|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验名称 | 编译码器的设计与实现 | 实验地点 | 小营机房 | 实验时间 |  |
| 1. 课程设计目的：   学习完《数据结构》这门课后，要求学生综合运用本门课各章节知识解决实际问题。编译码器在生活中应用广泛，学生设计这样的软件既提高学习的趣味性又增强解决实际问题的能力。  该项目涉及《数据结构》多个章节的重要内容，如链表、顺序表、二叉树、最优二叉树、哈夫曼编码、文件处理等，对提高学生的编程能力有很大的帮助。 | | | | | |
| 1. 课程设计内容：   该项目需设计并实现一个对任意英文文章的编译码器。  编码器的功能包括：   * + 统计英文文章中所有字母出现的概率；   + 根据字母出现的概率用HUFFMAN算法构造最优二叉树；   + 记录每个字母的HUFFMAN编码于文件中；   + 根据HUFFMAN编码把英文文章译成01编码的报文。   译码器的功能包括：   * 根据HUFFMAN编码把01编码报文还原为原来的英文文章。 | | | | | |
| 1. 课程设计要求：   该项目1人一组独立完成，一人一份实验报告，在“实验方法和步骤”中需写以下内容：   * + 概述编译码器的功能；   + 画出编译码器的代码执行流程图；   + 自己在项目中完成的任务；   + 附上自己编写的代码，其中重要算法的算法思想，要求对关键语句加上注释。   具体功能包括：   1. 建立哈夫曼树：读入文件(\*.souce)，统计文件中字符出现的频度，并以这些字符的频度作为权值，建立哈夫曼树。 2. 编码：利用已建立好的哈夫曼树，获得各个字符的哈夫曼编码，并对正文进行编码，然后输出编码结果，并存入文件(\*.code)中。 3. 译码：利用已建立好的哈夫曼树将文件(\*.code)中的代码进行译码，并输出译码结果，并存入文件(\*.decode)中。   执行流程图 | | | | | |
| 1. 实验条件：   测试环境：Win11，Dev-C++ | | | | | |

#### 1、实验方法与步骤：

**各个模块详细的功能描述。**

2、选择颜色界面：选择背景色及字体颜色。

3、选择功能界面：用户选择实现哪一种功能。

4、写入文件：将用户输入的字符写入.souce文件中。

5、统计字符出现的频度：统计文件中不同字符出现的频度并建立哈夫曼树。

6、哈夫曼编码：将.souce中的文件进行编码，并把编码内容写入.code文件中。

7、打印哈夫曼树：根据数组中存储的信息将哈夫曼树打印出来。

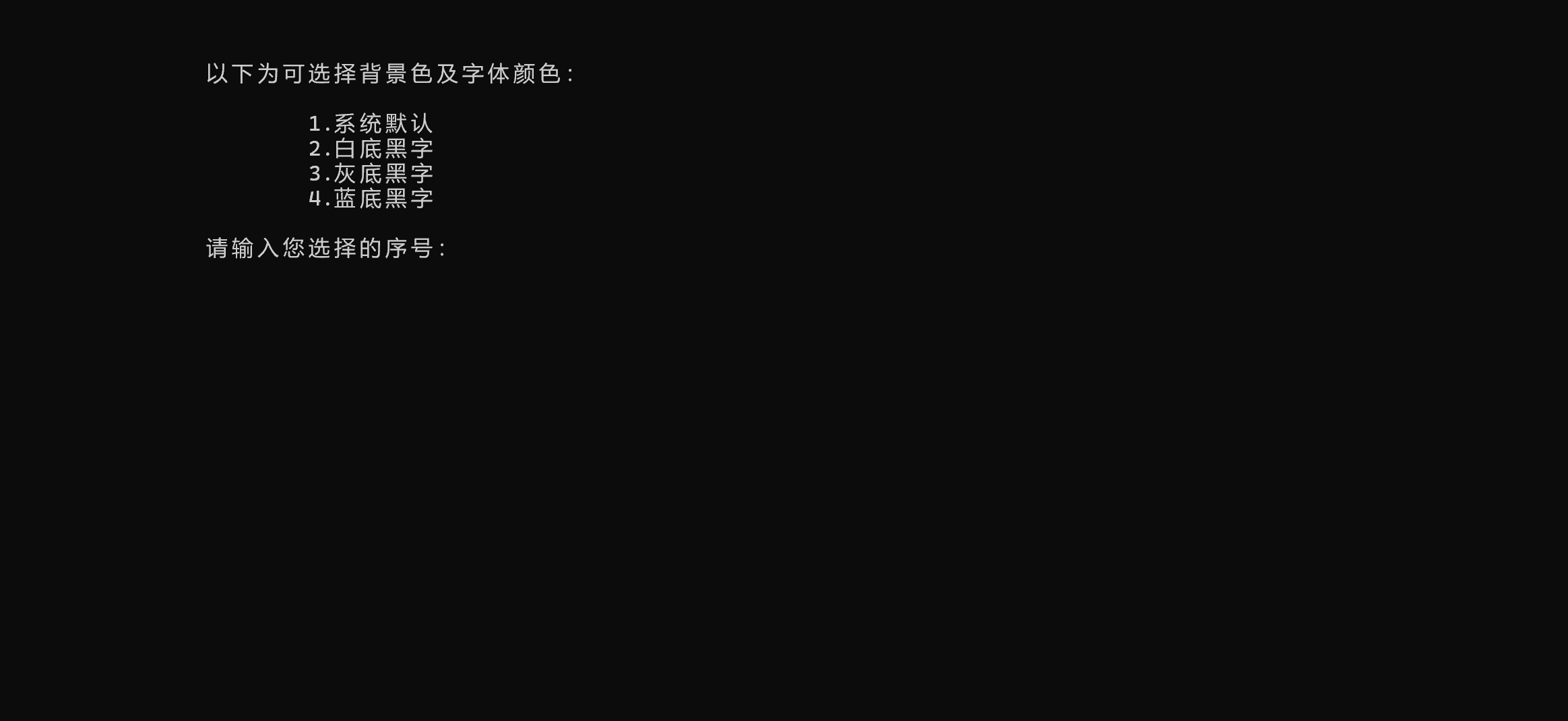
8、哈夫曼译码：对.code文件的内容进行译码，并将译码内容写入.decode文件中。

