**ФГБОУ ВО   
Уфимский университет науки и технологий**

**Кафедра ВМиК**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 90 |  |  |  |  |  |  |  |  |  |  |  |  |
| 80 |  |  |  |  |  |  |  |  |  |  |  |  |
| 70 |  |  |  |  |  |  |  |  |  |  |  |  |
| 60 |  |  |  |  |  |  |  |  |  |  |  |  |
| 50 |  |  |  |  |  |  |  |  |  |  |  |  |
| 40 |  |  |  |  |  |  |  |  |  |  |  |  |
| 30 |  |  |  |  |  |  |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

Построение хеш-таблицы

**ОТЧЕТ**

**по лабораторной работе**

**по** Структурам и алгоритмам

компьютерной обработки данных

(*наименование дисциплины*)

|  |
| --- |
| Лабораторная работа 1 (Вар. 11) |
| (обозначение документа) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Группа |  |  | Фамилия, И., О. | Подпись | Дата | Оценка |
| номер |  |
|  |  |
| Студент | | | Студент\_ФИО |  |  |  |
| Преподаватель | | | Преподаватель\_ФИО |  |  |  |
| Принял | | |  |  |  |  |

**Уфа 2024 г****.**

Содержание

[1 Цель работы 3](#_Toc179833769)

[2 Практическая часть 4](#_Toc179833770)

[2.1 Задание на лабораторную работу 4](#_Toc179833771)

[2.2 Входные и выходные данные 4](#_Toc179833772)

[2.3 Порядок построения хеш-таблицы 4](#_Toc179833773)

[2.4 Формулы расчета параметров хеш-таблицы 8](#_Toc179833774)

[3 Вывод 9](#_Toc179833775)

# Цель работы

Целью работы является построение хеш-таблицы, содержащей заданную последовательность элементов (ключей).

# Практическая часть

## Задание на лабораторную работу

Сгенерировать 45 шестизначных неповторяющихся чисел (элементов). Вывести их на экран. Построить хеш-таблицу, используя: хеш-функцию – сложение по модулю пары соседних цифр ключа, метод устранения коллизий – открытая адресация с квадратичным опробыванием.

## Входные и выходные данные

Входные данные:

* size\_sample – количество шестизначных элементов;
* sample[size\_sample] – массив сгенерированных элементов;

Промежуточные данные:

* transition – число шагов, на котором произойдет замена метода устранения коллизий с открытой адресации с квадратичным опробыванием на открытую адресацию с линейным опробыванием;
* steps – количество шагов, потраченное на размещение всех элементов в хеш-таблице.

Выходные данные:

* size\_table – размер таблицы, который равен количеству элементов, умноженных в полтора раза;
* table[size\_table] – построенная хеш-таблица;
* size\_sample / size\_table – коэффициент заполнения таблицы;
* steps / size\_sample – среднее число шагов.

## Порядок построения хеш-таблицы

Для построения хеш-таблицы необходимо следовать алгоритму, который представлен ниже.

Сначала необходимо сгенерировать случайную последовательность неповторяющихся элементов:

* запустим цикл «while», в котором в «r» применением комбинации с использованием функции «rand()» будем записывать случайное значение (рисунок 2.1);

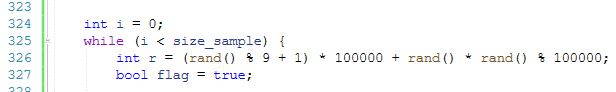


Рисунок 2.1 – Генерация случайного значения

* далее вложенным циклом «for» проходим по массиву сгенерированных элементов, и если полученный на i-ом шаге элемент «r» не встретился в массиве, то добавляем его (рисунок 2.2).

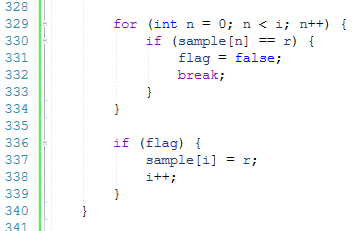


Рисунок 2.2 – Проверка уникальности

На рисунке 2.3 приведен пример сгенерированных уникальных неповторяющихся ключей.

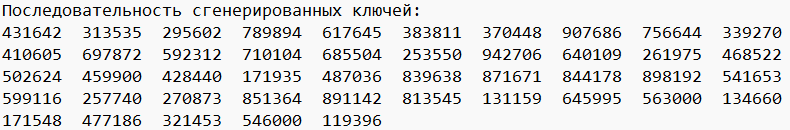


Рисунок 2.3 – Пример сгенерированных элементов

Теперь необходимо создать хеш-таблицу, которую нужно инициализировать нулями так, как показано на рисунке 2.4.

