется цикл while и библиотека fstream.

Блок-схема алгоритма представлена на рисунке 3.9

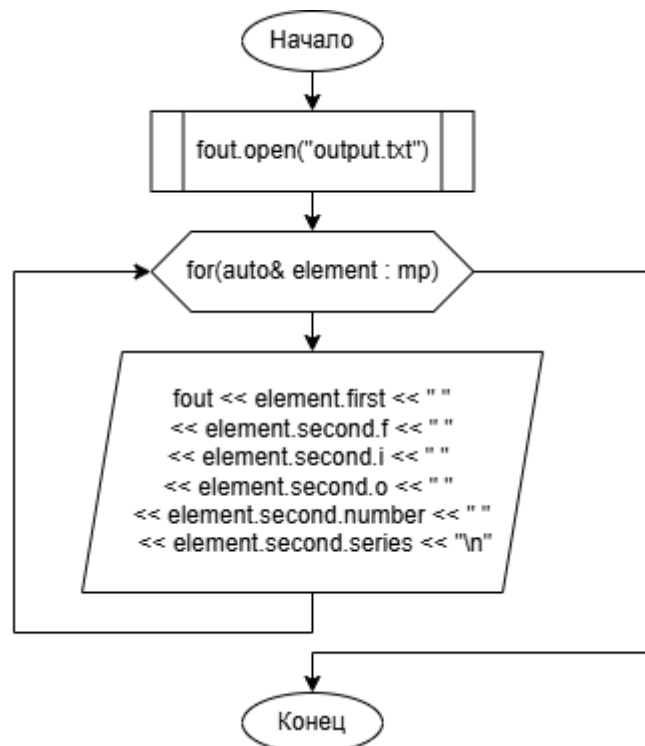


Рисунок 3.9 – Блок-схема алгоритма записи данных в текстовый файл

3.3.2 Структура программы с описанием функций составных частей и связей между ними

Для удобного взаимодействия с приложением было создано несколько функций, которые вызываются при разных выборе пунктов меню в главной функции запуска программы `int main()`:

- `print_instructions()` – функция вывода инструкций на экран
- `add_elem(&mp)` – функция добавления или изменения элемента в словаре
- `delete_elem(&mp)` – функция удаления элемента из словаря
- `find_elem(&mp)` – функция поиска элемента словаря
- `print_map(&mp)` – функция вывода словаря в консоль
- `read_instructions()` – функция ввода вариантов из меню
- `read_file(&mp)` – функция считывания данных из текстового файла
- `write_file(&mp)` – функция записи данных из словаря в файл

3.4 Технические средства, которые используются при работе программы

Для запуска программы пользователю необходимо открыть предоставляемый пользователю файл «.exe». Как было отмечено ранее, для корректной работы продукта необходимо наличие графического адаптера и операционной системы Windows.

3.5 Вызов программы

Исходный текстовый файл (Рисунок 3.10)

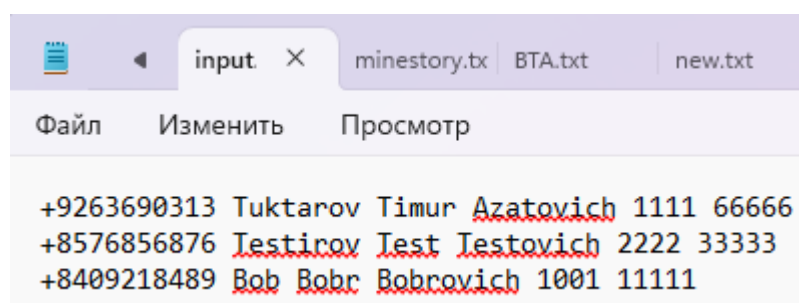


Рисунок 3.10 – тестовый файл

При запуске программного продукта появляется главное окно приложения (Рисунок 3.11), содержащее текстовое меню со всеми указанными в техническом задании операциями над словарем. При выборе определенного пункта меню производятся необходимые операции.

