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.

```
81041 86 kiocfyw
7462 50 gdutnup
5827 84 vzrniie
1714 24 ugktcnb
8665 40 uhvzlx
5664 93 jlyjiin
3035 33 glvmzxf
2842 87 mzmmrkj
7889 64 psyuuf
4622 47 fsswzvb
Список команд:
0. Ввести новую таблицу
1. Вставить новую запись
2. Удалить запись
3. Найти запись
4. Вывести всю таблицу
5. Очистить таблицу
Для выхода введите -1
Поле ввода: 0

Введите путь к файлу содержащему данные: input.dat

Список команд:
0. Ввести новую таблицу
1. Вставить новую запись
2. Удалить запись
3. Найти запись
4. Вывести всю таблицу
5. Очистить таблицу
Для выхода введите -1
Поле ввода: 4
5 5664 93 jlyjiin
8 7889 64 psyuuf
0 1041 86 kiocfyw
3 1714 24 ugktcnb
2 5827 84 vzrniie
1 7462 50 gdutnup
4 8665 40 uhvzlx
7 2842 87 mzmmrkj
6 3035 33 glvmzxf
9 4622 47 fsswzvb

Список команд:
0. Ввести новую таблицу
1. Вставить новую запись
2. Удалить запись
3. Найти запись
4. Вывести всю таблицу
5. Очистить таблицу
Для выхода введите -1
Поле ввода: 1

Введите данные о пациенте: 1022 65 abolevi

Список команд:
0. Ввести новую таблицу
1. Вставить новую запись
2. Удалить запись
3. Найти запись
4. Вывести всю таблицу
5. Очистить таблицу
Для выхода введите -1
Поле ввода: 4
5 5664 93 jlyjiin
8 7889 64 psyuuf
0 1041 86 kiocfyw
```

```
3 1714 24 ugktcnb
2 5827 84 vzrniie
1 7462 50 gdutnup
4 8665 40 uhvzlx
7 2842 87 mzmmrkj
6 3035 33 glvmzxf
10 1022 65 abolevi
9 4622 47 fsswzvb

Список команд:
0. Ввести новую таблицу
1. Вставить новую запись
2. Удалить запись
3. Найти запись
4. Вывести всю таблицу
5. Очистить таблицу
Для выхода введите -1
Поле ввода: 2

Введите номер карты пациента: 1022

Список команд:
0. Ввести новую таблицу
1. Вставить новую запись
2. Удалить запись
3. Найти запись
4. Вывести всю таблицу
5. Очистить таблицу
Для выхода введите -1
Поле ввода: 4
0 5664 93 jlyjiin
1 7889 64 psyuuf
2 1041 86 kiocfyw
3 1714 24 ugktcnb
4 5827 84 vzrniie
5 7462 50 gdutnup
6 8665 40 uhvzlx
7 2842 87 mzmmrkj
8 3035 33 glvmzxf
9 4622 47 fsswzvb

Список команд:
0. Ввести новую таблицу
1. Вставить новую запись
2. Удалить запись
3. Найти запись
4. Вывести всю таблицу
5. Очистить таблицу
Для выхода введите -1
Поле ввода: 5

Список команд:
0. Ввести новую таблицу
1. Вставить новую запись
2. Удалить запись
3. Найти запись
4. Вывести всю таблицу
5. Очистить таблицу
Для выхода введите -1
Поле ввода: 4
В таблице нет ни одной записи
```

Измерение скорости работы программы:

10 элементов:

```
Enter command code:
0. Enter new table
1. Add antoher patient
2. Delete patient from file
3. Find patient
4. Display table
5. Clear table
exit : -1
Your input: 0

Locate the file: input.dat
45093 92 muchkov 4

Enter command code:
0. Enter new table
1. Add antoher patient
2. Delete patient from file
3. Find patient
4. Display table
5. Clear table
exit : -1
Your input: 3

Enter patient's card number: 45093
45093 92 muchkov 4  //time - 160
```

1000 элементов:

```
Enter command code:
0. Enter new table
1. Add antoher patient
2. Delete patient from file
3. Find patient
4. Display table
5. Clear table
exit : -1
Your input: 0

Locate the file: input.dat
71037 5 alebiev 499

Enter command code:
0. Enter new table
1. Add antoher patient
2. Delete patient from file
3. Find patient
4. Display table
5. Clear table
exit : -1
Your input: 3

Enter patient's card number: 71037
71037 5 alebiev 501  //time - 170
```

100000 элементов:

```
Enter command code:
0. Enter new table
1. Add antoher patient
2. Delete patient from file
3. Find patient
4. Display table
5. Clear table
exit : -1
Your input: 0

Locate the file: input.dat
53557 8 buchkov 49999

Enter command code:
0. Enter new table
1. Add antoher patient
2. Delete patient from file
3. Find patient
4. Display table
5. Clear table
exit : -1
Your input: 3

Enter patient's card number: 53557
53557 8 buchkov 72961  //time - 170
```

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.

2. Что такое коллизия?

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

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

Открытый адрес - свободная ячейка таблицы

4. 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.

5. 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.

6. 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.

7. 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.

8. 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)