СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Телефонный справочник - одно из наиболее распространенных и полезных приложений, которое предоставляет доступ к контактным данным различных людей и организаций. Способность быстро и удобно находить контактные данные является неотъемлемой частью современной информационной среды. В связи с этим разработка программного приложения, позволяющего хранить и управлять данными в телефонном справочнике, играет важную роль.

Цель курсовой работы: получение практических навыков в области 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 «Телефонный справочник» будет напрямую связываться с темой курсовой работы: «Словарь». Данное название отражает предназначение будущего приложения. Английский вариант названия: «Telephone book». Краткое наименование: «ТВ».

1.1.2 Краткая характеристика области применения программы

Программа предназначена для редактирования и индексации по словарю. Это приложение будет полезно для всех, кому необходимо организовать упорядоченность имеющихся исходных данных при работе 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 Назначение разработки

Приложение может помочь государственным организациям, быстрее анализировать, внедрять и контролировать изменения данных.

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.

1.4.4 Требования к информационным структурам на входе и выходе программы

В качестве входных данных программа принимает файл, содержащий номер телефона, ФИО, паспортные данные (поле «номер телефона» является ключом, ФИО и паспортные данные являются значением элемента словаря).

Выходные данные представляют собой файл, содержащий ключ (номер телефона) и связанное с ключом значение (ФИО и паспортные данные).

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

4. Обзор способов организации данных и обоснование выбора структуры данных для эффективного выполнения операций 02.09.2023-22.09.2023.
5. Разработка программы: 22.09.2023-30.11.2023.
6. Разработка программной документации: 01.12.2023-10.12.2023.
7. Оформление пояснительной записки: 11.12.2023-16.12.2023.
8. Защита курсовой работы: 10.12.2023-15.12.2023.

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

С помощью различных типов структур данных можно организовать информацию, это повлияет на производительность программы. Рассмотрим несколько типов, чтобы выбрать наиболее простой и эффективный способ организации данных.

2.1 Массив

Массив является примитивной структурой данных. Представляет собой именованную типизированную область памяти, в которой ячейки идут последовательно друг за другом. Массив может быть статическим или динамическим. На основе массива создаются производные от него типы данных, например, стеки и очереди.

2.2 Стек

Стек – абстрактный тип данных, представляющий собой список элементов, которые организованны по принципу LIFO (last in – first out, «последним пришел – первым вышел»). Первым из стека удаляется элемент, который был помещен туда последним.

2.3 Очередь

Очередь – этот вид структуры такой же, как и стек, но работает по принципу FIFO (first in – first out, «первым пришел – первым ушел»). Первым из очереди удаляется элемент, который был помещен туда первым. Из очереди всегда выводится элемент, который находится в ее голове.

2.4 Карта (Map)

Карта или словарь – это ассоциативный массив, в котором данные хранятся в паре «ключ/значение». Словари могут быть с повторяющимися парами, либо с уникальными. Ключ должен быть уникальным.

2.5 Выбор структуры данных

В ходе своей работы, программа получает на вход текстовый файл, содержащий пары по принципу ключ-значение. Для считывания данных из текстового файла необходимо воспользоваться файловым хранилищем fstream.

Для хранения считанных данных в памяти программы следует использовать словарь. Использование словаря позволит установить связь между элементами считанных строк.

Благодаря преимуществу словарей, можно будет быстро и эффективно обращаться к элементам по уникальному ключу-идентификатору.

3 ОПИСАНИЕ ПРОГРАММЫ

3.1 Общие сведения

В ходе выполнения курсовой работы была создана программа с консольным интуитивно понятным интерфейсом для работы со 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: «Телефонный справочник» или на английском языке «Telephone book». 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). Также требуется наличие графического адаптера, чтобы устройство могло справиться с обработкой отображения консоли приложения. Другие требования к устройству пользователя не предусмотрены.

3.1.3 Язык программирования, на котором написана программа

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

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

Данная программа написана для понимания структуры данных словарь и для удобной работы с ним. Функциональные цели приложения включают операции с текстовым файлом для работы со словарем: создание пустого словаря, добавление элемента в словарь, исключение элемента из словаря, поиск элемента словаря по ключу, изменение значения элемента и вывод словаря в порядке возрастания ключей. Также предусмотрена проверка на правильность ввода паспортных данных, ФИО, и телефонного номера.

Разработанное приложение имеет некоторые ограничения. Пользователь должен заранее указать правильный путь к файлу. Кроме того, после создания пустого словаря, программу следует перезапустить.

3.3 Описание логической структуры программы

В программе используется процедурный подход реализации алгоритмов для упрощения данного приложения. Сама программа не содержит в себе огромных методов или функций, поэтому ООП использовать нет необходимости. Исходный код 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», для создания словаря необходима библиотека «map». Библиотека «algorithm» для функций сортировки.