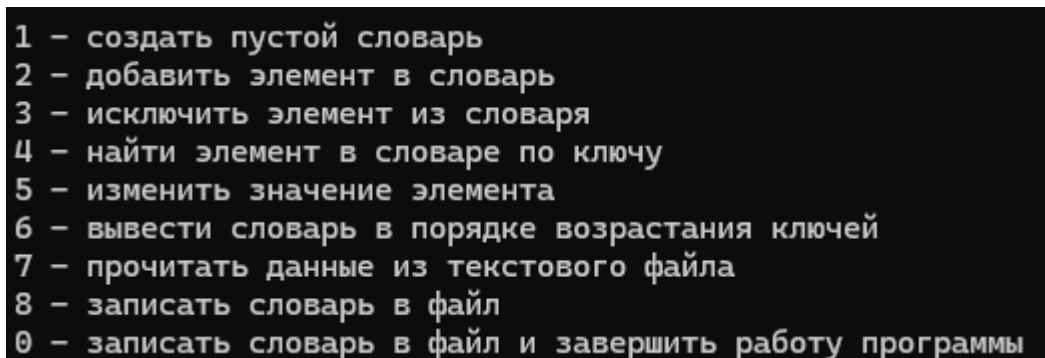


Рисунок 3.11 – Главное окно приложения

Далее будет показана работа всех выбранных пунктов меню на рисунках 3.12, 3.13, 3.14, 3.15, 3.16, 3.17, 3.18, 3.19, 3.20, 3.21, 3.22

Начало работы программы – создание пустого словаря (Рисунок 3.11). Стоит учесть, что для вывода и проверки команд используется алгоритм вывода словаря.

```
1
6
Словарь:
|
```

Рисунок 3.11 – Создание пустого словаря

Пример добавления данных о двух людях в словарь и их вывод (Рисунки 3.12 и 3.13)

```
2
Введите номер телефона, ФИО, номер и серию паспорта через пробел:
+9263790313 Tuktarov Timur Azatovich 1111 66666
2
Введите номер телефона, ФИО, номер и серию паспорта через пробел:
+8576856876 Testirov Test Testovich 2222 33333
|
```

Рисунок 3.12 – Добавление двух элементов в словарь

```
Словарь:
+8576856876 : Testirov Test Testovich | 2222 33333
+9263790313 : Tuktarov Timur Azatovich | 1111 66666
```

Рисунок 3.13 – Содержание словаря после добавления элементов

Далее значение из словаря удаляется по ключу, а затем ищутся значения по ключу (Рисунки 3.14 – 3.16)

```
3
Введите ключ элемента(номер телефона), который вы хотите удалить:
+9263790313
```

Рисунок 3.14 – Удаление элементов из словаря

```
Словарь:
+8576856876 : Testirov Test Testovich | 2222 33333
```

Рисунок 3.15 – Содержание словаря после удаления элемента

```
Введите ключ элемента(номер телефона), который вы хотите найти:
+8576856876
Информация о номере +8576856876: Testirov Test Testovich | 2222 33333
4
Введите ключ элемента(номер телефона), который вы хотите найти:
1
Элемент не найден
```

Рисунок 3.16 Пример поиска элемента по корректному и некорректному ключу

Пример изменения значения элемента по ключу приведен на рисунке 3.17, а результат изменения на рисунке 3.18

```
5
Введите номер телефона, ФИО, номер и серию паспорта через пробел:
+8576856876 Changed Cnahge Ch 8888 222222
```

Рисунок 3.17 – Изменение значения элемента

```
Словарь:
+8576856876 : Changed Cnahge Ch | 8888 222222
```

Рисунок 3.18 – Содержание словаря после изменения

После в словарь записываются значения из файла, который по казан на рисунке 3.10 в словарь (Рисунок 3.19) и выводим его значения (Рисунок 3.20)

```
7
Введите путь к текстовому файлу:
input.txt
```

