**实验报告**

题目：哈夫曼编译器

班级：电信1809 姓名：刘俊逸 学号：U201813492 完成日期：2019.4.23

1. 需求分析
2. 问题描述：利用赫夫曼编码进行通信可以大大提高信道利用率，缩短信息传输时间，降低传输成本。但是，这要求在发送端通过一个编码系统对待传数据预先编码，在接收端讲传来的数据进行译码（复原）。对于双工信道（即可以双向传输信息的信道），每端都需要一个完整的编/译码系统。
3. 基本要求

（1） I:初始化（Initialization）。从终端读入字符集大小n，以及n个字符和n个权值，建立赫夫曼树，并将它存于文件hfmTree中。

（2） E：编码（Encoding）。利用已建好的赫夫曼树，对文件ToBeTran中的正文进行编码，然后将结果存入文件CodeFile中。

（3） D:译码（Decoding）。利用已建好的赫夫曼树将文件CodeFile中的代码进行译码，结果存入文件TextFile中。

（4） P:印代码文件（Print）。将文件CodeFile以紧凑格式显示在终端上，每行50个代码。同时将此字符形式的编码文件写入文件CodePrin中。

（5） T:印赫夫曼树（TreePrinting）。将已在内存中的赫夫曼树以直观的方式（树或凹入表形式）显示在终端上，同时将此字符形式的赫夫曼树写入文件TreePrint中。

3. 实现提示

（1）   编码结果以文本方式存储在文件CodeFile中。

（2）   用户界面可以设计为“菜单”方式：显示上述功能符号，再加上“Q”，表示退出运行Quit。请用户键入一个选择功能符。此功能执行完毕后再显示此菜单，直至某次用户选择了“Q”为止。

二．概要设计

1）在分析题目要求的基础上，我首先设计了存储结构，定义如下

typedef struct{

int weight;

char ch;

int parent,lchild,rchild;

}HTNode,\*HuffmanTree;//节点信息

typedef char \*\*HuffmanCode; //存储哈夫曼编码

全局变量

HuffmanTree HT;

int n=0;

HuffmanCode HC;

int \*w;

char \*ch;

（2）基本操作：

void welcome(); //操作选择界面

void select(HuffmanTree HT,int j,int &s1,int &s2); //选择parent为0且weight最小的两个结点

void Init(); //输入字符及其权值，建立哈夫曼树

void Encoding(); //编码

void Decoding(); //译码

void Tree\_printing(); //打印哈夫曼树

void Print();//打印代码

void HuffmanCoding(HuffmanTree&HT,HuffmanCode &HC,int \*w,int n,char\*ch);

void Free();//释放动态分配的空间

1. 详细设计

#include<stdio.h>

#include<stdlib.h>

#define N\_LEAVE 26

#define N\_NODE (26\*2-1)

typedef struct \_Node

{

char character;

int weight;

int lchild;

int rchild;

int parent;

}Node,\*pNode;

typedef struct \_Code

{

int HufCode[N\_LEAVE];

int Start;

char Char;

}Code,\*pCode;

void Huffman(Node Ht[],int Wt[])

{

int i,j,x1,x2;

int min1,min2;

for(i=0;i<N\_NODE;i++)

{

Ht[i].parent = -1;

Ht[i].lchild = -1;

Ht[i].rchild = -1;

if(i<N\_LEAVE)

{

Ht[i].weight = Wt[i];

Ht[i].character = i+65;

}

else

{

Ht[i].weight = 0;

Ht[i].character = '?';

}

}

for(i=1;i<=N\_LEAVE-1;i++)

{

min1 = min2 = 1000;

x1 = x2 = 0;

for(j=0;j<N\_LEAVE-1+i;j++)

{

if(Ht[j].parent == -1 && Ht[j].weight<min1 )

{

min2 = min1;

x2 = x1;

min1 = Ht[j].weight;

x1 = j;

}

else

{

if( Ht[j].parent == -1 && Ht[j].weight<min2 )

{

min2 = Ht[j].weight;

x2 = j;

}

}

}

Ht[x1].parent = N\_LEAVE-1+i;

Ht[x2].parent = N\_LEAVE-1+i;

Ht[N\_LEAVE-1+i].weight = Ht[x1].weight + Ht[x2].weight;

Ht[N\_LEAVE-1+i].lchild = x1;