Основными алгоритмами для работы данного приложения являются алгоритм создания словаря и добавления в него элементов, алгоритм поиска элемента по ключу, алгоритм исключения элемента из словаря, алгоритм

изменения элемента, алгоритм вывода словаря на консоль и алгоритм запуска программы.

3.3.1.1 Алгоритм запуска программы

В начале программы пользователю предлагается текстовое меню с 7 пунктами на выбор, который осуществляется с помощью конструкции elseif:

1. Добавление элемента в словарь.
2. Удаление элемента.
3. Поиск значения элемента по ключу.
4. Изменение значения элемента.
5. Вывод словаря.
6. Создание пустого словаря.

В первом случае пользователю требуется ввести телефонный номер, ФИО и паспортные данные для того, чтобы добавить новый элемент в словарь (функция add). При вводе данных выполняется проверка на их корректность с помощью функций CheckName, CheckPhone и CheckPassport. Далее данные будут записаны в файл с помощью функции writeToFile.

Во втором случае удаление элемента из словаря будет выполняться функцией remove, операция сохраниться в файл после вызова функции writeToFile.

В третьем случае будет реализован алгоритм поиска значения элемента по ключу. Для этого пользователю необходимо ввести ключ (телефонный номер). Далее функция find выведет всю информацию об этом элементе.

В четвертом случае происходит обновление информации об элементе по ключу, для этого нужно ввести новые данные (ФИО и паспортные данные). Далее обновленная информация записывается в файл.

В пятом случае выводится словарь с помощью функции print.

И последний случай, шестой, чтобы создать пустой словарь.

Блок-схема алгоритма представлена на рисунке 3.1

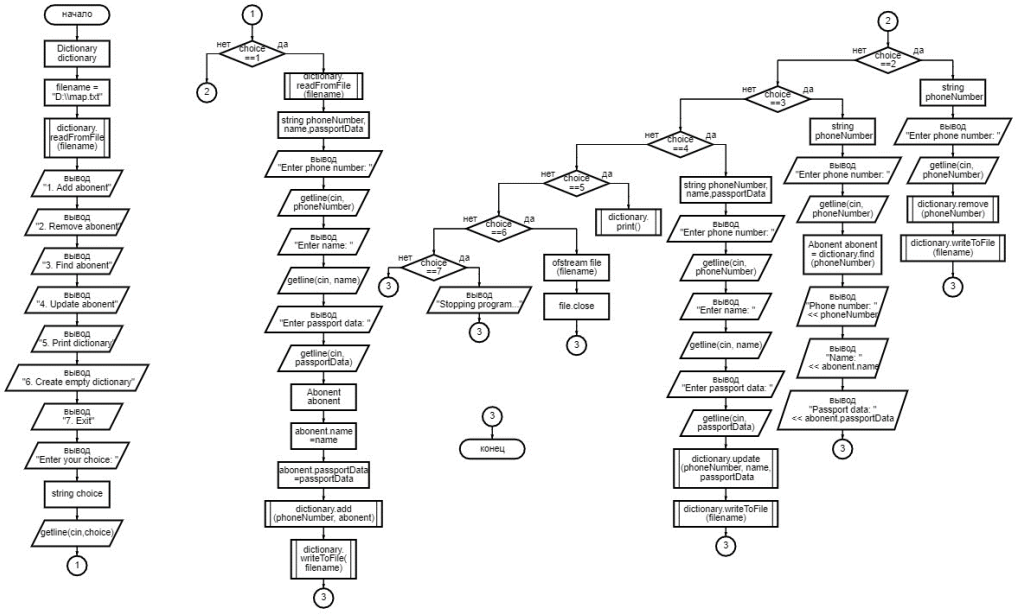

Рисунок 3.1 – Алгоритм запуска программы main()

3.3.1.2 Алгоритм добавления элемента в словарь

Этот алгоритм добавляет новый элемент в словарь, данные которого пользователь должен ввести.

Блок-схема алгоритма добавления элемента на рисунке 3.2

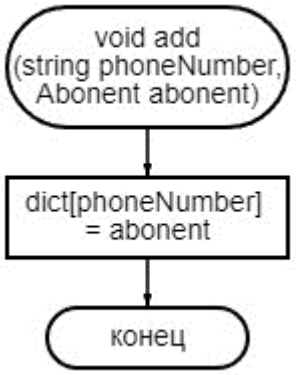

Рисунок 3.2 – Алгоритм добавления элемента в словарь

3.3.1.3 Алгоритм удаления элемента

Этот алгоритм используется для удаления элемента из словаря с помощью встроенной функции erase, для этого необходимо ввести телефонный номер (ключ).

