Лабораторная работа

Хэш-таблицы

### Цель работы:

Создать у студентов представление о принципе хэширования и функционировании хэш-таблиц.

### Теоретические сведения:

Хэширование – это преобразование входного массива данных определенного типа и произвольной длины в выходную битовую строку (целое число) фиксированной длины. Такие преобразования также называются хэш-функциями, а их результаты называют хэшем.

Хэш-функция — это функция, которая осуществляет превращение входных данных в число, причём один и тот же набор данных всегда даёт один и тот же результат.

Некоторые виды хэш-функций:

1. **Метод остатков от деления:**

Простейшей хэш-функцией является деление по модулю числового значения ключа Key на размер хэш-таблицы. Эта функция достаточно легко вычислима, но вместе с тем выдаёт достаточно большое число коллизий. Хотя на практике является наиболее распространённой.

int Hash(int Key, int HashTableSize) {

return Key % HashTableSize;

};

1. **Метод середины квадрата:**

Следующей хэш-функцией является функция середины квадрата. Значение ключа преобразуется в число, это число затем возводится в квадрат. Хэш получается из путём выделения нескольких цифр из середины полученного числа.

int Hash(int Key) {

int temp = pow(Key, 2); //возводим число в квадрат

if (temp > 100) { //проверяем величину квадрата

char buffer[25];

char \*p = (char\*)\_itoa\_s(temp, buffer, 10); //перевод числа в строку

std::string str(buffer); //получение представления в std::string

if (str.size() > 3) {

str = str.substr(str.size() / 2, 2);

}

else {

str = str.substr(str.size() / 2, 1);

}

temp = stoi(str);

}

return temp;

};

### Преобразование систем счисления:

Как хэш-функцию можно использовать преобразования систем счисления. Ключ, записанный как число в некоторой системе счисления P, представляется как число в системе счисления Q>P. (обычно Q=P+1). Это число переводится из системы Q обратно в систему P, приводится к размеру пространства записей и используется в качестве адреса.

**Эффективная хэш-функция:**

1. Проста с вычислительной точки зрения;
2. Минимизирует число коллизий;
3. Не даёт сходным ключам сходные значения.

**Хэш-таблица** – это структура данных, представляющая из себя ассоциативный массив (элементами которого являются пары «ключ-значение»). Принцип хэш-таблицы включает в себя три действия:

1. Добавление нового элемента;
2. Поиск элемента по ключу;
3. Удаление элемента по ключу.

Выполнение операции в хэш-таблице начинается с вычисления хэш-функции от ключа. Получившийся хэш является индексом элемента в массиве.

**Коэффициент заполнения хэш-таблицы** - количество хранимых элементов массива, деленное на число возможных значений хэш-функции. От него зависит среднее время выполнение операций в хэш-таблице

**Коллизия –** это ситуация, которая возникает, когда хэш-функция превращает два разных набора данных в одинаковый хэш (вследствие чего им соответствует одна и та же ячейка таблицы).

Основные принципы разрешения коллизий:

1. **Метод цепочек:**

Данный подход к хэшированию подразумевает хранение в хэш-таблице списков в качестве элементов (пустых по умолчанию). При добавлении данных в таблицу вычисляется их хэш, и они добавляются в соответствующий хэшу список. Весь процесс проиллюстрирован рисунком 1.

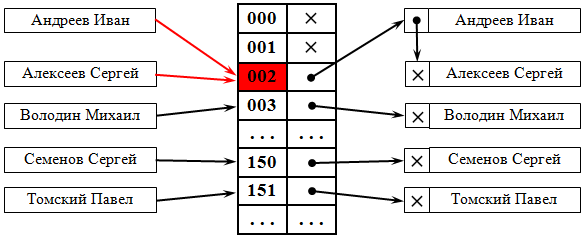


Рисунок 1 – Метод цепочек при хэшировании.

Коллизии приводят к тому, что появляются цепочки длиной более одного элемента. Операции поиска или удаления данных требуют просмотра всех элементов соответствующего ему списка, чтобы найти в нём элемент с заданным ключом. Данные добавляются в начало или конец списка, и, если коэффициент заполнения превысит 1, имеет смысл увеличить размер массива и перестроить таблицу.