Ht[N\_LEAVE-1+i].rchild = x2;

}

}

void Code\_Ht(Node Ht[],Code Hc[])

{

int i,d,p,j;

Code x;

for(i=0;i<N\_LEAVE;i++)

{

x.Char = Ht[i].character;

x.Start = N\_LEAVE-1;

d = i;

p = Ht[i].parent;

while(1)

{

if(Ht[p].lchild == d)

x.HufCode[x.Start] = 0;

else if(Ht[p].rchild == d)

x.HufCode[x.Start] = 1;

else

printf("ERROR!");

d = p;

p = Ht[d].parent;

if(p == -1) break;

x.Start--;

}

for(j=x.Start;j<=N\_LEAVE-1;j++)

{

Hc[i].HufCode[j] = x.HufCode[j];

}

Hc[i].Start = x.Start;

Hc[i].Char = x.Char;

}

}

/\*

void Code\_Ht(Node Ht[],Code Hc[])

{

int i;

for(i=0;i<N\_LEAVE;i++)

{

Hc[i].Char = Ht[i].character;

Hc[i].Start = N\_LEAVE-1;

while(Ht[i].parent != -1 )

{

if(Ht[Ht[i].parent].lchild == i)

Hc[i].HufCode[Hc[i].Start] = 0;

if(Ht[Ht[i].parent].rchild == i)

Hc[i].HufCode[Hc[i].Start] = 1;

Hc[i].Start--;

i = Ht[i].parent;

}

}

}

\*/

void PrintCode(Code Hc[])

{

int i,j;

for(i=0;i<N\_LEAVE;i++)

{

for(j=Hc[i].Start;j<N\_LEAVE;j++)

{

printf("%d",Hc[i].HufCode[j]);

}

printf("%5c\n",Hc[i].Char);

}

}

void FindCode(Code Hc[],char s[])

{

int i,j,k=0;

printf("\n");

printf("\n");

while(s[k]!='\0'){

for(i=0;i<N\_LEAVE;i++)

{

if( s[k] == Hc[i].Char )

{

for(j=Hc[i].Start;j<N\_LEAVE;j++)

{

printf("%d",Hc[i].HufCode[j]);

}

}

}

k++;

}

}

void trans(char index[],Node hc[])

{

int i,root,p;

for(i=0;i<N\_NODE;i++)

{

if(hc[i].parent==-1)

{

root=i;

break;

}

}

p=root;

for(i=0;index[i]!='\0';i++)

{

if(index[i]=='1')

{

p=hc[p].rchild;

}

if(index[i]=='0')

{

p=hc[p].lchild;

}

if(hc[p].lchild==-1&&hc[p].rchild==-1)

{

printf("%c",hc[p].character);

p=root;

}

}

}

int main()

{

Node HufTree[N\_NODE];

Code HCode[N\_LEAVE];

int Wt[N\_LEAVE] = {64,13,22,32,103,21,15,47,57,1,5,32,20,57,63,15,1,48,51,80,23,8,18,1,16,1};

char s[]={"THISPROGRAMISMYFAVORITE"};

Huffman(HufTree,Wt);

Code\_Ht(HufTree,HCode);

PrintCode(HCode);

FindCode(HCode,s);

char string[20];

scanf("%s",string);

trans(string,HufTree);

}

四．小结

一、这次课程设计的心得体会通过实践我的收获如下

1、巩固和加深了对数据结构的理解，提高综合运用本课程所学知识的能力。

2、培养了我选用参考书，查阅手册及文献资料的能力。培养独立思考，深入研究，分析问题、解决问题的能力。

3、通过实际编译系统的分析设计、编程调试，掌握应用软件的分析方法和工程设计方法。

4、通过课程设计，培养了我严肃认真的工作作风，逐步建立正确的生产观念、经济观念和全局观念。

二、根据我在实习中遇到得问题，我将在以后的学习过程中注意以下几点：

1、认真上好专业实验课，多在实践中锻炼自己。

2、写程序的过程中要考虑周到，严密。

3、在做设计的时候要有信心，有耐心，切勿浮躁。

4、认真的学习课本知识，掌握课本中的知识点，并在此基础上学会灵活运用。

5、在课余时间里多写程序，熟练掌握在调试程序的过程中所遇到的常见错误，以便能节省调试程序的时间。