Блок-схема алгоритма удаления элемента на рисунке 3.3

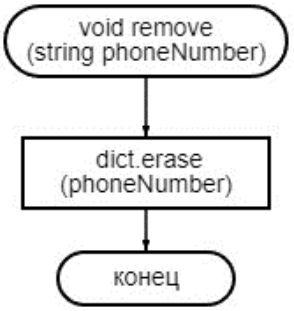

Рисунок 3.3 – Алгоритм удаления элемента

3.3.1.4 Алгоритм поиска элемента по ключу

Алгоритм поиска элемента будет настроен следующим образом. На вход будет подан телефонный номер (ключ), по которому будет происходить поиск, далее произойдет поиск данного ключа в словаре с помощью встроенной функции find далее будут возвращены значения, принадлежащие этому ключу (ФИО и паспортные данные).

Блок-схема алгоритма поиска по ключу на рисунке 3.4

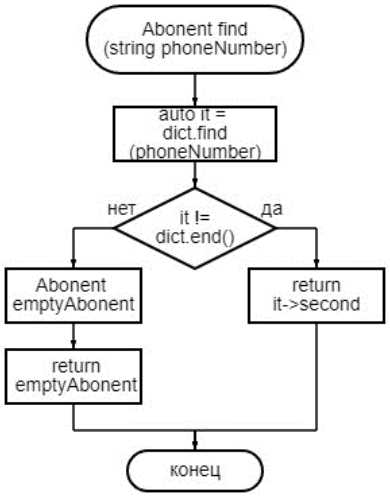

Рисунок 3.4 – Алгоритм поиска элемента по ключу

3.3.1.5 Алгоритм изменения значения

Этот алгоритм необходим для изменения значения словаря по его ключу. Для этого пользователь должен ввести ключ элемента, который он хочет

изменить, а также новые данные: ФИО и паспортные данные. С помощью встроенной функции find будет найден необходимый элемент в словаре и с помощью оператора косвенного обращения изменены его значения, на те которые были введены ранее.

Блок-схема алгоритма изменения значений на рисунке 3.5

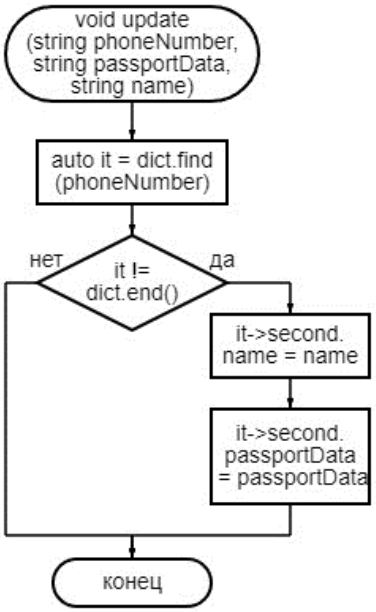

Рисунок 3.5 – Алгоритм изменения значений элемента

3.3.1.6 Алгоритм вывода словаря

Алгоритм позволяет вывести все данные словаря в консоль. Для этого с помощью цикла for по всем элементам словаря выводиться поочередно телефонный номер, ФИО и паспортные данные.

Блок-схема алгоритма вывода словаря на рисунке 3.6

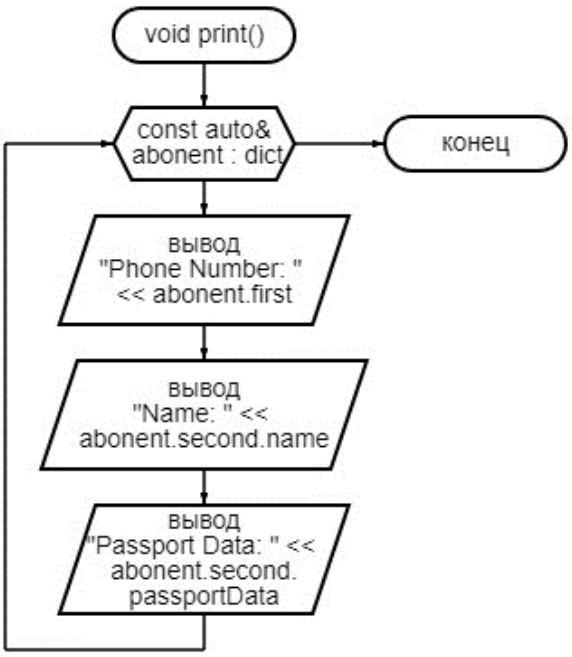

Рисунок 3.6 – Алгоритм вывода

3.3.1.7 Алгоритм создания пустого словаря

Этот алгоритм используется для создания словаря, в котором нет элементов. Для этого файл открывается, с помощью библиотеки fstream, в режиме записи, таким образом словарь, который ранее был в файле удаляется, далее файл закрывается для сохранения операции.