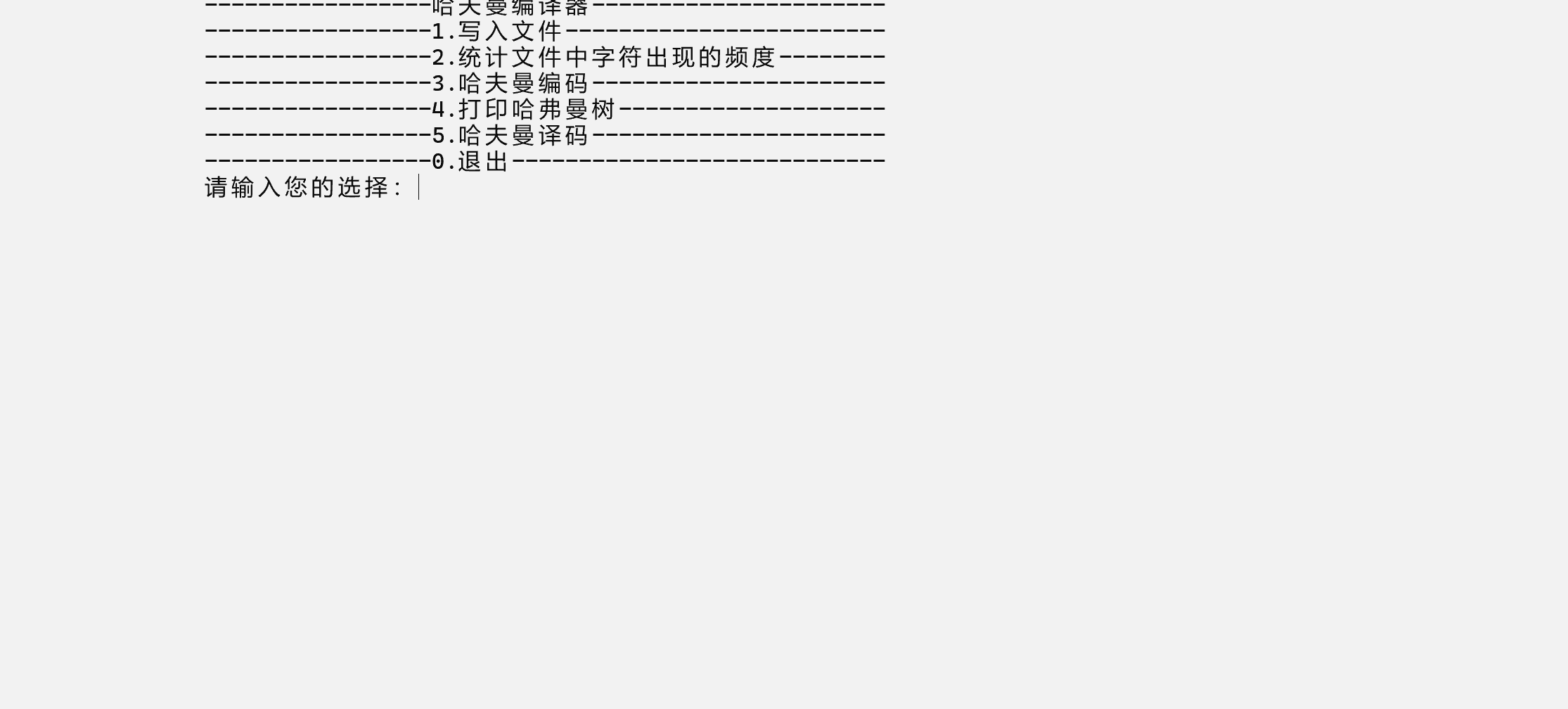
9、退出：程序结束运行，用户结束操作。

各功能函数的数据流程

#### 1、setColor()函数，使操作能美观个性一点：

void setColor()   
{   
 int c;   
 printf("\t\t以下为可选择背景色及字体颜色：\n\n");   
 printf("\t\t\t1.系统默认\n");   
 printf("\t\t\t2.白底黑字\n");   
 printf("\t\t\t3.灰底黑字\n");   
 printf("\t\t\t4.蓝底黑字\n\n");   
 printf("\t\t请输入您选择的序号：");   
 scanf("%d",&c);   
 switch(c)   
 {   
 case 1:break;   
 case 2:system("color F0"); break;   
 case 3:system("color 70"); break;   
 case 4:system("color B0"); break;   
 default:printf("\n\t\t输入错误，请您重新输入!\n\n");   
 setColor();   
 }   
 system("cls");//清空控制台屏幕   
 menu();   
}





其他效果图省略

#### 2、menu(void)函数，实现 写入文件、统计频率、编码、打印、译码等功能的选择

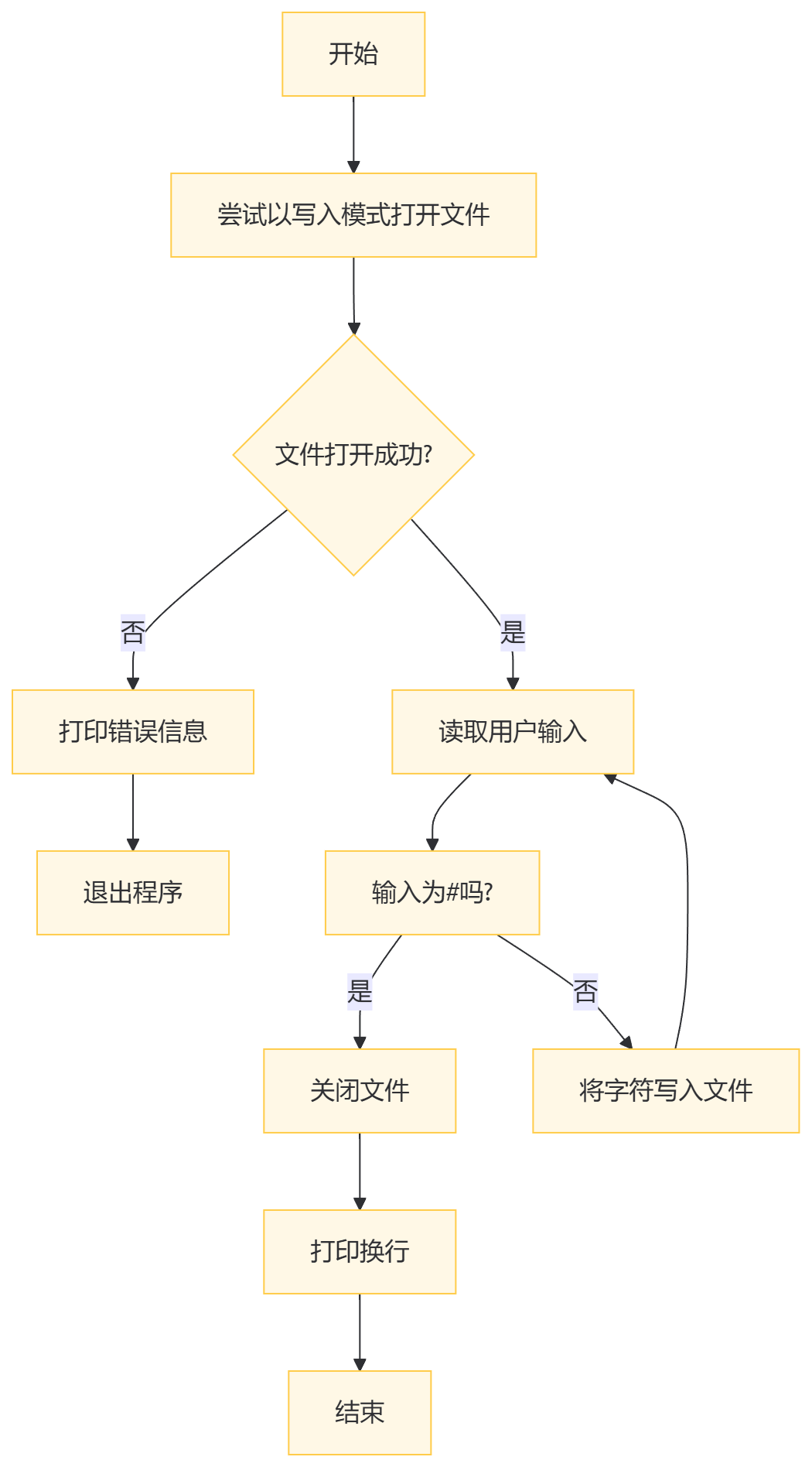
void menu(void)   
{   
 printf("\t\t-----------------哈夫曼编译器----------------------\n");   
 printf("\t\t-----------------1.写入文件------------------------\n");   
 printf("\t\t-----------------2.统计文件中字符出现的频度--------\n");   
 printf("\t\t-----------------3.哈夫曼编码----------------------\n");   
 printf("\t\t-----------------4.打印哈弗曼树--------------------\n");   
 printf("\t\t-----------------5.哈夫曼译码----------------------\n");   
 printf("\t\t-----------------0.退出----------------------------\n");   
 printf("\t\t请输入您的选择：");   
 int choose;   
 scanf("%d",&choose);   
 switch(choose)   
 {   
 case 0:exit(0);   
 case 1:write();break;   
 case 2:statistics();break;   
 case 3:Code();break;   
 case 4:Print();break;   
 case 5:Decode();break;   
 default:printf("\n\t\t输入错误，请您重新输入！\n");   
 printf("\t\t请按任意键继续！\n\n");getchar();getchar();   
 system("cls");   
 menu();   
 }   
 printf("\t\t请按任意键继续。") ;   
 getchar();getchar();   
 system("cls");   
 menu();   
}

