数据结构上机报告

题目：哈希表设计

班级：04班 姓名： 学号： 完成日期：2020.12.20

一、需求分析

1.问题描述

针对某个集体（比如你所在的班级）中的“人名”设计一个哈希表，使得平均查找长度均不超过 R ，完成相应的建表和查表顺序。

2.基本要求

假设人名为中国人姓名的汉语拼音形式。待填入哈希表的人名共有 30 个，取平均查找长度的上限为2。哈希函数用除留余数法构造，用伪随机探测再散列法处理冲突。

1. 预设条件

如果随机数自行构造，则应首先调整好随机函数，使其分布均匀。人名的长度均不超过 19 个字符（最长的人名如：庄双双（ Zhuang Shuangshuang ））。字符的取码方法可直接利用 C 语言中的 toascii 函数，并可先对过长的人名先作折叠处理。

二、概要设计:

程序用结构体类型存储哈希表，先求出伪随机数列作为对应的散列函数参数，并根据用户的相应选择实现相应操作。

1. 哈希表的定义和相关操作

ADT Hash {

数据对象 D ： D 是具有相同特征的数据元素的集合。各数据元素均含有类型相同，可唯一标识数据元素的关键字。

数据关系 R ：数据元素同属一个集合。

InitNameTable()

操作结果： 初始化姓名表 。

CreateHashTable()

操作结果： 建立哈希表 。

DisplayNameTable()

操作结果： 显示姓名表 。

DisplayHashTable()

操作结果： 显示哈希表 。

FindName()

操作结果： 查找姓名 。

}ADT Hash

三、详细设计

各操作实现函数

void CreateHashTable() {

// 建立哈希表

for(i=0; i<HASH\_LEN; i++) {

HashTable[i].py="\0";

HashTable[i].m =0;

HashTable[i].si=0;

}

for(i=0; i<NAME\_LEN; i++) {

int sum=1,j=0;

int adr=(NameTable[i].m)%P; // 除留余数法 H(key)=key MOD p,p<=m

if(HashTable[adr].si==0) { // 如果不冲突，将姓名表赋值给哈希表

HashTable[adr].m =NameTable[i].m;

HashTable[adr].py=NameTable[i].py;

HashTable[adr].si=1;

} else { // 如果冲突

while(HashTable[adr].si!=0) {

adr=(adr+d[j++])%HASH\_LEN; // 伪随机探测再散列法处理冲突

sum=sum+1; // 查找次数加 1

}

HashTable[adr].m =NameTable[i].m; // 将姓名表复制给哈希表对应的位置上

HashTable[adr].py=NameTable[i].py;

HashTable[adr].si=sum;

}

}

}

void DisplayNameTable() {

// 显示姓名表

printf("\n 地址 \t\t 姓名 \t\t 关键字 \n");

for (i=0; i<NAME\_LEN; i++)

printf("%2d %18s \t\t %d \n",i,NameTable[i].py,NameTable[i].m);

}

void DisplayHashTable() {

// 显示哈希表

float asl=0.0;

printf("\n\n 地址 \t\t 姓名 \t\t 关键字 \t 搜索长度 \n"); // 显示的格式

for (i=0; i<HASH\_LEN; i++) {

printf("%2d %18s \t\t %d \t\t %d\n",i,HashTable[i].py,HashTable[i].m,HashTable[i].si);

asl+=HashTable[i].si;

}

asl/=NAME\_LEN; // 求得 ASL

printf("\n\n 平均查找长度： ASL(%d)=%f \n",NAME\_LEN,asl);

}

void FindName() {

// 查找

char name[20]= {0};

int s=0,sum=1,adr;

printf("\n 请输入想要查找的姓名的拼音 :");

scanf("%s",name);

getchar();

for (j=0; j<20; j++) // 求出姓名的拼音所对应的 ASCII 作为关键字

s+=toascii(name[j]);

adr=s%P; // 除留余数法

j=0;

if(HashTable[adr].m==s&&!strcmp(HashTable[adr].py,name)) // 分 3 种情况进行判断，并输出超找结果

printf("\n 姓名 :%s 关键字 :%d 查找长度为 : 1\n",HashTable[adr].py,s);

else if (HashTable[adr].m==0)

printf(" 没有想要查找的人 !\n");

else {

while(1) {

adr=(adr+d[j++])%HASH\_LEN; // 伪随机探测再散列法处理冲突

sum=sum+1; // 查找次数加 1

if(HashTable[adr].m==0) {

printf(" 没有想要查找的人 !\n");

break;