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 выборах пунктов меню в главной функции запуска программы int main():

- add(phoneNumber, abonent) – функция добавления элемента в словарь, данные вводятся с клавиатуры;
- remove(phoneNumber) – функция удаления элемента по его ключу, ключ вводится с клавиатуры;
- find(phoneNumber) – функция поиска элемента по ключу, который

вводится с клавиатуры;

- update(phoneNumber, name, passportData) – функция изменения значений элемента по его ключу, данные вводятся с клавиатуры;
- print() – функция вывода словаря в консоль.

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 3.7):

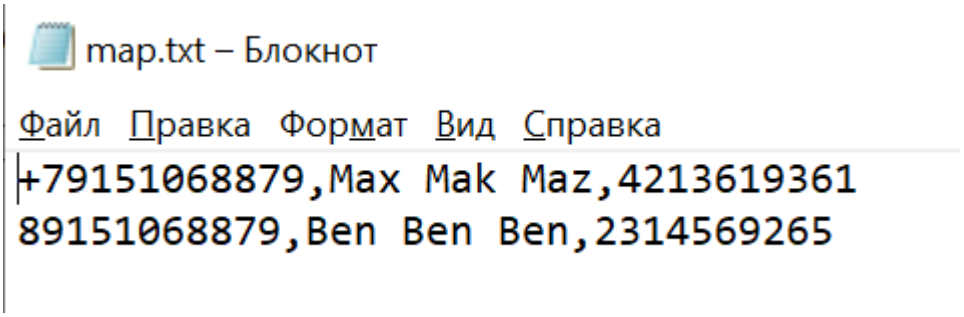

Рисунок 3.7 – Исходный текстовый файл

При запуске программного продукта появляется главное окно приложения (Рисунок 3.8), содержащие текстовое меню со всеми указанными в техническом задании операциями над словарем. При выборе определенного пункта меню производятся необходимые операции.

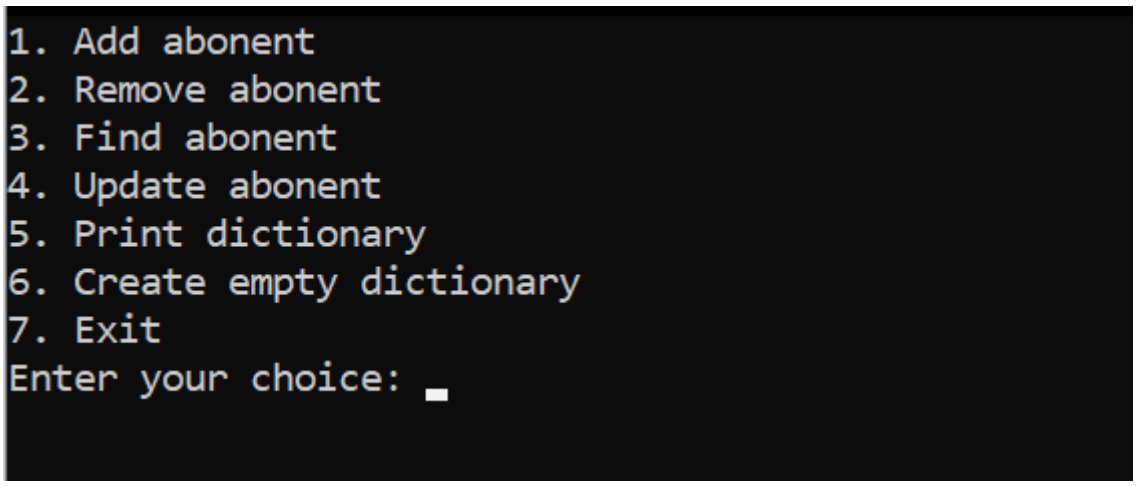

Рисунок 3.8 – Главное окно приложения

Далее будет показана работа всех выбранных пунктов меню на рисунках 3.9, 3.11, 3.13, 3.14, 3.16, а на рисунках 3.10, 3.12, 3.15, 3.17 представлены примеры текстовых файлов, куда записывалась или откуда считывалась информация словаря.

При вводе цифры 1 пользователю нужно ввести телефонный номер, ФИО и паспортные данные (Рисунок 3.9), далее новый элемент добавится в словарь (Рисунок 3.10).

