|  |
| --- |
| https://www.mirea.ru/bitrix/templates/unlimtech/images/logo.png |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |

|  |  |
| --- | --- |
| Отчет по выполнению практического задания №2  По дисциплине «Структуры и алгоритмы обработки данных» | |
| **Тема:**  Хеширование и организация быстрого поиска данных. | |
|  | |
| Выполнил студент | Сидоров С.Д. |
| группа | ИКБО-20-21 |

Тема. Хеширование и организация быстрого поиска данных.

Цель. Получить навыки по разработке хеш-таблиц и их применении.

**Задание:** Разработайте приложение, которое использует хеш-таблицу для организации

прямого доступа к записям файла, структура записи которого приведена в

варианте.

Разработайте и реализуйте функции для операций:

1) Хеш-функцию (метод определите сами).

2) Прочитать запись из файла и вставки запись в таблицу (запись

включает: ключ и номер записи с этим ключом в файле).

3) Удалить запись из таблицы и соответственно из файла.

4) Найти запись с заданным ключом в файле, используя хеш-таблицу.

5) Выполнить рехеширование.

2. Разработайте такие тесты, чтобы возникли коллизии.

3. Разработайте такие тесты, чтобы требовалось рехеширование.

4. Заполните файл большим количеством записей. 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.

5. Выведите список индексов, которые формируются при вставке элементов в

таблицу.

**Вариант 5:** Пациент поликлиники: номер карточки, код

хронического заболевания, Фамилия лечащего

врача

**Исполнение**:

Дано: Хэш-таблица для реализации коллизий по методу *Цепное хеширование.*

Файл двоичный с записями фиксированный длины.

Структура записи в файл:

Номер карты, код болезни, фамилия врача.

Разработка:

Создана struct содержащие структуру записи с номером строки в файле (Листинг 1.1).

Создан класс hash\_table, который использует хэш-функцию ( номер карты mod размер хэш-таблицы ) для добавления записи из файла, и рехэширование таблицы. Рехэширование таблицы происходит засчет создания таблицы удвоенной длины и переноса уже существующих данных. Также внутри класса реализованы методы для удаления и поиска записи среди таблицы. ( Листинг 1.2 -> расположено в файле hash\_table.docx)

Класс hash\_table имеет поля real\_size - для хранения настоящего размера таблицы, current\_size - для хранения количества непустых элементов в таблице, vector<list<patient>> table - вектор содержащей списки структур, каждый из которых являются цепью для реализации коллизии по методу *Цепное хэширование.*

|  |
| --- |
| struct patient {  int card\_number;  int illness\_code;  string doctor\_surname;  int index;  }; |

Листинг 1.2

Работа с файлом: использована библиотека fstream, с помощью нее создавались объекты классов ofstream и ifstream, благодаря которым можно открывать бинарные файлы для чтения и записи. Используя данные объекты считывается таблица с файла, добавляются новые записи, удаляются записи, а также выдаются записи в файле по индексу хранящемся в хэш-таблице. Работа с файлом и работа с пользователя находится в файле main.docx

Для генерации тестов использовалась функция generate\_file которая генерирует информацию о пациентах и записывает в файл. (Листинг 1.3)

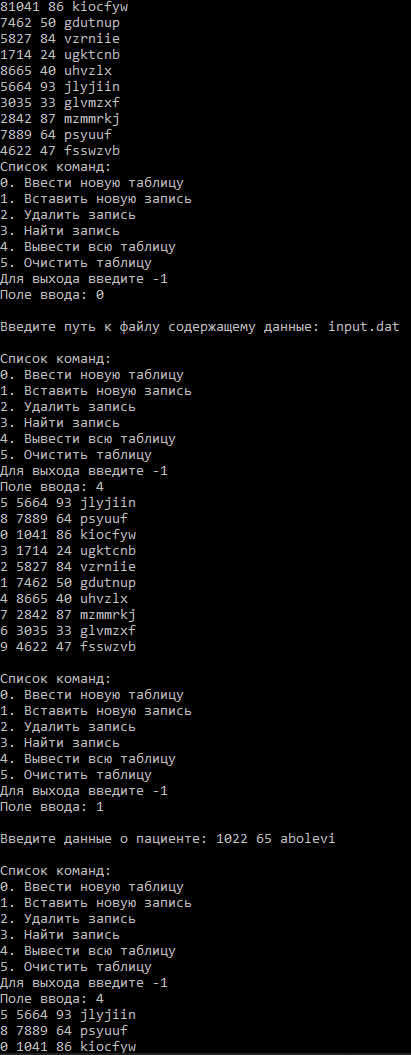
|  |
| --- |
| string generate\_string(int length) {  string alphabet[] = {"alebiev" ,"kusckov", "muchkov", "buchkov"};  string result = "";  result = alphabet[rand() % 4 ];  return result;  }  void generate\_file(ofstream& of, int count\_lines) {  patient tmp;  for (int i = 0; i < count\_lines; i++) {  std::default\_random\_engine g(std::chrono::system\_clock::now().time\_since\_epoch().count());  std::uniform\_int\_distribution<int> distribution(10000, 99999);  auto roll = distribution(g);  tmp.card\_number = roll;  tmp.illness\_code = rand() % (101) + 1;  tmp.doctor\_surname = generate\_string(7);  tmp.index = i;  of.write((char\*)&tmp, sizeof(patient));  }  of.close();  } |