Примерный класс хэш-таблицы:

class HashTable{

List\* table; //указатель для инициализации массива списков.

int table\_size;//размер таблицы

HashTable();//конструктор по умолчанию

int Hash(int key); //хэш-функция

void Insert(std::string s); //функции вставки, поиска удаления

void Search(std::string s); //s – ключ

void Delete(std::string s);

};

В таблице содержатся списки элементов типа string, присутствуют функции вставки поиска и удаления (в данном случае рекомендуется написать соответствующие функции для списка, чтобы облегчить работу с хэш-таблицей);

1. **Метод открытой адресации:**

При открытой адресации в хэш-таблице хранятся непосредственно данные. При добавлении нового элемента вычисляется его хэш, и элемент записывается в соответствующую ячейку. Если ячейка занята, то путём повторного хеширования вычисляется новое место для данного элемента. Метод открытой адресации изображён на рисунке 2.

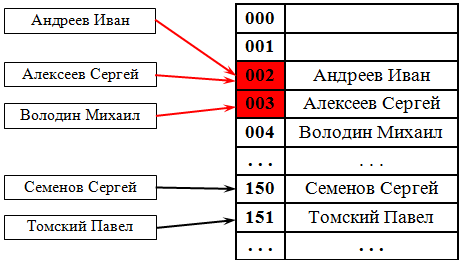


Рисунок 2 – Метод открытой адресации при хэшировании.

Если ячейка с хэшем от некоторых данных уже занята, то для поиска нового места применяется повторное хэширование. При помощи того же принципа осуществляется поиск и удаление этих данных. От добавления этот процесс отличается только проверкой, совпадает ли элемент в ячейке с искомым.

**Методы повторного хеширования:**

1. Линейный поиск. При коллизии для поиска нового места используется линейный перебор индексов массива. Примерная формула для вычислений: *адрес=h(x)+ci,* где i – номер попытки разрешить коллизию; c – константа, определяющая шаг перебора; h(x) – хэш-функция;
2. Квадратичный поиск. От линейного отличается только наличием квадратичной зависимости в формуле вычислений: *адрес=h(x)+ci+ai^2, где* где i – номер попытки разрешить коллизию; a, c – константы; h(x) – хэш-функция;
3. Двойное хэширование. Для поиска нового места используется дополнительная хэш-функция: *адрес=h(x)+ih2(x),* где i – номер попытки разрешить коллизию; h(x),h2(x) – хэш-функции.

Примерный класс хэш-таблицы в таком случае выглядит так:

class HashTable{

std::string\* table; //указатель для инициализации массива таблицы.

int table\_size;//размер таблицы

HashTable();//конструктор по умолчанию

int Hash(int key); //хэш-функция

void Insert(std::string s); //функции вставки, поиска удаления

void Search(std::string s);

void Delete(std::string s);

};

В таблице содержатся элементы типа string, присутствуют функции вставки поиска и удаления. При коллизии для элемента линейным перебором (присутствует в каждой из необходимых функций) будет найдено новое место для вставки. Если его нет – хэш-таблица расширяется и перестраивается.

### Постановка задачи:

1. Создать динамический массив из записей (в соответствии с вариантом), содержащий не менее 100 элементов;
2. Создать хэш-таблицу, у которой реализованы все основные функции (добавление элемента, удаление элемента, поиск по ключу) и внести в неё массив. Для хэширования применить функцию, указанную в варианте;
3. Выполнить поиск элемента в массиве по ключу, указанному в варианте;
4. Подготовить отчёт о проделанной работе.