}

if(HashTable[adr].m==s&&!strcmp(HashTable[adr].py,name)) {

printf("\n 姓名 :%s 关键字 :%d 查找长度为 :%d\n",HashTable[adr].py,s,sum);

break;

}

}

}

}

四、调试分析

1、调试问题

（1）人名的关键字出现重合现象，需要对散列函数做出改造

（2）初期实现时求ASL算法未考虑特殊情况，后经过改进算法解决了问题。

1. 算法改进设想

目前程序时间复杂度较低，但存在交互性较差等问题，后期可通过完善图形化界面，增强程序健壮性，优化基本操作算法提高程序的可用性。

1. 经验和体会

熟悉了哈希表的建立与基本操作，学会了对散列函数出现重复性的处理方法，以及数据的基本存储知识。

五、用户使用说明

根据图形界面的输入提示进行相应操作即可。

六、测试结果

输入样例

请选择 :B

请选择 :C

输出样例：

地址 姓名 关键字 搜索长度

0 0 0

1 0 0

2 yuhaiyuan 989 1

3 hanle 520 1

4 0 0

5 songxiaofei 1180 1

6 0 0

7 0 0

8 0 0

9 nieyanshun 1090 1

10 xuyang 668 1

11 0 0

12 sunyingjie 1093 1

13 hujiaming 956 2

14 dongshuliang 1283 1

15 0 0

16 dutongtong 1097 1

17 qiuhouyang 1098 1

18 0 0

19 zhaoqing 865 1

20 xunwenwen 1007 1

21 zhangxu 773 1

22 0 0

23 sunhaohao 966 2

24 wangbo 638 2

25 shuxiang 871 1

26 caowukui 872 1

27 langzhijie 1061 1

28 0 0

29 0 0

30 haopengcheng 1255 2

31 0 0

32 0 0

33 wangbaoshan 1161 1

34 duhuan 645 1

35 fanqing 740 1

36 0 0

37 0 0

38 shanjianfeng 1260 1

39 shenjinfeng 1167 1

40 chenhongcong 1265 2

41 lvdelu 652 1

42 jiangkaiqiang 1358 1

43 houfeng 748 1

44 0 0

45 0 0

46 xuhui 563 1

47 zhangyanyan 1192 3

48 0 0

49 0 0

平均查找长度： ASL(30)=1.233333

请输入想要查找的姓名的拼音 :lvdelu

姓名 :lvdelu 关键字 :652 查找长度为 : 1

1. 附录

#include<stdio.h>

#include<time.h> //time 用到的头文件

#include<stdlib.h> // 随机数用到的头文件

#include<ctype.h> //toascii() 用到的头文件

#include<string.h> // 查找姓名时比较用的头文件

#define HASH\_LEN 50 // 哈希表的长度

#define P 47 // 小于哈希表长度的 P

#define NAME\_LEN 30 // 姓名表的长度

typedef struct {

// 姓名表

char \*py; // 名字的拼音

int m; // 拼音所对应的

} NAME;

NAME NameTable[HASH\_LEN]; // 全局定义姓名表

typedef struct {

// 哈希表

char \*py; // 名字的拼音

int m; // 拼音所对应的 ASCII 总和

int si; // 查找长度

} HASH;

HASH HashTable[HASH\_LEN]; // 全局定义哈希表

int d[30],i,j; // 全局定义随机数 , 循环用的 i 、 j

void InitNameTable() {

// 姓名表的初始化

NameTable[0].py="caowukui";

NameTable[1].py="langzhijie";

NameTable[2].py="dongshuliang";

NameTable[3].py="qiuhouyang";

NameTable[4].py="zhangxu";

NameTable[5].py="duhuan";

NameTable[6].py="fanqing";

NameTable[7].py="songxiaofei";

NameTable[8].py="dutongtong";

NameTable[9].py="sunhaohao";

NameTable[10].py="shanjianfeng";

NameTable[11].py="wangbaoshan";

NameTable[12].py="houfeng";

NameTable[13].py="hujiaming";

NameTable[14].py="jiangkaiqiang";

NameTable[15].py="xuyang";

NameTable[16].py="lvdelu";

NameTable[17].py="shenjinfeng";

NameTable[18].py="xuhui";

NameTable[19].py="hanle";

NameTable[20].py="xunwenwen";

NameTable[21].py="chenhongcong";

NameTable[22].py="zhangyanyan";

NameTable[23].py="nieyanshun";

NameTable[24].py="haopengcheng";

NameTable[25].py="yuhaiyuan";

NameTable[26].py="shuxiang";

NameTable[27].py="sunyingjie";

NameTable[28].py="wangbo";

NameTable[29].py="zhaoqing";