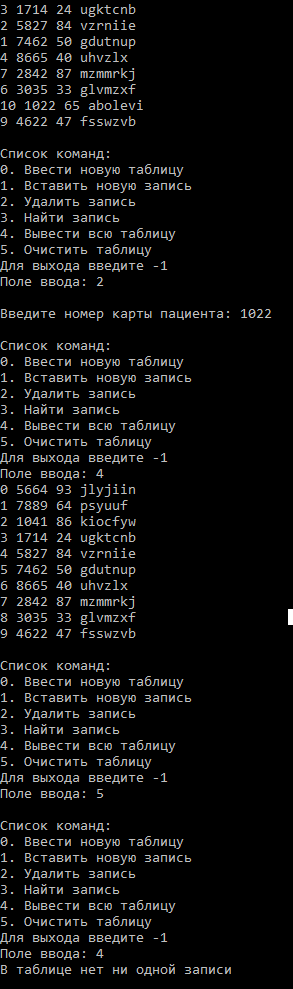
Листинг 1.3

**Тестирование:**

Проверка работоспособности программы:

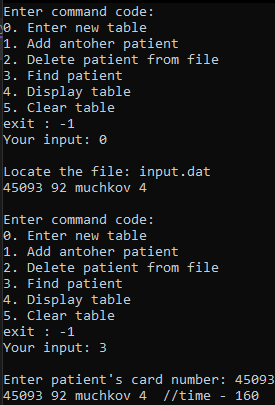
Данный тест проверял все функции программы на файле с 10-ю записями, изначальный размер таблицы - 8, следовательно во время чтения с файла происходило рехэширование, для точного появления коллизии в таблицу добавлялся отдельный элемент 1022, который попадает в ту же ячейку вектора, что и элемент 4622. Тестирование показало, что программа работает корректно.



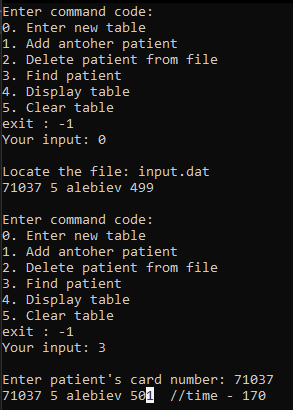


Измерение скорости работы программы*:*

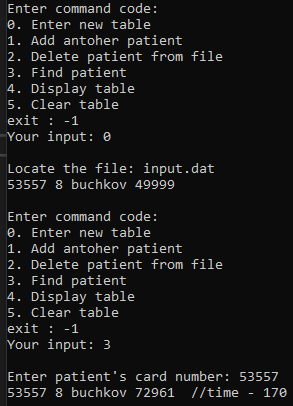
10 элементов:



1000 элементов:



100000 элементов:



В ходе данного тестирования было выявлено, что в независимости от количества элементов, время работы примерно одинаково. Следовательно хеш-таблица работает корректно.

**Ответы на вопросы из задания:**

1. Расскажите о назначении хеш-функции:

Хэш-функция преобразует значение ключа в индекс хеш-таблцы.

1. Что такое коллизия?

Коллизия - это ситуация, когда для двух разных ключей хеш-функция создаёт одинаковый индекс.

1. Что такое «открытый адрес» по отношению к хеш-таблице?

Открытый адрес - свободная ячейка таблицы

1. Как в хеш-таблице с открытым адресом реализуется коллизия?

Коллизия в хеш-таблице с открытым адресом реализуется посредством поиска открытого адреса по ячейки полученной на выходе из хеш-функции.

1. Какая проблема, может возникнуть после удаления элемента из хеш-таблицы с открытым адресом и как ее устранить?

Во время поиска перебираются все элементы начиная с индекса полученного от хеш-функции заканчивая пустой ячейкой, но если ячейка перед искомой была заранее удалена, то мы не дойдём до искомой в процессе поиска. Для решения данной проблемы можно ввести параметр показывающий, что ячейка была удалена, а не просто свободна, в таком случае поиск не прервётся на удалённой ячейке.

1. Что определяет коэффициент нагрузки в хеш-таблице?

Коэффициент нагрузки в хеш-таблице это отношение количества размещённых в хеш таблице записей с данными к длине таблицы. Позволяет определить момент в который необходимо совершить рехеширование.

1. Что такое «первичный кластер» в таблице с открытым адресом?

Первичная кластеризация – это явление, которое происходит, когда обработчик коллизий создаёт условие роста кластера. Обработчик коллизий со смещением 1 способствует первичной кластеризации. Первичная кластеризация порождает длинные пути.

1. Как реализуется двойное хеширование?

Двойное хеширование - устранение проблемы первичной кластеризации, посредством установки смещения в зависимости от ключа, т.е. получать его посредством хеш-функции.

**Выводы**:

В ходе выполнения работы была реализована хеш-таблица с цепным хешированием. Были реализованы алгоритмы поиска, вставки и удаления записей в таблице, также был реализован алгоритм вывода таблицы в консоль, а также применена хеш-таблица для работы с бинарным файлом.

**Список литературы:**

⦁ Лекции по структурам и алгоритмам обработки данных Рысин М.Л.

⦁ Методическое пособие по выполнению задания 1(битовые операции)