Рисунок 3.19 – Запись значений в словарь из файла

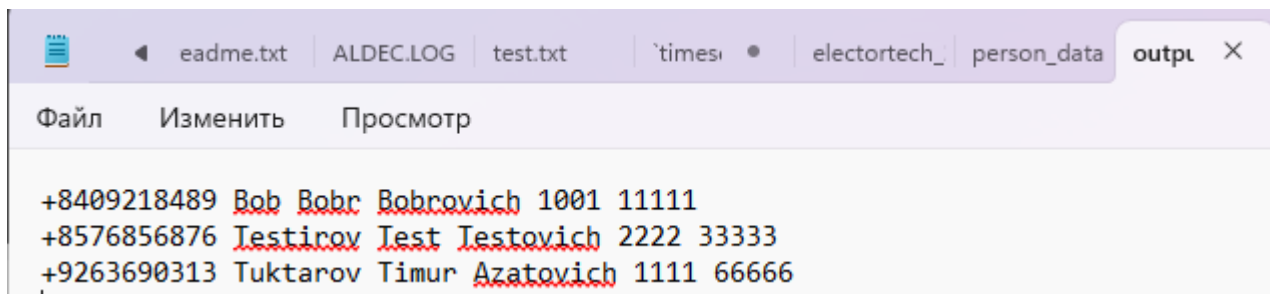
```
6
Словарь:
+8409218489 : Bob Bobr Bobrovich | 1001 11111
+8576856876 : Testirov Test Testovich | 2222 33333
+9263690313 : Tuktarov Timur Azatovich | 1111 66666
```

Рисунок 3.20 – Вывод словаря в порядке возрастания ключей

На рисунок 3.22 Показан результат завершения программы – запись словаря в текстовый файл с именем output.txt

```
0
D:\Files\Algo_dann_2_curs\Progs\PR_2\x64\Debug\KURSOVAYA.exe (процесс 3996) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 3.21 – Вывод значений словаря в текстовый файл



```
Файл  Изменить  Просмотр

+8409218489 Bob Bobr Bobrovich 1001 11111
+8576856876 Testirov Test Testovich 2222 33333
+9263690313 Tuktarov Timur Azatovich 1111 66666
```

Рисунок 3.22 – Текстовый файл output.txt

3.6 Входные данные (организация и предварительная подготовка входных данных)

В качестве входных данных программа принимает текст, который вводит пользователь. Также программа может принять данные из текстовых файлов, расположение которых указывает пользователь в консоли.

3.7 Выходные данные

Выходные данные – Пары ключ-значения в упорядоченном виде.

ЗАКЛЮЧЕНИЕ

На протяжении всего процесса проектирования и создания программного продукта были получены практические навыки в области структур данных, которые используются для хранения и обработки различных типов данных. Также были освоены алгоритмы работы с файлами в C++.

Успешно выполнены поставленные задачи: создание программы, позволяющей осуществлять операции над словарем, отладка и проверка работоспособности конечной программы, применение алгоритмов для работы с словарем. Среди упомянутых операций: создание пустого словаря, вывод элементов в порядке возрастания ключей и поиск элемента словаря по ключу, сохранение данных в файл и считывание их из текстового файла.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лозовский В.В. Алгоритмические основы обработки данных: учебное пособие / Лозовский В.В., Платонова О.В., Штрекер Е.Н. — М.: МИРЭА – Российский технологический университет, 2022. — 337 с.
2. Платонова О.В. Алгоритмические основы обработки данных: методические указания / Платонова О.В., Асадова Ю.С., Расулов М.М. — М.: МИРЭА – Российский технологический университет, 2022. — 73 с.
3. Белик А.Г. Алгоритмы и структуры данных: учебное пособие / А.Г. Белик, В.Н. Цыганенко. — Омск: ОмГТУ, 2022. — 104 с. — ISBN 978-5-8149-3498-7. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/343688> (дата обращения: 03.09.2024)
4. Павлов Л.А. Структуры и алгоритмы обработки данных / Л.А. Павлов, Н.В. Первова. — 2-е изд., стер. — Санкт-Петербург: Лань, 2022. — 256 с. — ISBN 978-5-507-44105-1. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/207563> (дата обращения: 03.09.2024)
5. Пантелеев Е.Р. Алгоритмы и структуры данных: учебное пособие / Е.Р. Пантелеев, А.Л. Алыкова. — Иваново: ИГЭУ, 2018. — 142 с. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/154576> (дата обращения: 03.09.2024)

ПРИЛОЖЕНИЕ А

Код программы приведен в листинге 4.1

Листинг 1 - код программы для работы со словарями.