NameTable[30].py="zhangshengqian";

for (i=0; i<NAME\_LEN; i++) { // 将字符串的各个字符所对应的 ASCII 码相加，所得的整数做为哈希表的关键字

int s=0;

char \*p=NameTable[i].py;

for (j=0; \*(p+j)!='\0'; j++)

s+=toascii(\*(p+j));

NameTable[i].m=s;

}

}

void CreateHashTable() {

// 建立哈希表

for(i=0; i<HASH\_LEN; i++) {

HashTable[i].py="\0";

HashTable[i].m =0;

HashTable[i].si=0;

}

for(i=0; i<NAME\_LEN; i++) {

int sum=1,j=0;

int adr=(NameTable[i].m)%P; // 除留余数法 H(key)=key MOD p,p<=m

if(HashTable[adr].si==0) { // 如果不冲突，将姓名表赋值给哈希表

HashTable[adr].m =NameTable[i].m;

HashTable[adr].py=NameTable[i].py;

HashTable[adr].si=1;

} else { // 如果冲突

while(HashTable[adr].si!=0) {

adr=(adr+d[j++])%HASH\_LEN; // 伪随机探测再散列法处理冲突

sum=sum+1; // 查找次数加 1

}

HashTable[adr].m =NameTable[i].m; // 将姓名表复制给哈希表对应的位置上

HashTable[adr].py=NameTable[i].py;

HashTable[adr].si=sum;

}

}

}

void DisplayNameTable() {

// 显示姓名表

printf("\n 地址 \t\t 姓名 \t\t 关键字 \n");

for (i=0; i<NAME\_LEN; i++)

printf("%2d %18s \t\t %d \n",i,NameTable[i].py,NameTable[i].m);

}

void DisplayHashTable() {

// 显示哈希表

float asl=0.0;

printf("\n\n 地址 \t\t 姓名 \t\t 关键字 \t 搜索长度 \n"); // 显示的格式

for (i=0; i<HASH\_LEN; i++) {

printf("%2d %18s \t\t %d \t\t %d\n",i,HashTable[i].py,HashTable[i].m,HashTable[i].si);

asl+=HashTable[i].si;

}

asl/=NAME\_LEN; // 求得 ASL

printf("\n\n 平均查找长度： ASL(%d)=%f \n",NAME\_LEN,asl);

}

void FindName() {

// 查找

char name[20]= {0};

int s=0,sum=1,adr;

printf("\n 请输入想要查找的姓名的拼音 :");

scanf("%s",name);

getchar();

for (j=0; j<20; j++) // 求出姓名的拼音所对应的 ASCII 作为关键字

s+=toascii(name[j]);

adr=s%P; // 除留余数法

j=0;

if(HashTable[adr].m==s&&!strcmp(HashTable[adr].py,name)) // 分 3 种情况进行判断，并输出超找结果

printf("\n 姓名 :%s 关键字 :%d 查找长度为 : 1\n",HashTable[adr].py,s);

else if (HashTable[adr].m==0)

printf(" 没有想要查找的人 !\n");

else {

while(1) {

adr=(adr+d[j++])%HASH\_LEN; // 伪随机探测再散列法处理冲突

sum=sum+1; // 查找次数加 1

if(HashTable[adr].m==0) {

printf(" 没有想要查找的人 !\n");

break;

}

if(HashTable[adr].m==s&&!strcmp(HashTable[adr].py,name)) {

printf("\n 姓名 :%s 关键字 :%d 查找长度为 :%d\n",HashTable[adr].py,s,sum);

break;

}

}

}

}

int main() {

// 主函数

char c;

int a=1;

srand((int)time(0));

for(i=0; i<30; i++) // 用随机函数求得伪随机数列 d[i] （在 1 到 50 之间）

d[i]=1+(int)(HASH\_LEN\*rand()/(RAND\_MAX+1.0));

InitNameTable();

CreateHashTable();

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\* \*\n");

printf("\* 哈希表设计 \*\n");

printf("\* \*\n");

printf("\* A: 显示姓名表 B: 显示哈希表 \*\n");

printf("\* \*\n");

printf("\* C: 查找 D: 退出 \*\n");

printf("\* \*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

while(a) {

printf("\n 请选择 :");

scanf("%c",&c);

getchar();

switch(c) { // 根据选择进行判断，直到选择退出时才可以退出

case 'A':

case 'a':

DisplayNameTable();

break;

case 'B':

case 'b':

DisplayHashTable();

break;

case 'C':

case 'c':

FindName();

break;

case 'D':

case 'd':

a=0;

break;

default:

printf("\n 请输入正确的选择 !\n");

break;

}

}

return 0;

}