|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 3** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Разработка и программная реализация задач поиска данных в таблицах с применением механизма хеширования.»** | |
|  | |
| Выполнил студент группы ИКБО-11-22 | Гришин А. В. |
| Принял преподаватель | Скворцова Л. А. |

|  |  |  |
| --- | --- | --- |
| Самостоятельная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2023

**Содержание**

[1. Цель работы 3](#_Toc148908830)

[2. Задание 3](#_Toc148908831)

[2.1 Постановка задачи 3](#_Toc148908832)

[2.2 Код программы (персональный вариант 15) 6](#_Toc148908833)

[2.3. Тестирование программы. 10](#_Toc148908834)

[2.4. Входные данные программы. 12](#_Toc148908835)

# **Цель работы**

Получить навыки по разработке хеш-таблиц и их применении при поиске

данных в других структурах данных (файлах).

# **Задание**

## **Постановка задачи**

Разработать приложение, которое использует хеш-таблицу для организациипрямого доступа к записям двоичного файла, реализованного в практическойработе 2.

Метод разрешения коллизии представлен в вашем варианте задания в таб.1.

Требования к выполнению

1. Создать приложение и включить в него три заголовочных файла:управление хеш-таблицей, управление двоичным файлом (практическая работа2), управление двоичным файлом посредством хеш-таблицы. Именазаголовочным файлам определите сами. Подключите заголовочные файлы кприложению.

2. Для обеспечения прямого доступа к записи в файле элемент хеш-таблицыдолжен включать обязательные поля: ключ записи в файле, номер записи с этимключом в файле. Элемент может содержать другие поля, требующиеся методу(указанному в вашем варианте), разрешающему коллизию.

3. Управление хеш-таблицей.

1) Определить структуру элемента хеш-таблицы и структуру хеш-таблицы всоответствии с методом разрешения коллизии, указанном в варианте.Определения разместить в соответствующем заголовочном файле. Всеоперации управления хеш-таблицей размещать в этом заголовочномфайле.

2) Тестирование операций выполнять в функции main приложения по мере ихреализации.После тестирования всех операций, создать в заголовочном файлефункцию с именем testHeshT переместить в нее содержание функции main,проверить, что приложение выполняется.Разработать операции по управлению хеш-таблицей.

3) Разработать хеш-функцию (метод определить самостоятельно), выполнитьее тестирование, убедиться, что хеш (индекс элемента таблицы)формируется верно.

4) Разработать операции: вставить ключ в таблицу, удалить ключ из таблицы,найти ключ в таблице, рехешировать таблицу. Каждую операциютестируйте по мере ее реализации.

5) Подготовить тесты (последовательность значений ключей),обеспечивающие:

− вставку ключа без коллизии

− вставку ключа и разрешение коллизии

− вставку ключа с последующим рехешированием

− удаление ключа из таблицы

− поиск ключа в таблице

Примечание. Для метода с открытым адресом подготовить тест для поискаключа, который размещен в таблице после удаленного ключа, с одним значениемхеша для этих ключей.

6) Выполнить тестирование операций управления хеш-таблицей. Притестировании операции вставки ключа в таблицу предусмотрите выводсписка индексов, которые формируются при вставке элементов в таблицу.

4. Управление двоичным файлом.

Операции управления двоичным файлом: создание двоичного файла изтекстового, добавить запись в двоичный файл, удалить запись с заданнымключом из файла, прочитать запись файла по заданному номеру записи.

Примечание. Эти операции должны быть отлажены в практической работе 2, или уже в этой работе, если их пока нету.

Структура записи двоичного файла и все операции, по управлению файлом,должны быть размещены в соответствующем заголовочном файле.

Выполнить тестирование операций в main приложения, и содержаниефункции main переместить в соответствующую функцию заголовочного файла сименем testBinF.

5. Управление файлом посредством хеш-таблицы.

В заголовочный файл управления файлом посредством хеш-таблицыподключить заголовочные файлы: управления хеш-таблицей, управлениядвоичным файлом. Реализовать поочередно все перечисленные ниже операции вэтом заголовочном файле, выполняя их тестирование из функции mainприложения. После разработки всех операций выполнить их комплексноетестирование.

Разработать и реализовать операции.

1) Прочитать запись из файла и вставить элемент в таблицу (элемент включает: ключ и номер записи с этим ключом в файле, и для метода с открытой адресацией возможны дополнительные поля).

2) Удалить запись из таблицы при заданном значении ключа и соответственно из файла.

3) Найти запись в файле по значению ключа (найти ключ в хеш-таблице, получить номер записи с этим ключом в файле,выполнить прямой доступ к записи по ее номеру).

4) Подготовить тесты для тестирования приложения: Заполните файл небольшим количеством записей.

− Включите в файл записи как не приводящие к коллизиям, так и приводящие.

