闭散列法

闭散列法简介

- 闭散列方法把所有记录直接存储在散列表中，如果发生冲突则根据某种程序员自己设定的方式继续进行探查

闭散列法实现

- 模板代码

```
const int SIZE = 360007;  // SIZE 是最大可以存储的元素
class Hash
{
    int keys[SIZE];
    int values[SIZE];

    int& operator[](int n)
    {
        // 返回一个指向对应 Hash[Key] 的引用
        // 修改成不为 0 的值 0 时候视为空
        int idx = (n % SIZE + SIZE) % SIZE, cnt = 1;
        while (keys[idx] != n && values[idx] != 0)   // 发生哈希冲突
        {
            // 二次探测法：
            // 当有哈希冲突时，寻找下一个空闲位置时，首先在该位置处加1的平方，若加1的平方的位置处依然有元素，那就加2的平方，知道找到一个空闲的位置为止
            idx = (idx + cnt * cnt) % SIZE;
            cnt += 1;
        }
        keys[idx] = n;
        return values[idx];
    }
};
```

哈希表解决哈希冲突应用示例

题目描述

如题，给定 N 个字符串（第 i 个字符串长度为 M_i，字符串内包含数字、大小写字母，大小写敏感），请求出 N 个字符串中共有多少个不同的字符串。

- 输入格式

第一行包含一个整数 N，为字符串的个数。

接下来 N 行每行包含一个字符串，为所提供的字符串。

- 输出格式

输出包含一行，包含一个整数，为不同的字符串个数。

题目分析与算法设计

本题可以使用各种方法求解，但是考虑时间限制，使用哈希表能够大幅降低时间花销

只要设计特殊的转换函数，将字符串哈希成一个单独的键值，并保证键和字符串之间是一个一一对应函数，就可以通过哈希表的直接对应优势节约搜索时间

如果出现哈希冲突，再设置方法解决哈希冲突即可

程序清单

- 解题代码见`coding/code3.cpp`