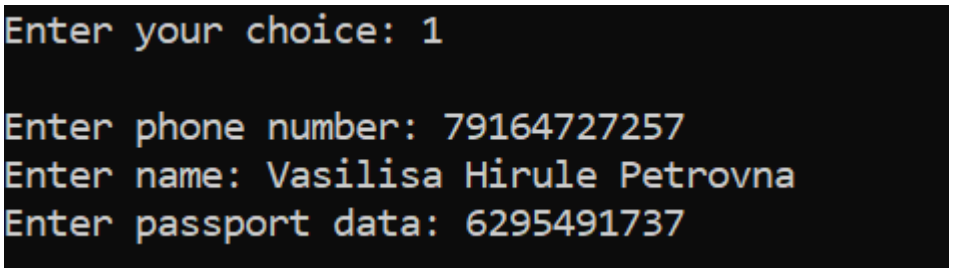

Рисунок 3.9 – Добавление нового элемента в словарь

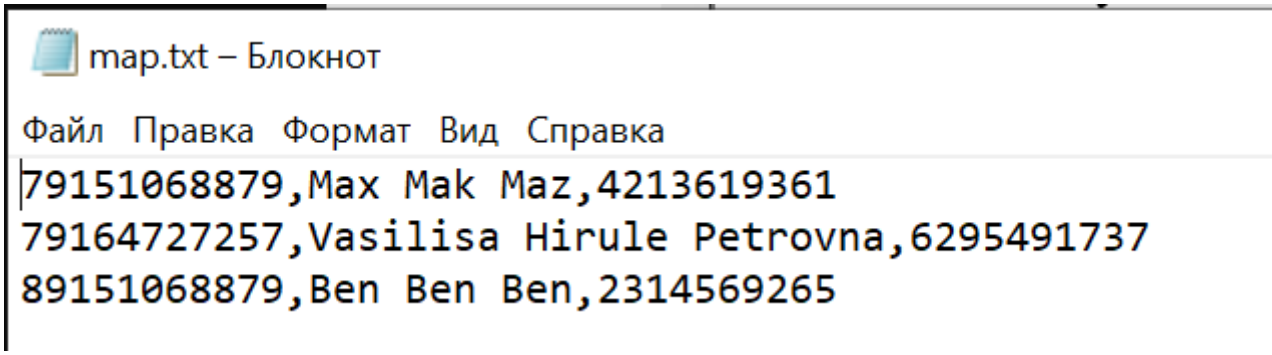

Рисунок 3.10 – Сохранение нового элемента в файл map.txt

При вводе цифры 2 пользователю нужно ввести телефонный номер (ключ), по которому удалится элемент (Рисунок 3.11). Все изменения будут отображены в текстовом файле (Рисунок 3.12).

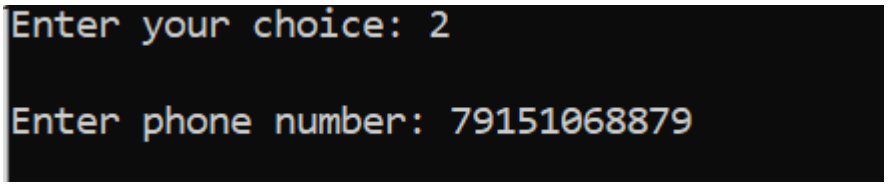

Рисунок 3.11 – Удаление элемента из словаря по его ключу

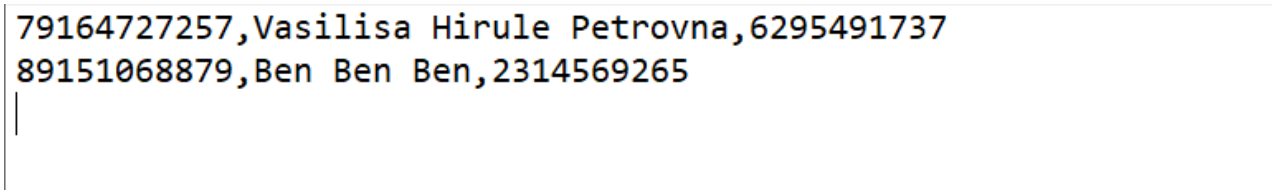

Рисунок 3.12 – Удаление элемента из текстового файла map.txt

При вводе цифры 3 пользователю будет предложено ввести номер телефона (ключ), для поиска информации об элементе (Рисунок 3.13).

