**实验报告：Hash表**

**实验题目：**

Hash表

**实验要求：**

1. 输入由n个班级同学拼音组成的文件，建立对应的Hash表
2. 计算平均查找次数

**编程思路：**

1. 利用字符的ASCII码值和平方取中法生成Hash值
2. 运用闭散列的方法解决Hash值的冲突问题

**核心代码：**

1. 利用字符的ASCII码值和平方取中法生成Hash值

**unsigned long int SquMid(int a){//平方取中法，将a平方直到5位数以上，然后取中**

if(a==1||a==0) return SquMid(a+1);

if(a<10000) return SquMid(a\*a);

return (a/100)%M;

}

**int Hash(char \*s){//将单词转为hash值，首先将单词字母的ASCII码相加，然后平方取中**

long int sum=0;

for(int i=0;s[i]!='\0';i++){

sum+=s[i];

}

return SquMid(sum);

}

1. 运用闭散列的方法解决Hash值的冲突问题

**void hashlist\_generate(LIST Wordlist,char Hashlist[M][MAX\_WORD\_SIZE]){//单词表转Hash表**

for(int i=0;i<M;i++){//初始化清空

Hashlist[i][0]='\0';

}

for(int i=0;i<Wordlist.listnum;i++){

int hash=Hash(Wordlist.list[i]);**//首先计算初始Hash值**

while(Hashlist[hash][0]!='\0'&&strcmp(Hashlist[hash],Wordlist.list[i])!=0)**{//当hash值冲突时，使用闭散列方法：hash++**

**hash++;**

hash=hash%M;

}

strcpy(Hashlist[hash],Wordlist.list[i]);//将单词复制到对应的hash值上

}

}

1. 统计平均查找次数

double Search(LIST Wordlist,char Hashlist[M][MAX\_WORD\_SIZE]){//平均查找次数

int Num=0;//统计查找次数

for(int i=0;i<Wordlist.listnum;i++){

int hash=Hash(Wordlist.list[i]);

int CompareNum=0;//比较次数，防止死循环

Num++;

while(strcmp(Wordlist.list[i],Hashlist[hash])!=0&&CompareNum<M){**//开始比较，以闭散列的方式查找**

**hash++;**

hash=hash%M;

Num++;

CompareNum++;

}

}

return ((double)Num/(double)Wordlist.listnum);

}

**实验结果：**

**Text.txt文件内容：（52个单词）**

q w e r t y u i o p a s d f g h j k l z x c v b n m qq ww ee rr tt yy uu ii oo pp aa ss dd ff gg hh jj kk ll zz xx cc vv bb nn mm

**输出内容：**

1. **M=1.05N**

平均查找次数为：2.557692

1. **M=1.1N**

平均查找次数为：2.403846

1. **M=1.15N**

平均查找次数为：2.384615

可以看出，随着M的增大，平均查找次数有所减小

**源码全文：**

#include<stdio.h>

#include<string.h>

#include<ctype.h>

#define MAX\_TEXT\_SIZE 2000

#define MAX\_WORD\_NUM 1000

#define MAX\_WORD\_SIZE 20

#define N 50

#define M (int)(1.15\*N)

typedef struct{//单词表，记录内容和单词个数

char list[MAX\_WORD\_NUM][MAX\_WORD\_SIZE];

int listnum;

}LIST;

void input\_text(char \*Text);//从文件读取文本，输入到字符串

void wordlist\_generate(char \*Text,LIST &Wordlist);//字符串转单词表

unsigned long int SquMid(int a);//平方取中法

int Hash(char \*s);//将单词转为hash值

void hashlist\_generate(LIST Wordlist,char Hashlist[M][MAX\_WORD\_SIZE]);//单词表转Hash表

double Search(LIST Wordlist,char Hashlist[M][MAX\_WORD\_SIZE]);//平均查找次数

int main(){

char text[MAX\_TEXT\_SIZE];

LIST wordlist;

char hashlist[M][MAX\_WORD\_SIZE];

input\_text(text);

wordlist\_generate(text,wordlist);

hashlist\_generate(wordlist,hashlist);

printf("平均查找次数为：%lf",Search(wordlist,hashlist));

return 0;

}

void input\_text(char \*Text){//从文件读取文本，输入到字符串

FILE \*read;

read=fopen("Text.txt","r");

fread(Text,sizeof(char),MAX\_TEXT\_SIZE,read);

fgets(Text,MAX\_TEXT\_SIZE,read);

fclose(read);

return;

}

void wordlist\_generate(char \*Text,LIST &Wordlist){//字符串转单词表

int last=0,next=0;//last为上一个字符是否为字母，next为下一个字符是否为字母

int n=0;//正在写入的第n个单词

int m=0;//正在写入单词的第m个字母

Wordlist.listnum=0;

for(int i=0;Text[i]!='\0';i++){

if(islower(Text[i])||isupper(Text[i])){//next为下一个字符是否为字母

next=1;

}

else next=0;

//四种状态，对应四种操作

if(last==0&&next==0){//单词未开始

}

else if(last==0&&next==1){//单词开始

Wordlist.list[n][m]=Text[i];//写入

m++;//后移

Wordlist.listnum++;//单词数+1

}

else if(last==1&&next==1){//单词未结束

Wordlist.list[n][m]=Text[i];//写入

m++;//后移

}

else if(last==1&&next==0){//单词结束

Wordlist.list[n][m]='\0';//收尾

n++;m=0;

}

//改变状态

last=next;

}

Wordlist.list[n][m]='\0';

}

unsigned long int SquMid(int a){//平方取中法

if(a==1||a==0) return SquMid(a+1);

if(a<10000) return SquMid(a\*a);

return (a/100)%M;

}

int Hash(char \*s){//将单词转为hash值

long int sum=0;

for(int i=0;s[i]!='\0';i++){

sum+=s[i];

}

return SquMid(sum);

}

void hashlist\_generate(LIST Wordlist,char Hashlist[M][MAX\_WORD\_SIZE]){//单词表转Hash表

for(int i=0;i<M;i++){//初始化清空

Hashlist[i][0]='\0';

}

for(int i=0;i<Wordlist.listnum;i++){

int hash=Hash(Wordlist.list[i]);

while(Hashlist[hash][0]!='\0'&&strcmp(Hashlist[hash],Wordlist.list[i])!=0){//当hash值冲突时，使用闭散列方法：hash++

hash++;

hash=hash%M;

}

strcpy(Hashlist[hash],Wordlist.list[i]);//将单词复制到对应的hash值上

}

}

double Search(LIST Wordlist,char Hashlist[M][MAX\_WORD\_SIZE]){//平均查找次数

int Num=0;//统计查找次数

for(int i=0;i<Wordlist.listnum;i++){

int hash=Hash(Wordlist.list[i]);

int CompareNum=0;//比较次数，防止死循环

Num++;

while(strcmp(Wordlist.list[i],Hashlist[hash])!=0&&CompareNum<M){//开始比较，以闭散列的方式查找

hash++;

hash=hash%M;

Num++;

CompareNum++;

}

}

return ((double)Num/(double)Wordlist.listnum);

}