− Обеспечьте включение в файл такого количества записей, чтобы потребовалось рехеширование.

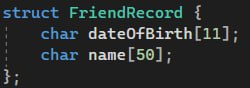
Заполните файл большим количеством записей (до 1 000 000).

− Определите время чтения записи с заданным ключом: для первой записи файла, для последней и где-то в середине. Убедитесь (или нет), что время доступа для всех записей одинаково.

## **Код программы (персональный вариант 15)**

Помимо функций, которые были реализованы в практической работе №2 (такие как добавление записи, удаление записи и т.д.) были добавлены следующие функции.

Структура записи двоичного файла (вариант 15): дата рождения, имя (рис. 1).



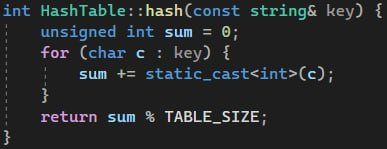
*Рисунок 1 – структура FriendRecord*

Функция insertDataToHashTable записывает данные бинарного файла в хэш-таблицу (рис. 2).



*Рисунок 2 – Функция insertDataToHashTable*

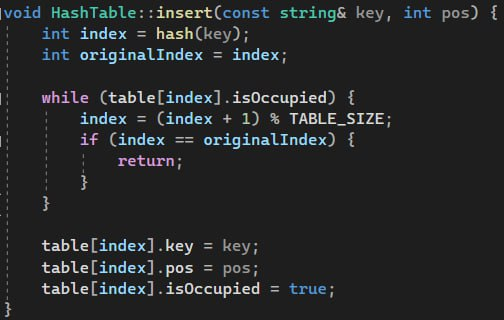
Функция HashTable::hash используется для хэширования ключа (рис. 3).



*Рисунок 3 – Функция HashTable::hash*

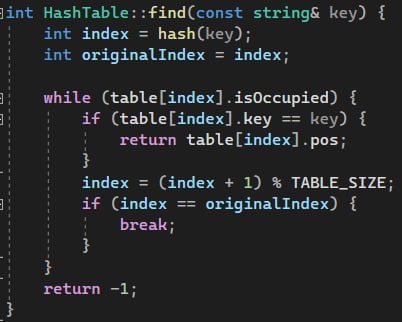
Функция HashTable::insert реализует тип хэширования «Открытый адрес (смещение на 1)» (рис. 4).

Тип хэширования "Открытый адрес (смещение на 1)" представляет собой метод разрешения коллизий в хэш-таблице, где элементы, имеющие одинаковое значение хэша, размещаются в той же таблице, смещаясь на фиксированное смещение (обычно 1) от исходной позиции, до тех пор, пока не будет найдена свободная ячейка.



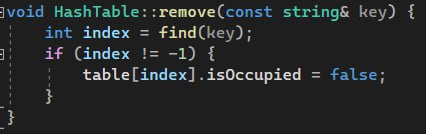
*Рисунок 4 – Функция HashTable::insert*

Функция HashTable::find используется для поиска элемента по ключу в хэш-таблице (рис. 5)



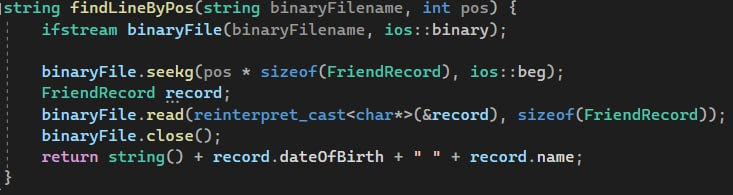
*Рисунок 5 – Функция HashTable::find*

Функция HashTable::remove используется для удаления элемента из хэш-таблицы по ключу (рис. 6)



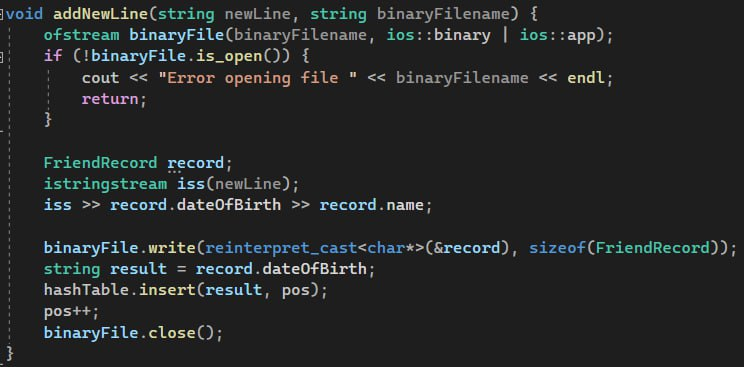
*Рисунок 6 – Функция HashTable::remove*

Функция findLineByPos реализует поиск записи в бинарном файле с именем “binaryFilename” по заданной позиции “pos” (рис. 7). Поиск реализуется посредством перемещения указателя в файле, чтения соответствующей записи и возврата информации о записи в виде строки.