```
// KURSOVAYA.cpp : Этот файл содержит функцию "main". Здесь начинается и
// заканчивается выполнение программы.
//

#include <iostream>
#include <fstream>
#include <string>
#include <map>
using namespace std;
struct info {
    string f, i, o;
    int number, series;
    bool operator<(const info& other) const
    {
        string s1 = "", s2 = "";
        s1 += this->f + this->i + this->o;
        s2 += other.f + other.i + other.o;
        return (s1 < s2);
    }

    string get_data() {
        return f + " " + i + " " + o + " | " + to_string(number) + " " +
to_string(series);
    }
};

ifstream fin;
ofstream fout;

void print_instructions() {
    cout << "1 - создать пустой словарь\n";
    cout << "2 - добавить элемент в словарь\n";
    cout << "3 - исключить элемент из словаря\n";
    cout << "4 - найти элемент в словаре по ключу\n";
    cout << "5 - изменить значение элемента\n";
    cout << "6 - вывести словарь в порядке возрастания ключей\n";
    cout << "7 - прочитать данные из текстового файла\n";
    cout << "8 - записать словарь в файл\n";
    cout << "0 - записать словарь в файл и завершить работу программы\n";
}

void add_elem(map<string, info>& mp) {
    cout << "Введите номер телефона, ФИО, номер и серию паспорта через
пробел:\n";
    string phone;
    string fio[3];
    int number, series;
    cin >> phone >> fio[0] >> fio[1] >> fio[2] >> number >> series;
    mp[phone] = info{ fio[0], fio[1], fio[2], number, series };
}

void delete_elem(map<string, info>& mp) {
    cout << "Введите ключ элемента(номер телефона), который вы хотите
удалить:\n";
    string s;
    cin >> s;
```

Продолжение листинга 1

```
        mp.erase(s);
    }

    void find_elem(map<string, info>& mp) {
        cout << "Введите ключ элемента(номер телефона), который вы хотите
        найти:\n";
        string s;
        cin >> s;
        if (mp.count(s) == 0) cout << "Элемент не найден\n";
        else cout << "Информация о номере " << s << ": " << mp[s].get_data() <<
        "\n";
    }

    void print_map(map<string, info>& mp) {
        cout << "Словарь:\n";
        for (auto a : mp) {
            cout << a.first << " : " << a.second.get_data() << "\n";
        }
    }

    int read_instructions() {
        int n;
        while (1) {
            cin >> n;
            if (n >= 0 && n <= 8) return n;
        }
    }

    void read_file(map<string, info>& mp) {
        cout << "Введите путь к текстовому файлу:\n";
        string path;
        cin >> path;
        fin.open(path);
        if (!fin.is_open()) {
            cout << "Не удалось открыть файл\n";
            return;
        }
        while (!fin.eof()) {
            string phone;
            string fio[3];
            int number, series;
            fin >> phone >> fio[0] >> fio[1] >> fio[2] >> number >> series;
            mp[phone] = info{ fio[0], fio[1], fio[2], number, series };
        }
        fin.close();
    }

    void write_file(map<string, info>& mp) {
        fout.open("output.txt");
        for (const auto& element : mp) {
            fout << element.first << " " << element.second.f << " " <<
            element.second.i << " " << element.second.o << " " << element.second.number <<
            " " << element.second.series << "\n";
        }
        fout.close();
    }

    int main()
    {
        setlocale(LC_ALL, "Ru");
        map<string, info> mp;
        print_instructions();
        bool f = true;
```

Продолжение листинга 1

```
while (f) {
    int instruction = read_instructions();
    //cout << instruction << "\n";
    switch (instruction)
    {
    case 0:
        f = false;
        write_file(mp);
        break;
    case 1:
        mp.clear();
        break;
    case 2:
        add_elem(mp);
        break;
    case 3:
        delete_elem(mp);
        break;
    case 4:
        find_elem(mp);
        break;
    case 5:
        add_elem(mp);
        break;
    case 6:
        print_map(mp);
        break;
    case 7:
        read_file(mp);
        break;
    case 8:
        write_file(mp);
        break;
    default:
        break;
    }
}

return 0;
}
```