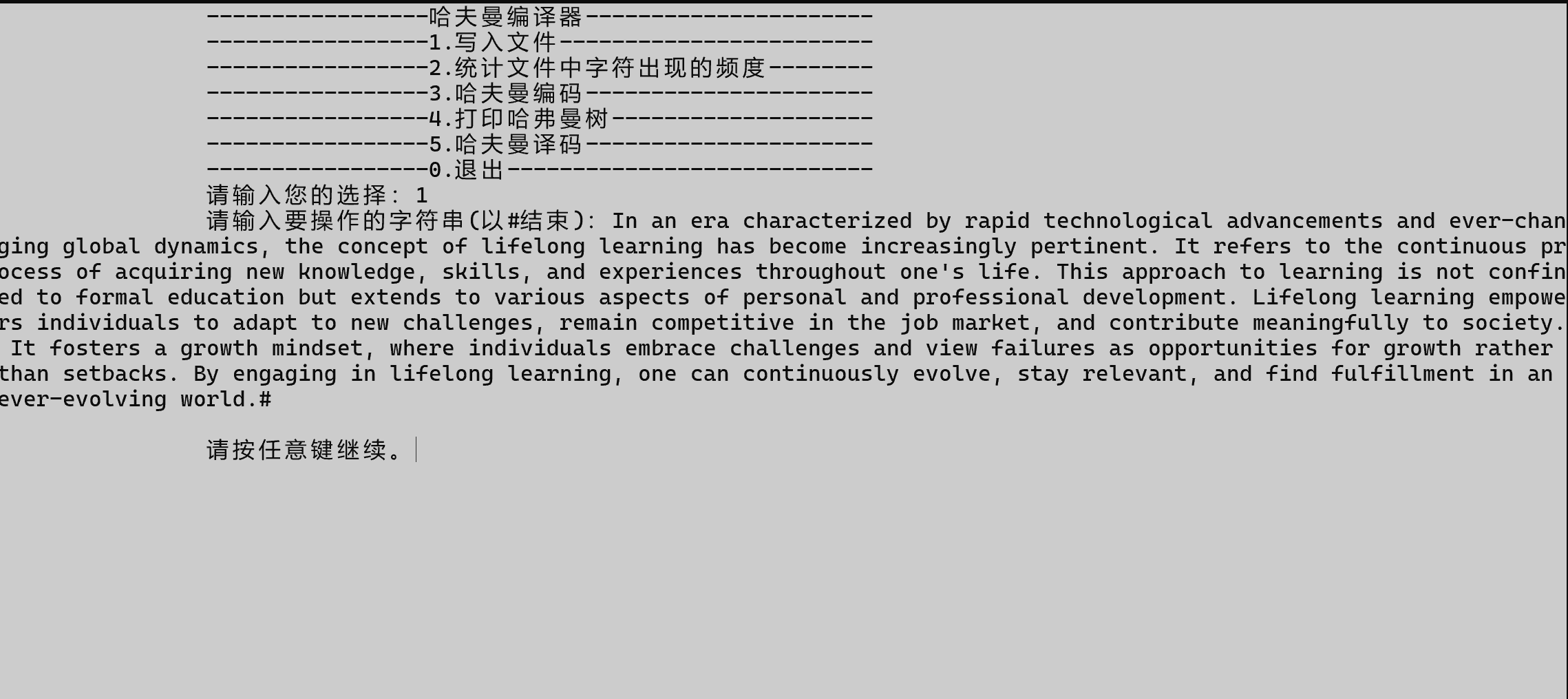


各个具体功能将在下文介绍展示

#### 3、write()函数，使写入文本内容：

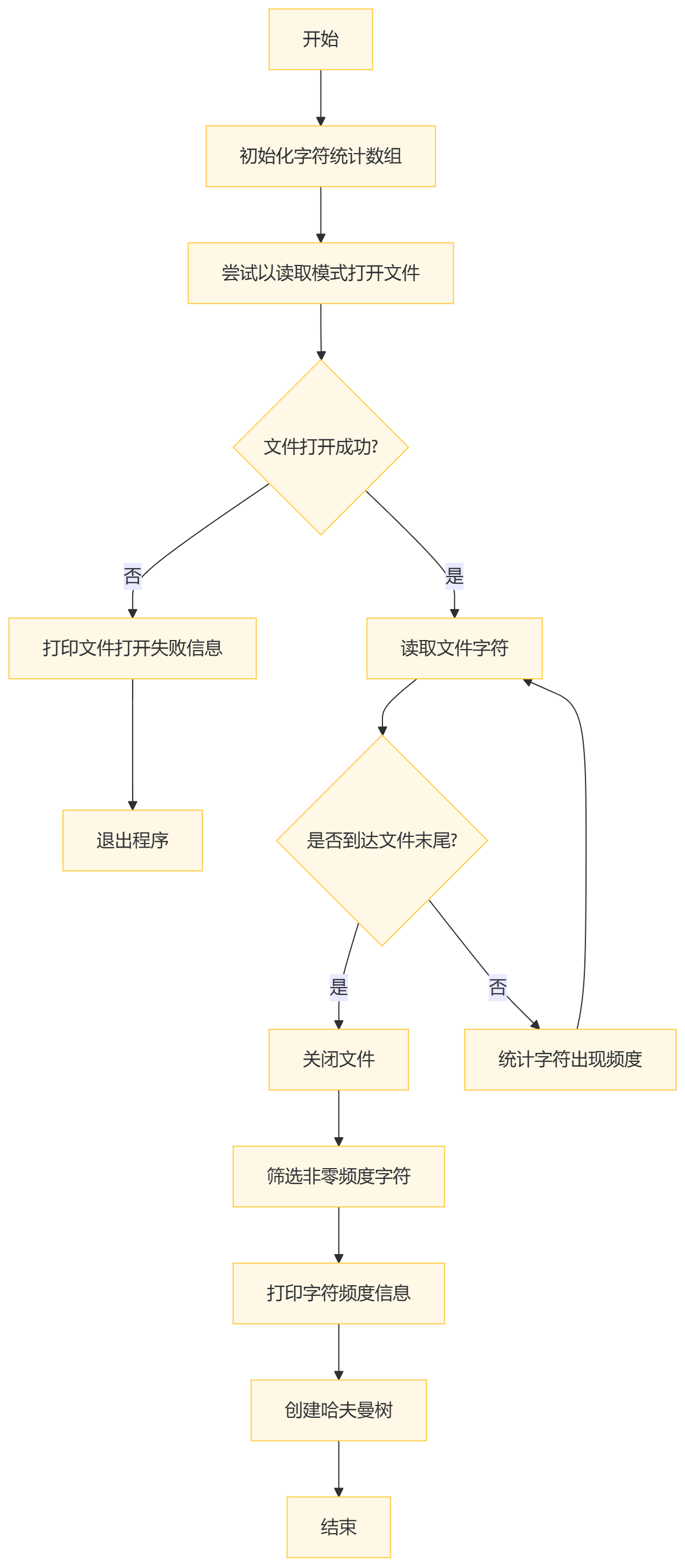
void write()   
{   
 FILE \*fp;   
 char ch;   
 //尝试以写入模式('w')打开文件   
 if((fp = fopen("Haffman.souce","w")) == NULL){   
 printf("发生错误！\n");   
 exit(1);   
 }   
 ch = getchar();   
 printf("\t\t请输入要操作的字符串(以#结束)：");   
 ch = getchar();   
 while(ch != '#'){   
 //将字符写入文件中   
 fputc(ch,fp);   
 ch = getchar();   
 }   
 fclose(fp);   
 printf("\n");   
}



