．哈夫曼编码与译码 ①统计一个文本文件的各个字符出现的频次 ②依此文本出现的字符和出现的频次，建立该文本字符集的哈夫曼编码表，以 文件形式存储并显示。 ③给出该字符集的任意文本文件的哈夫曼编码文件。 ④对哈夫曼编码文件进行译码。 ⑤显示相应的结果。

#include<stdio.h>

#include<string>

#include<iostream>

#include<fstream>

using namespace std;

struct huffman

{

int root;

int right;

int left;

int weight;

int word;

}tree[513]; //哈夫曼树的结构体，root表示其上一个节点，right，left是左右儿子的标号，weight表示权重，word代表该节点代表的字符（如果有的话）

int point;

int number[256]; //统计每个字符出现的次数

int code[256][100]; //存储每个字符的编码

int codelen[256]; //存储每个

void clear() //清空数组和哈夫曼树

{

for (int i=0;i<=512;i++)

tree[i].root=tree[i].right=tree[i].left=tree[i].weight=tree[i].word=-1;

point=0;

memset(number,0,sizeof(number));

memset(code,0,sizeof(code));

memset(codelen,0,sizeof(codelen));

return;

}

void makenode(int weight,int word) //生成一个新的节点

{

point++;

tree[point].root=-1;

tree[point].right=tree[point].left=-1;

tree[point].weight=weight;

tree[point].word=word;

return;

}

void insert(int weight,int p1,int p2) //插入操作

{

point++;

tree[point].root=-1;

tree[point].right=p2;

tree[point].left=p1;

tree[point].weight=weight;

tree[point].word=-1;

tree[p1].root=tree[p2].root=point;

return ;

}

void makecode(int po, string cod) //通过DFS为每一个字符编码

{

if (tree[po].word!=-1)

{

for (int i=0;i<cod.length();i++)

code[tree[po].word][i]=cod[i]-'0';

codelen[tree[po].word]=cod.length();

}

else

{

makecode(tree[po].left,cod+'0');

makecode(tree[po].right,cod+'1');

}

return;

}

main()

{

int i,j,k,l,w; // 循环变量，无意义。

char text[10001]; //用于存储用于统计的文件

int min1,min2,p1,p2;// 构造哈夫曼树时用于表示最小的2个权值的大小和序号

char name[101]; //输入的文件名

char name1[101]; //输入的文件名

int tot; //需要统计的文件的长度

int list[512]; // 翻译时 用于临时存储一小段哈夫曼编码

int tra[10001]; //存储需要翻译的文件

int top=0,flag=0; // top为list中的哈夫曼编码长度 flag标记是否找到相对应的字符

int tot1=0;// 需要翻译的文件的长度

lab:

printf("请输入你需要统计的文件名:\n");

scanf("%s",name);

ifstream in(name);

clear();

tot=0;

while (in.get(text[tot]))

{

number[text[tot]]++;

tot++;

} //从文件中读取字符，保存到text中

for (i=0;i<=255;i++)

if (number[i]!=0)

makenode(number[i],i); //生成节点

int tp=point;

for (i=tp+1;i<=2\*tp-1;i++)

{

min1=min2=1<<20;

for (j=1;j<=i-1;j++)

{

if (tree[j].root==-1)

{

if (tree[j].weight<min1)

{

min1=tree[j].weight;

p1=j;

}

else if (tree[j].weight<min2)

{

min2=tree[j].weight;

p2=j;

}

}

}

insert(min1+min2,p1,p2); // 生成哈夫曼树

}

string cod="";

makecode(2\*tp-1,cod); //编码

in.close();

printf("统计完成！\n\n");

printf("请输入一个文件名，程序将会把原始文件的信息写入该文件中:\n");

scanf("%s",name);

ofstream out(name);

out<<"哈夫曼编码表："<<endl; //输出所统计的文件的相关信息

for (i=0;i<=511;i++)

if (number[i]!=0)

{

if (i!='\n')

out<<" "<<(char)i<<" :";

else out<<"\\n :";

for (j=0;j<codelen[i];j++)

out<<code[i][j];

out<<" 所占百分比为: "<<(float) number[i]/tot\*100<<"%";

out<<endl;

}

out<<endl<<"正文:"<<endl;

for (k=0;k<tot;k++)

{

for (w=0;w<codelen[text[k]];w++)

out<<code[text[k]][w];

}

out<<endl;

out.close();

printf("输出完成！\n\n");

lab2:

printf("1: 统计其他文件，清空当前结果\n");

printf("2: 根据统计结果，解析其他文件 \n");

printf("3: 退出\n");

printf("\n请输入指令:");

int num;

lab1:

scanf("%d",&num);

if (num==1) //回到程序开始

goto lab;

else if (num==3) //程序结束

{

printf("\n程序结束\n");

return 0;

}

else if (num==2) //读入文件进行翻译

{

printf("\n请输入需要翻译的文件名:");

scanf("%s",name);

ifstream in(name);

char temp;

tot1=top=0;

memset(list,-1,sizeof(list));

memset(tra,-1,sizeof(tra));

while (in.get(temp)) //读取需要翻译的文件

{

tra[tot1]=temp-'0';

tot1++;

}

printf("文件读取完成，请输入保存翻译结果的文件名:");

scanf("%s",name1);

ofstream out(name1);

for (k=0;k<tot1;k++) //每读入一个字符，就加在list中最后一个字符的后面，并所有的哈夫曼编码对比

{

list[top]=tra[k];

if (tra[k]!=0&&tra[k]!=1)

{

printf("文件中有非法字符，翻译终止！\n");

out<<"文件中有非法字符，翻译终止！\n";

in.close();

out.close();

goto lab2;

}

top++;

for (i=0;i<256;i++)

if (number[i]!=0)

{

flag=1;

for (j=0;j<codelen[i];j++)

{

if (code[i][j]!=list[j])

{

flag=0;

break;

}

}

if (flag==1)

{

out<<(char)i;

memset(list,-1,sizeof(list));

top=0;

break;

}

}

}

printf("翻译完成\n\n");

in.close();

out.close();

goto lab2;

}

else //异常处理

{

printf("\n错误的指令！请重新输入:");

goto lab1;

}

return 0;

}