**Варианты:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вариант** | **Данные** | **Ключ (string)** | **Хэш-функция** | **Метод разрешения коллизий** |
| 1 | ФИО, группа, рейтинг | ФИО | Остаток от деления на размер таблицы. | Метод цепочек |
| 2 | ФИО, №счета, сумма | №счета | Метод середины квадрата | Метод открытой адресации |
| 3 | ФИО, №счета, сумма | ФИО | Остаток от деления на размер таблицы. | Метод открытой адресации |
| 4 | ФИО, №паспорта, адрес, №телефона | №паспорта | Ключ умножить на 0.76, взять дробную часть, перевести её в целую, взять остаток от деления на размер таблицы. | Метод цепочек |
| 5 | ФИО, №паспорта, адрес | ФИО | Остаток от деления на размер таблицы. | Метод цепочек |
| 6 | ФИО, №паспорта, адрес | Адрес | Ключ умножить на 0.23, взять дробную часть, перевести её в целую, взять остаток от деления на размер таблицы. | Метод открытой адресации |
| 7 | ФИО, №телефона, адрес | №телефона | Метод середины квадрата | Метод открытой адресации |
| 8 | ФИО, №телефона, адрес | ФИО | Ключ умножить на 0.56, взять дробную часть, перевести её в целую, взять остаток от деления на размер таблицы. | Метод цепочек |
| 9 | ФИО, №телефона, адрес | Адрес | Остаток от деления на размер таблицы. | Метод цепочек |
| 10 | ФИО, №паспорта, №телефона | №телефона | Метод середины квадрата | Метод открытой адресации |
| 11 | ФИО, №паспорта, №телефона | №паспорта | Остаток от деления на размер таблицы. | Метод открытой адресации |
| 12 | ФИО, №паспорта, №телефона | ФИО | Ключ умножить на 0.46, взять дробную часть, перевести её в целую, взять остаток от деления на размер таблицы. | Метод цепочек |
| 13 | ФИО, дата\_рождения, адрес | дата\_рождения | Остаток от деления на размер таблицы. | Метод цепочек |
| 14 | ФИО, дата\_рождения, адрес | ФИО | Остаток от деления на размер таблицы. | Метод открытой адресации |
| 15 | ФИО, дата\_рождения, адрес | адрес | Метод середины квадрата | Метод открытой адресации |
| 16 | ФИО, дата\_рождения, №телефона | дата\_рождения | Ключ умножить на 0.99, взять дробную часть, перевести её в целую, взять остаток от деления на размер таблицы. | Метод цепочек |
| 17 | ФИО, дата\_рождения, №телефона | ФИО | Остаток от деления на размер таблицы. | Метод цепочек |
| 18 | ФИО, дата\_рождения, №телефона | №телефона | Ключ умножить на 0.41, взять дробную часть, перевести её в целую, взять остаток от деления на размер таблицы. | Метод открытой адресации |
| 19 | ФИО, дата\_рождения, №паспорта, | №паспорта, | Остаток от деления на размер таблицы. | Метод открытой адресации |
| 20 | ФИО, дата\_рождения, №паспорта, | дата\_рождения, | Ключ умножить на 0.68, взять дробную часть, перевести её в целую, взять остаток от деления на размер таблицы. | Метод цепочек |
| 21 | ФИО, №паспорта, №телефона, адрес | Адрес | Ключ умножить на 0.79, взять дробную часть, перевести её в целую, взять остаток от деления на размер таблицы. | Метод открытой адресации |
| 22 | ФИО, дата\_рождения, №паспорта, №телефона | №паспорта | Остаток от деления на размер таблицы. | Метод открытой адресации |
| 23 | ФИО, №счета, сумма, №паспорта | №паспорта | Ключ умножить на 0.59, взять дробную часть, перевести её в целую, взять остаток от деления на размер таблицы. | Метод цепочек |
| 24 | ФИО, дата\_рождения, №телефона, №счёта | №счёта | Метод середины квадрата | Метод открытой адресации |
| 25 | ФИО, группа, рейтинг, №телефона | №телефона | Остаток от деления на размер таблицы. | Метод открытой адресации |
| 26\* | ФИО, №счета, сумма, №паспорта | ФИО | Перевод из 10-чной СС в 12-ричную СС. | Метод цепочек |
| 27\* | ФИО, дата\_рождения, №телефона | ФИО | Перевод из 10-чной СС в 16-ричную СС. | Метод цепочек |
| 28\* | ФИО, группа, рейтинг | ФИО | Перевод из 10-чной СС в 15-ричную СС | Метод цепочек |