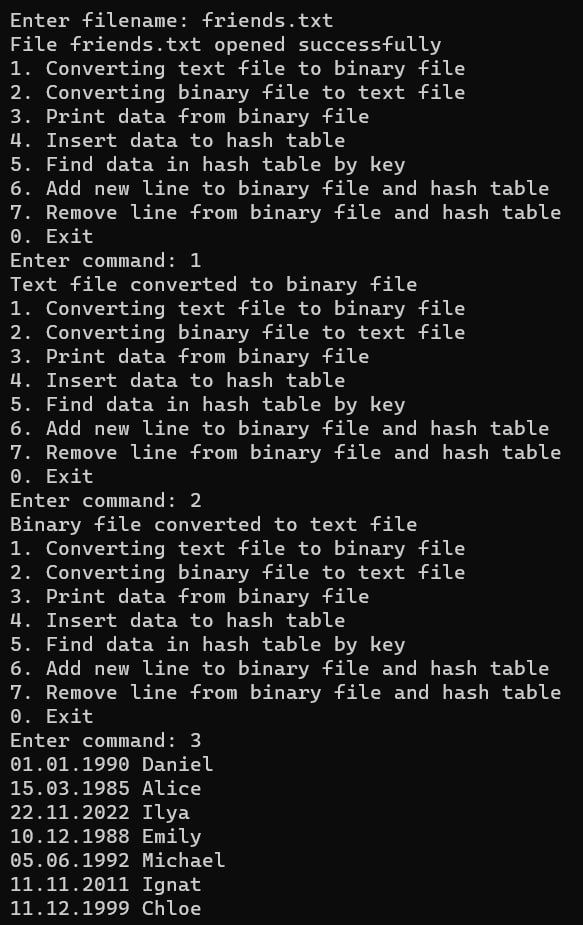
*Рисунок 7 – Функция findLineByPos*

Функция addNewLine реализует добавление новой записи в бинарный файл и хэш-таблицу (рис. 8).

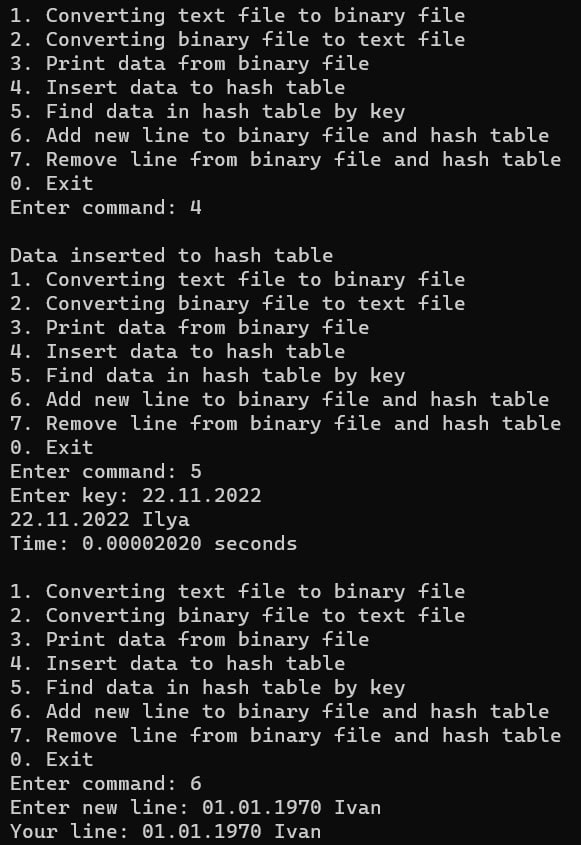


*Рисунок 8 – Функция addNewLine*

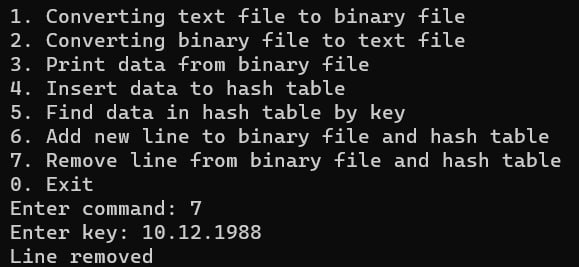
## **2.3. Тестирование программы.**



*Рисунок 9 — вывод программы.*

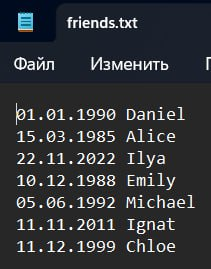


*Рисунок 10 — вывод программы.*



*Рисунок 11 — вывод программы.*

## **2.4. Входные данные программы.**



*Рисунок 12 — входные данные программы.*