实验报告

题目：6.2

班级：信卓1901班 姓名：刘可立 学号：U201913498

一、需求分析

1. 将全班30名同学的名字以汉语拼音形式填入哈希表。
2. 设计合适的Hash函数使得平均查找长度小于2。
3. 哈希函数用除留余数法构造，用伪随机再探测散列法处理冲突。
4. 测试数据：

30个姓名的拼音

二、概要设计

为实现以上需求，需要使用哈希表抽象数据类型：

1. 所使用的哈希表的抽象数据结构定义如下：

ADT HashTable{

数据对象：D = {ai | ai ∈ CharacterSet, i = 1, 2, ..., n, n>=0}

数据关系：R = {< ai-1 , ai > | ai-1 , ai ∈ D, i = 2, ..., n}

基本操作：

InitHashTable(&T)

初始化哈希表

Insert(&T, s)

将字符串s插入到hash表的适当位置

Hash(n)

哈希函数，确定数值与存储位置的关系

Str2int(s, &n)

按一定规则将字符串化为一个整数，方便执行Hash函数

Search(s, &n)

搜索字符串s，返回其位置序号

若不存在则返回-1

Collision(&n)

处理冲突

}

1. 本程序包含两个模块：
2. 主程序模块：

int main(int argc, const char \* argv[]) {

初始化哈希表

将实现储存的30个姓名插入到哈希表

输出平均查找长度

输出Hash表

while (用户输入 != EOF) {

获取用户输入并搜索

}

}

1. 实现哈希表的抽象数据类型

三、详细设计

1. 主程序需要的全程量

//记录查找长度

**int** cnt = 0;

//姓名集

**char** \* names[] = {"WangHaoPeng", "DuWei", "GaoPeiLi", "HuYaoZhou", "Xiangjunjie", "XuNuo", "YangTianCheng", "ZhaiYuQi", "ChenQiuHang", "ChenXuanChi", "DuanKangSheng", "GuoXiangMin", "HeChengTian", "HuangJinDiao", "KangTongJia", "LiuKeLi", "YangRuoHai", "LiChunJian", "LiMengHan", "LiZhuoFan", "WangLi", "XiaChen", "XiongQiLong", "YinShiQiu", "YuanXinHe", "YuXiaoYu", "ZhangChaoHui", "ZhangXinYu", "ZhuYiQing", "ZengYangZhu"};

//哈希表抽象数据结构

**class** HashTable {

**public**:

**char** names[41][15];

//成员函数...

}

1. 哈希表抽象数据结构的基本操作伪代码如下：

HashTable(){ //初始化函数

将每个串置为空

}

**void** insert(**char** \* str){

i = hash(str2int(str));

**if** (第i个串为空){

将str拷贝到第i个串位置;

} **else** {

**while** (第i个串不为空) {

i = collision(i); //冲突处理

}

将str拷贝到第i个串位置;

}

}

**int** hash(**int** k){

返回 k mod 41

}

**int** str2int(**char** \* str){

将名字奇数位置上的字符相加并返回

}

**int** search(**char** \* str){

i = hash(str2int(str));

**if** (第i个位置上与str内容相同 && ++比较次数){

**return** i

} **else** {

**while** (第i个位置上与str内容不相同 && ++比较次数) {

**if** (第i个串为空) {

**return** -1;

}

collision(i);

}

}

**return** i;

}

**int** collision(**int** i){

r = 伪随机序列的下一个数

i = (i + r + 1) % 41

}

1. 函数调用关系

主函数调用：insert(), search()函数

Insert(), search()函数调用hash(), collision()函数

四、调试分析

1. 对设计过程总结反思：
2. 一共有30个人名。考虑到：（1）必须要做到尽可能地减小哈希冲突；（2）也要尽可能地不浪费空间；（3）当一个数除以一个素数的时候，会产生最分散的余数；（4）装填因子设置在 0.7 ~ 0.8。

将Hash表大小设置为 41。

1. 为比较不同的hash函数的效率，设置了全局变量cnt。每当比较一次就同步的把cnt自增一次。程序把所有名字搜索一次，就得到了总的比较次数。再除以30就得到了ASL。
2. 主要算法的时空分析
3. 平均查找长度

本程序的平均查找长度为1.633333，最小查找长度1，最大为6。

装填因子 = 30 / 41 = 0.732

构造的哈希表如下：

0 : YangRuoHai

1 : HuYaoZhou

2 : XiaChen

3 : YuXiaoYu

4 :

5 : ZhangXinYu

6 : XiongQiLong

7 : ZhangChaoHui

8 :

9 : HuangJinDiao

10 : WangHaoPeng

11 :

12 : LiZhuoFan

13 : YinShiQiu

14 : DuWei

15 : GuoXiangMin

16 :

17 : KangTongJia

18 :

19 :

20 :

21 : LiChunJian

22 : DuanKangSheng

23 :

24 :

25 : HeChengTian

26 :

27 : WangLi

28 : ZhuYiQing

29 : ZhaiYuQi

30 : LiuKeLi

31 : GaoPeiLi

32 : ZengYangZhu

33 : XuNuo

34 : YuanXinHe

35 : LiMengHan

36 : YangTianCheng

37 : ChenQiuHang

38 : Xiangjunjie

39 : ChenXuanChi

40 :

五、用户手册

1. 本程序的测试运行环境为MacOS，在 Terminal 中定位到 a.out 所在的文件夹，输入 ./a.out 回车运行
2. 键入待搜索的人名（拼音）。注意每一个字的首字母应大写，如：LiuKeLi
3. 输入 q 可结束运行
4. 运行程序后的用户界面：

手机屏幕截图

描述已自动生成

六、测试结果

1. 构造的哈希表见上文。
2. 本程序的平均查找长度为1.633333，最小查找长度1，最大为6。

装填因子 = 30 / 41 = 0.732

七、附录

清单：

/ HashTable.xcodeproj 项目文件

/ HashTable/main.cpp 主程序

/ HashTable/Header.h 头文件

/ HashTable/a.out 可执行文件