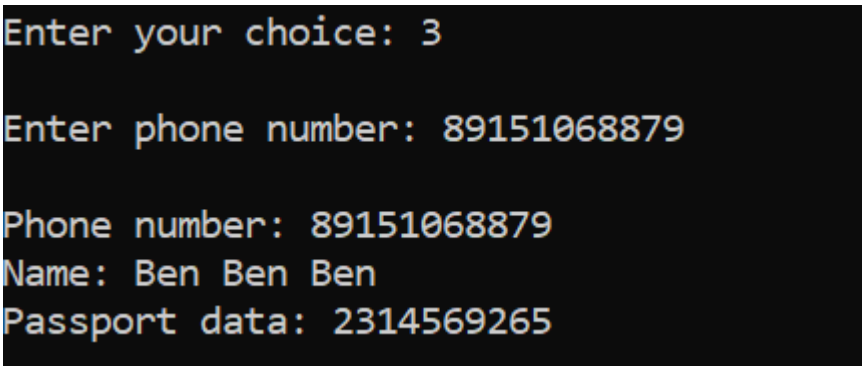

Рисунок 3.13 – Поиск элемента в словаре по его ключу

При вводе цифры 4 необходимо ввести телефонный номер (ключ элемента, который нужно изменить), далее ввести необходимые поправки (Рисунок 3.14). Все изменения сохраняться в текстовый файл (Рисунок 3.15)

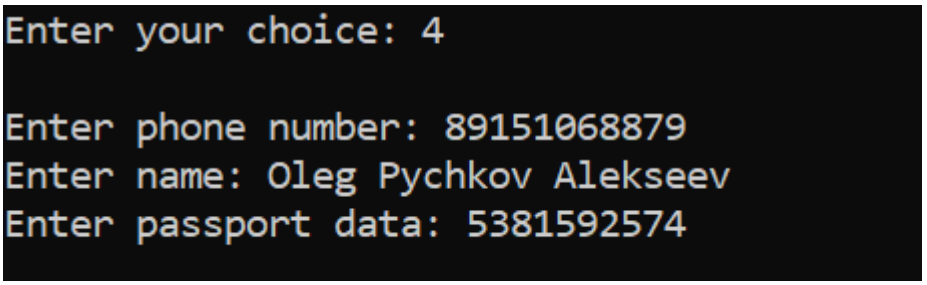

Рисунок 3.14 – Изменение элемента словаря по его ключу

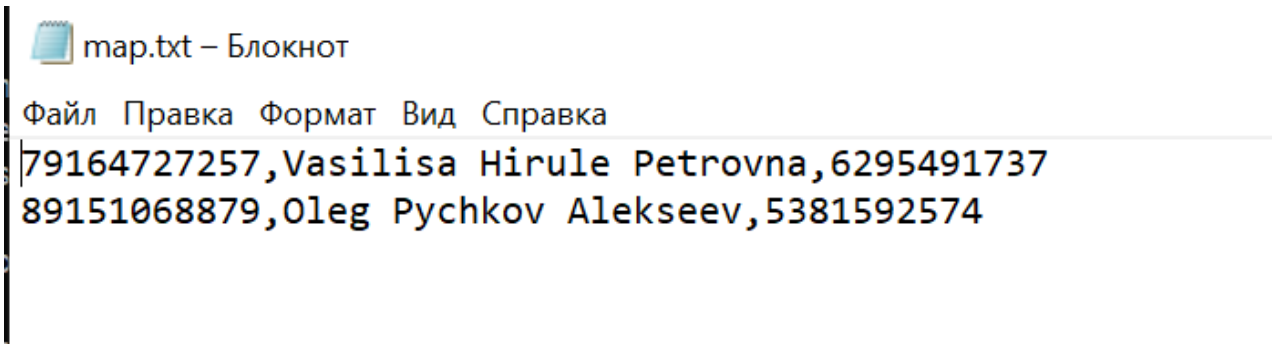

Рисунок 3.15 – Сохранение изменения данных в текстовый файл map.txt

При вводе цифры 5 в консоль будет выведены все элементы словаря (Рисунок 3.16).

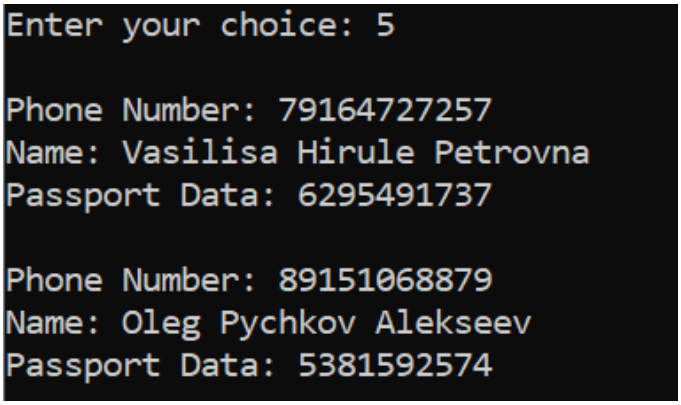

Рисунок 3.16 – Вывод словаря

Если пользователь введет цифру 6, то создастся пустой словарь, в котором не будет элементов (Рисунок 3.17).