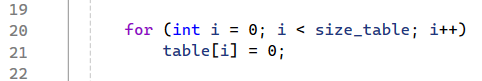


Рисунок 2.4 – Инициализация хеш-таблицы

Далее заполним хеш-таблицу случайными элементами, для чего выполним программную реализацию хеш-функции и метода устранения коллизий следующим образом:

* вычисляем хеш-функцию, как представлено на рисунке 2.5;

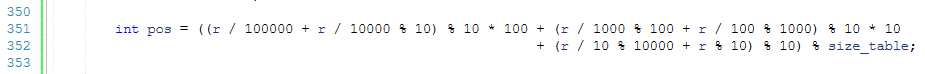


Рисунок 2.5 – Хеш-функция

* если в хеш-таблице по полученному индексу находится ноль, то в него [индекс] заносится элемент (рисунок 2.6);

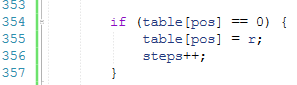


Рисунок 2.6 – Вставка элемента в хеш-таблицу

* если в данном индексе уже находится другой элемент, то применяется метод устранения коллизий – открытая адресация с квадратичным опробыванием (рисунок 2.7);

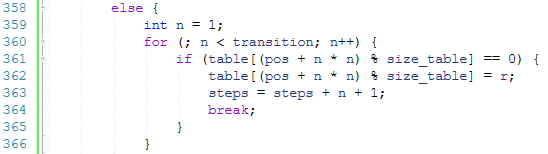


Рисунок 2.7 – Открытая адресация с квадратичным опробыванием

* если за 50[[1]](#footnote-1) итераций этого метода так и не удалось занести элемент в таблицу, то метод устранения коллизий заменяется на открытую адресацию с линейным опробыванием (рисунок 2.8).

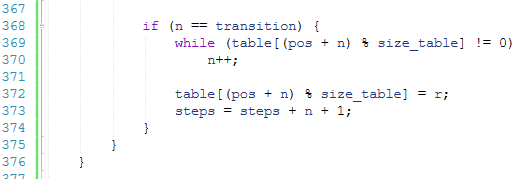


Рисунок 2.8 – Открытая адресация с линейным опробыванием

Наконец при помощи цикла «for» выведем в консоль заполненную хеш-таблицу так, как показано на рисунке 2.9. Пример вывода представлен на рисунке 2.10.

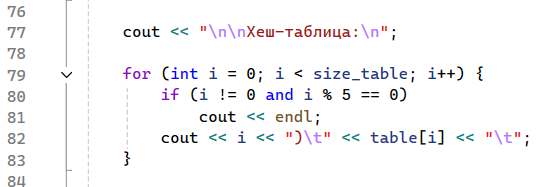


Рисунок 2.9 – Вывод хеш-таблицы в консоль

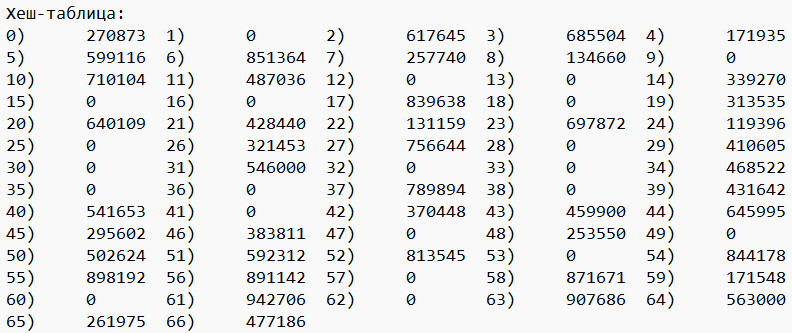


Рисунок 2.10 – Пример полученной хеш-таблицы

## Формулы расчета параметров хеш-таблицы

Коэффициент заполнения таблицы рассчитывается делением количества элементов на размер таблицы:

Среднее число шагов, необходимых для размещения некоторого ключа в таблице – это отношение количества шагов к количеству элементов:

В ЯП «C++» расчёт параметров реализовывается так, как показано на рисунке 2.11. Пример вычисленных параметров изображен на рисунке 2.12.

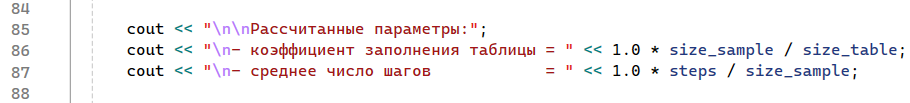


Рисунок 2.11 – Формулы расчета параметров

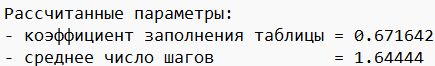


Рисунок 2.12 – Пример вычисленных параметров

# Вывод

В ходе лабораторной работы мы реализовали построение хеш-таблицы, которая содержит заданную последовательность элементов (ключей).

1. 50 итераций, так как «transition = 50». [↑](#footnote-ref-1)