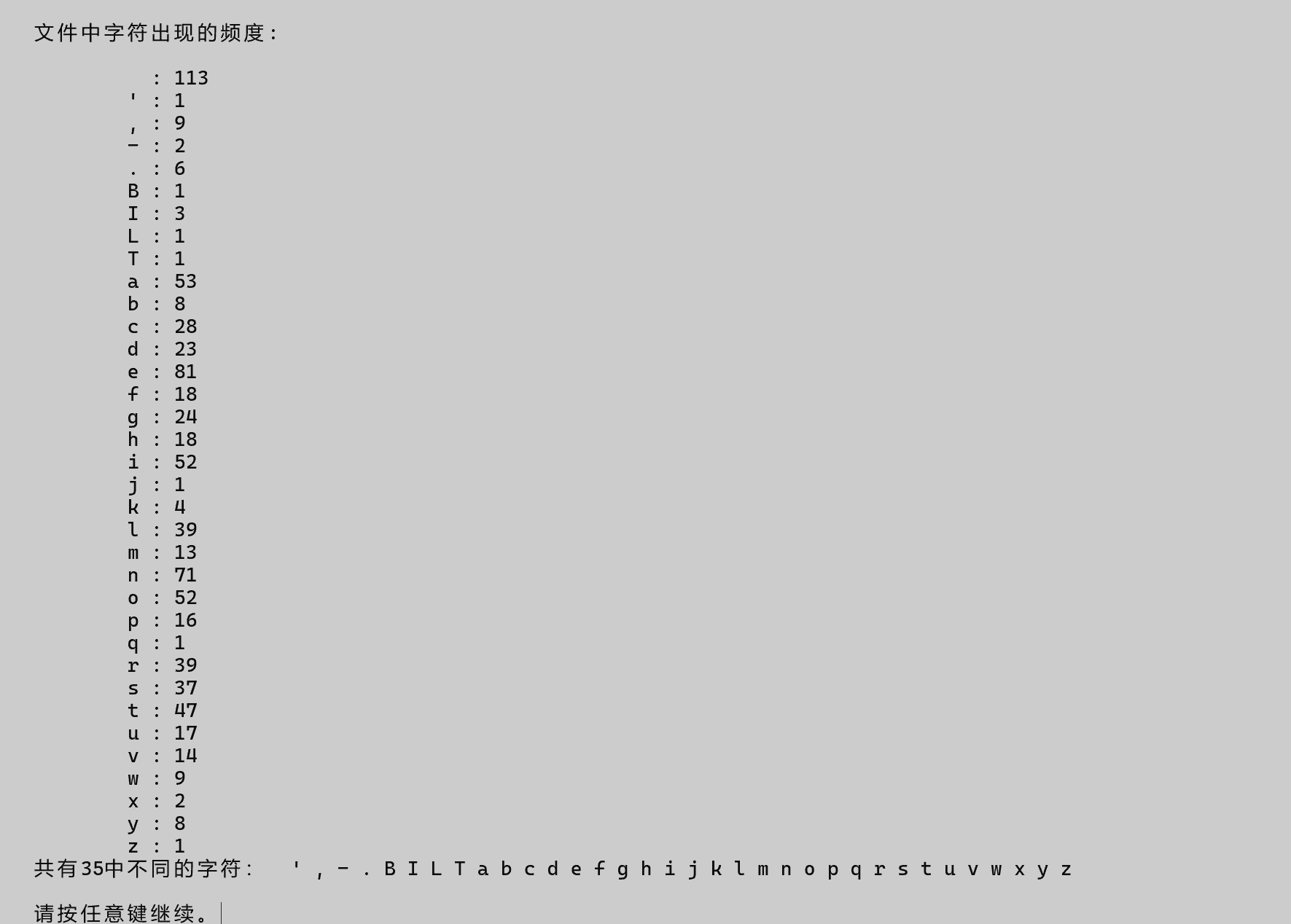


#### 4、statistics()函数，统计各个字符的频率

void statistics()   
{   
 FILE \*fp;   
 char ch;   
 str s1[128],s2[128]; //ASCII字符集有128个可能的字符   
   
 for(int i=0;i<128;i++){   
 s1[i].data = 0;   
 s1[i].num = 0;   
 s2[i].data = 0;   
 s2[i].num = 0;   
 }   
 //以读取模式('r')打开文件   
 if((fp = fopen("Haffman.souce","r")) == NULL){   
 printf("\n文件打开失败！\n");   
 exit(1);   
 }   
   
 printf("\n\t\t读取到的字符串为：");   
 ch = fgetc(fp);   
 while(!feof(fp)){ //直到文件末尾   
 printf("%c",ch);   
 s1[ch].data = ch;   
 s1[ch].num ++;   
 ch = fgetc(fp);   
 }   
 fclose(fp);   
 count=1;   
 for (int i=1;i<128;i++){   
 if(s1[i].num!=0){   
 s2[count].data = s1[i].data;   
 s2[count].num = s1[i].num;   
 count++;   
 }   
 }   
 count-=1;   
 printf("\n\n\t\t文件中字符出现的频度:\n\n");   
 for(int i=1;i<=count;i++)   
 {   
 printf("\t\t\t%c : %d\n",s2[i].data,s2[i].num);   
 }   
 printf("\t\t共有%d中不同的字符：",count);   
   