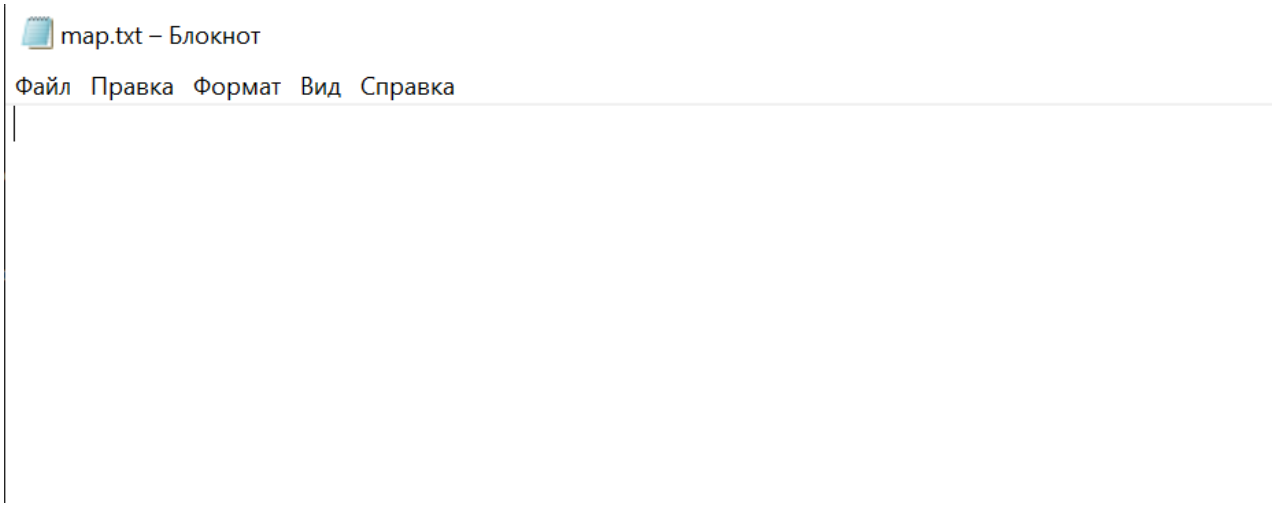

Рисунок 3.17 – Создание пустого словаря в текстовом файле map.txt

3.6 Входные данные (организация и предварительная подготовка входных данных)

В качестве входных данных программа принимает текст, который вводит пользователь (номер телефона, ФИО и паспортные данные). Также программа может принять данные из текстовых файлов, предварительно указав путь к ним.

3.7 Выходные данные

Выходные данные – это телефонный номер, ФИО и паспортные данные.

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++.

Выполнены поставленные задачи: создание программы, позволяющей работать со словарем, отладка и проверка работоспособности конечной программы.

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: 01.12.2023)

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: 01.12.2023)

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: 01.12.2023)

Приложение А

Листинг А – код реализации программы

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <algorithm>

struct Abonent {
    std::string name;
    std::string passportData;
};

class Dictionary {
private:
    std::map<std::string, Abonent> dict;

public:
    Dictionary() {}

    void add(const std::string& phoneNumber, const Abonent& abonent) {
        dict[phoneNumber] = abonent;
    }

    void remove(const std::string& phoneNumber) {
        dict.erase(phoneNumber);
    }

    Abonent find(const std::string& phoneNumber) const {
        auto it = dict.find(phoneNumber);

        if (it != dict.end()) {
            return it->second;
        }

        Abonent emptyAbonent;
        return emptyAbonent;
    }

    void update(const std::string& phoneNumber, const std::string& name,
const std::string& passportData) {
        auto it = dict.find(phoneNumber);

        if (it != dict.end()) {
            it->second.name = name;
            it->second.passportData = passportData;
        }
    }

    void print() const {
        for (const auto& abonent : dict) {
            std::cout << "Phone Number: " << abonent.first << std::endl;
            std::cout << "Name: " << abonent.second.name << std::endl;
            std::cout << "Passport Data: " <<
abonent.second.passportData << std::endl;
            std::cout << std::endl;
        }
    }
```

```
    void readFromFile(const std::string& filename) {
        std::ifstream file(filename);

        if (!file) {
            std::cerr << "Failed to open file" << std::endl;
            return;
        }

        std::string line;
        while (std::getline(file, line)) {
            std::string phoneNumber, name, passportData;
            size_t pos = line.find(',');

            if (pos != std::string::npos) {
                phoneNumber = line.substr(0, pos);
                line = line.substr(pos + 1);

                pos = line.find(',');
                if (pos != std::string::npos) {
                    name = line.substr(0, pos);
                    passportData = line.substr(pos + 1);

                    Abonent abonent;
                    abonent.name = name;
                    abonent.passportData = passportData;

                    dict[phoneNumber] = abonent;
                }
            }
        }

        file.close();
    }

    void writeToFile(const std::string& filename) const {
        std::ofstream file(filename);

        if (!file) {
            std::cerr << "Failed to open file" << std::endl;
            return;
        }

        for (const auto& abonent : dict) {
            file << abonent.first << "," << abonent.second.name << ","
<< abonent.second.passportData << std::endl;
        }

        file.close();
    }
};

bool CheckPhone(const std::string& phoneNumber) {
    if (phoneNumber.length() != 11) {
        return false;
    }
    if (phoneNumber[0] != '8' && phoneNumber[0] != '7') {
        return false;
    }
    for (char digit : phoneNumber) {
        if (!isdigit(digit)) {
            return false;
        }
    }
```

```
    return true;
}

bool CheckName(const std::string& name) {
    for (char a : name) {
        if (isdigit(a)) {
            return false;
        }
    }
    return true;
}

bool CheckPassport(const std::string& passportData) {
    if (passportData.length() != 10) {
        return false;
    }
    for (char digit : passportData) {
        if (!isdigit(digit)) {
            return false;
        }
    }
    return true;
}

int main() {
    Dictionary dictionary;
    std::string filename = "D:\\map.txt";
    dictionary.readFromFile(filename);

    std::string choice;
    while (true) {
        std::cout << "1. Add abonent" << std::endl;
        std::cout << "2. Remove abonent" << std::endl;
        std::cout << "3. Find abonent" << std::endl;
        std::cout << "4. Update abonent" << std::endl;
        std::cout << "5. Print dictionary" << std::endl;
        std::cout << "6. Create empty dictionary" << std::endl;
        std::cout << "7. Exit" << std::endl;
        std::cout << "Enter your choice: ";
        std::cin >> choice;
        std::cout << std::endl;

        if (choice == "1") {
            dictionary.readFromFile(filename);
            std::string phoneNumber, name, passportData;
            std::cout << "Enter phone number: ";
            std::getline(std::cin, phoneNumber);
            if (!CheckPhone(phoneNumber)) {
                std::cout << "Wrong phone mumber";
                return 0;
            }
            std::cout << "Enter name: ";
            std::getline(std::cin, name);
            if (!CheckName(name)) {
                std::cout << "Wrong name";
                return 0;
            }
            std::cout << "Enter passport data: ";
            std::getline(std::cin, passportData);
            if (!CheckPassport(passportData)) {
                std::cout << "Wrong passport data";
                return 0;
            }
```

```
            std::cout << std::endl;

            Abonent abonent;
            abonent.name = name;
            abonent.passportData = passportData;

            dictionary.add(phoneNumber, abonent);
            dictionary.writeToFile(filename);
        }
        else if (choice == "2") {
            std::string phoneNumber;
            std::cout << "Enter phone number: ";
            std::getline(std::cin, phoneNumber);
            std::cout << std::endl;
            dictionary.remove(phoneNumber);
            dictionary.writeToFile(filename);
        }
        else if (choice == "3") {
            std::string phoneNumber;
            std::cout << "Enter phone number: ";
            std::getline(std::cin, phoneNumber);
            std::cout << std::endl;

            Abonent abonent = dictionary.find(phoneNumber);

            if (abonent.name.empty()) {
                std::cout << "Abonent not found";
            }
            else {
                std::cout << "Phone number: " << phoneNumber <<
std::endl;
                std::cout << "Name: " << abonent.name << std::endl;
                std::cout << "Passport data: " << abonent.passportData
<< std::endl;
            }
            std::cout << std::endl;
        }
        else if (choice == "4") {
            std::string phoneNumber, name, passportData;
            std::cout << "Enter phone number: ";
            std::getline(std::cin, phoneNumber);
            std::cout << "Enter name: ";
            std::getline(std::cin, name);
            if (!CheckName(name)) {
                std::cout << "Wrong name";
                return 0;
            }
            std::cout << "Enter passport data: ";
            std::getline(std::cin, passportData);
            if (!CheckPassport(passportData)) {
                std::cout << "Wrong passport data";
                return 0;
            }
            std::cout << std::endl;
            dictionary.update(phoneNumber, name, passportData);
            dictionary.writeToFile(filename);
        }
        else if (choice == "5") {
            dictionary.print();
        }
        else if (choice == "6") {
            std::ofstream file(filename);
            file.close();
```

```
            break;
        }
        else if (choice == "7") {
            std::cout << "Stopping program...";
            break;
        }
    }
}
```