 CrtHuffmanTree(ht,s2);   
}

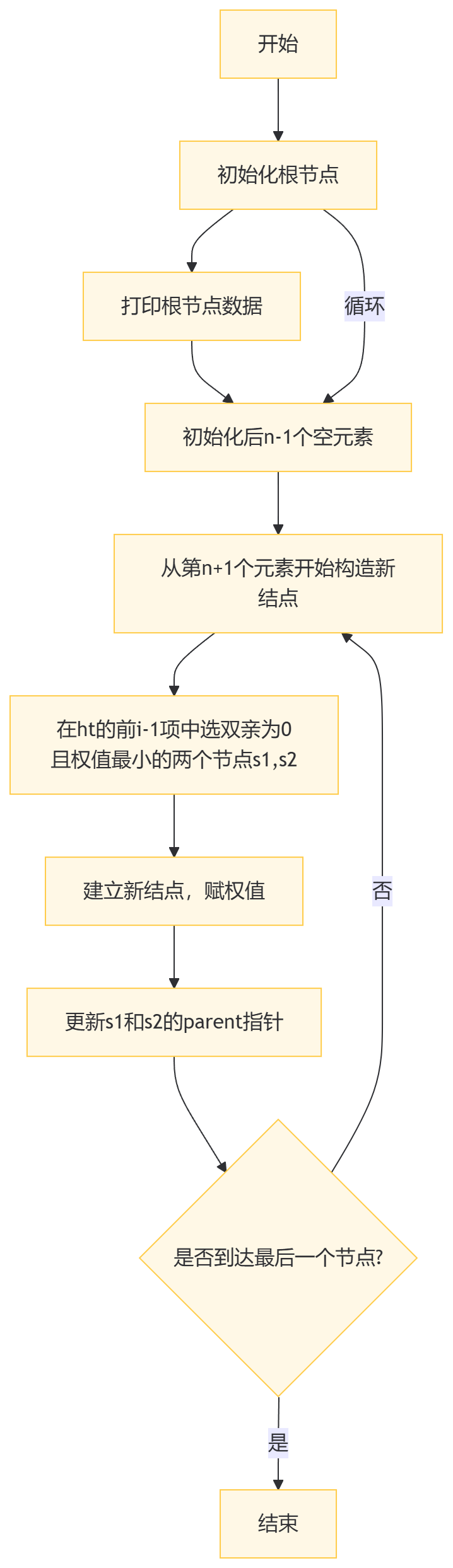


效果图：



#### 5、CrtHuffmanTree(HuffmanTree ht,str \*s)函数，建立哈夫曼树：

//建立哈夫曼树   
void CrtHuffmanTree(HuffmanTree ht,str \*s)   
{   
 //初始化前n个元素成为根节点   
 for(int i=1;i<=count;i++){   
 ht[i].data = s[i].data;   
 printf("%c ",ht[i].data);   
 ht[i].weight = s[i].num;   
 ht[i].parent = 0;   
 ht[i].lchild = 0;   
 ht[i].rchild = 0;   
 }   
 printf("\n\n");   
 //初始化后n-1个空元素   
 int m = 2\*count-1;   
 for(int i=count+1;i<=m;i++){   
 ht[i].data = 0;   
 ht[i].weight = 0;   
 ht[i].parent = 0;   
 ht[i].lchild = 0;   
 ht[i].rchild = 0;   
 }   
 //从第n+1个元素开始构造新结点   
 int s1,s2;   
 for(int i=count+1;i<=m;i++){   
 //在ht的前i-1项中选双亲为0且权值最小的两个节点s1,s2   
 select(ht,i-1,&s1,&s2);   
 //建立新结点，赋权值   
 ht[i].weight = ht[s1].weight + ht[s2].weight;   
 ht[i].lchild = s1;   
 ht[i].rchild = s2;   
 ht[s1].parent = i;   
 ht[s2].parent = i;   
 }   
}

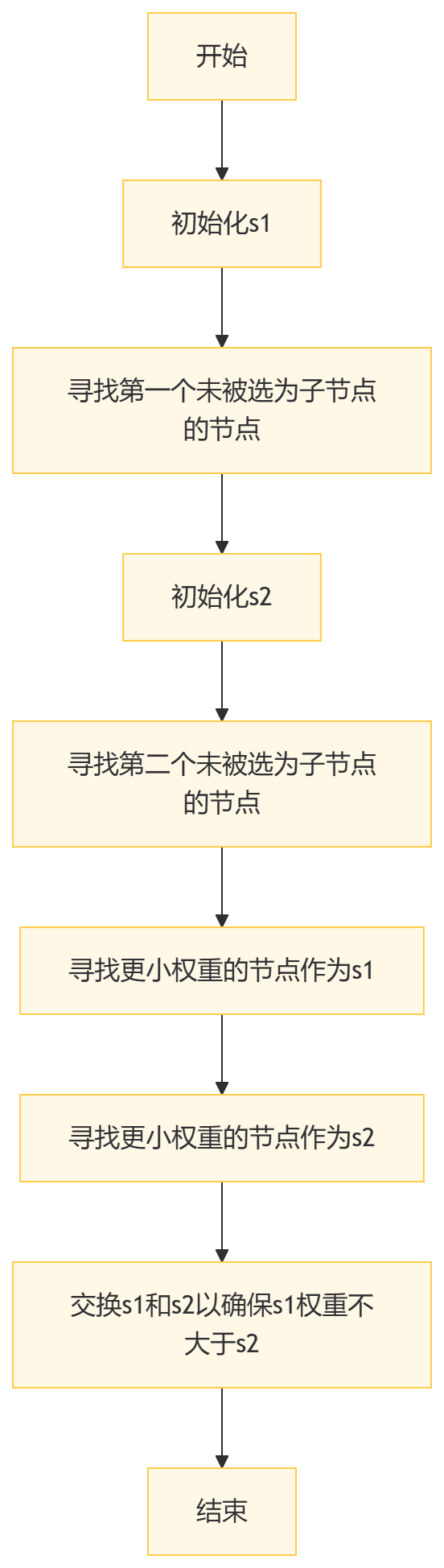


效果图：



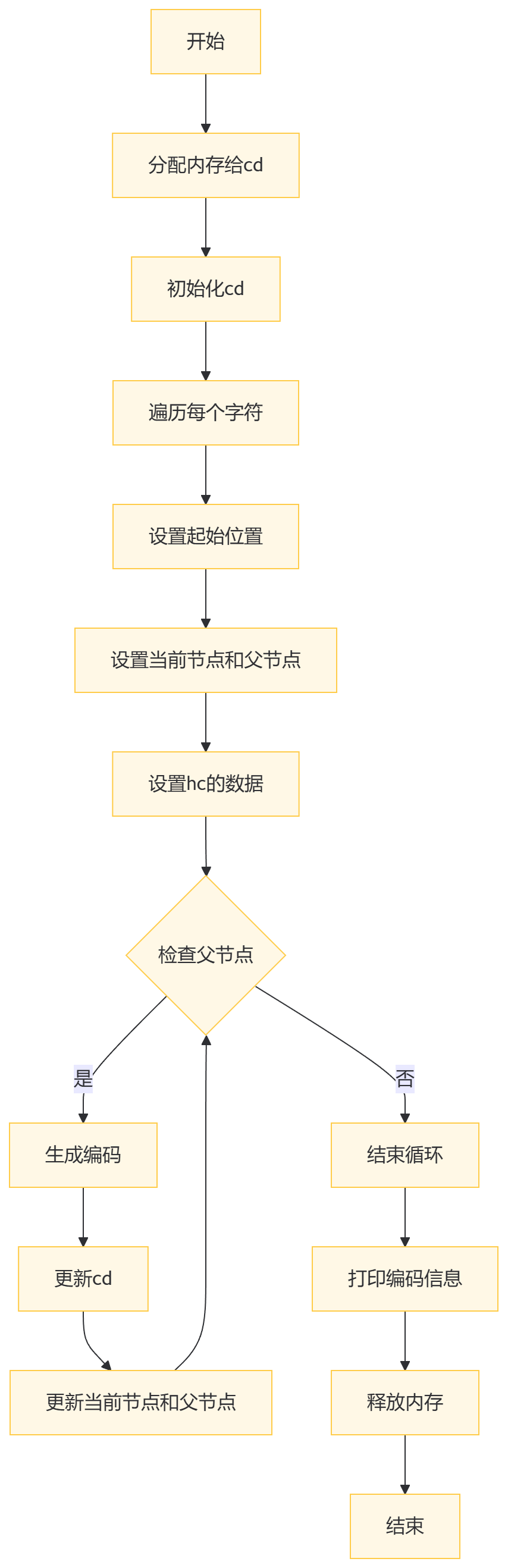
#### 6、select(HuffmanTree ht,int n,int \*s1,int \*s2)函数，挑选出权值最小的两个结点

void select(HuffmanTree ht,int n,int \*s1,int \*s2)   
{   
 int i,j;   
 for(i=1;i<=n;i++){   
 if(ht[i].parent == 0){   
 \*s1 = i;   
 break;   
 }   
 }   
 for(j=i+1;j<=n;j++){   
 if(ht[j].parent == 0){   
 \*s2 = j;   
 break;   
 }   
 }   
 for(i=1;i<=n;i++){   
 if((ht[i].weight<ht[\*s1].weight)&&(ht[i].parent==0)&&(\*s2!=i))   
 \*s1 = i;   
 }   
 for(j=1;j<=n;j++){   
 if((ht[j].weight<ht[\*s2].weight)&&(ht[j].parent==0)&&(\*s1!=j))   
 \*s2 = j;   
 }   
 if(ht[\*s1].weight>ht[\*s2].weight){   
 int temp = \*s1;   
 \*s1 = \*s2;   
 \*s2 = temp;   
 }   
}



#### 7、CrtHuffmanCode(HuffmanTree ht,HuffmanCode hc)函数，实现编码

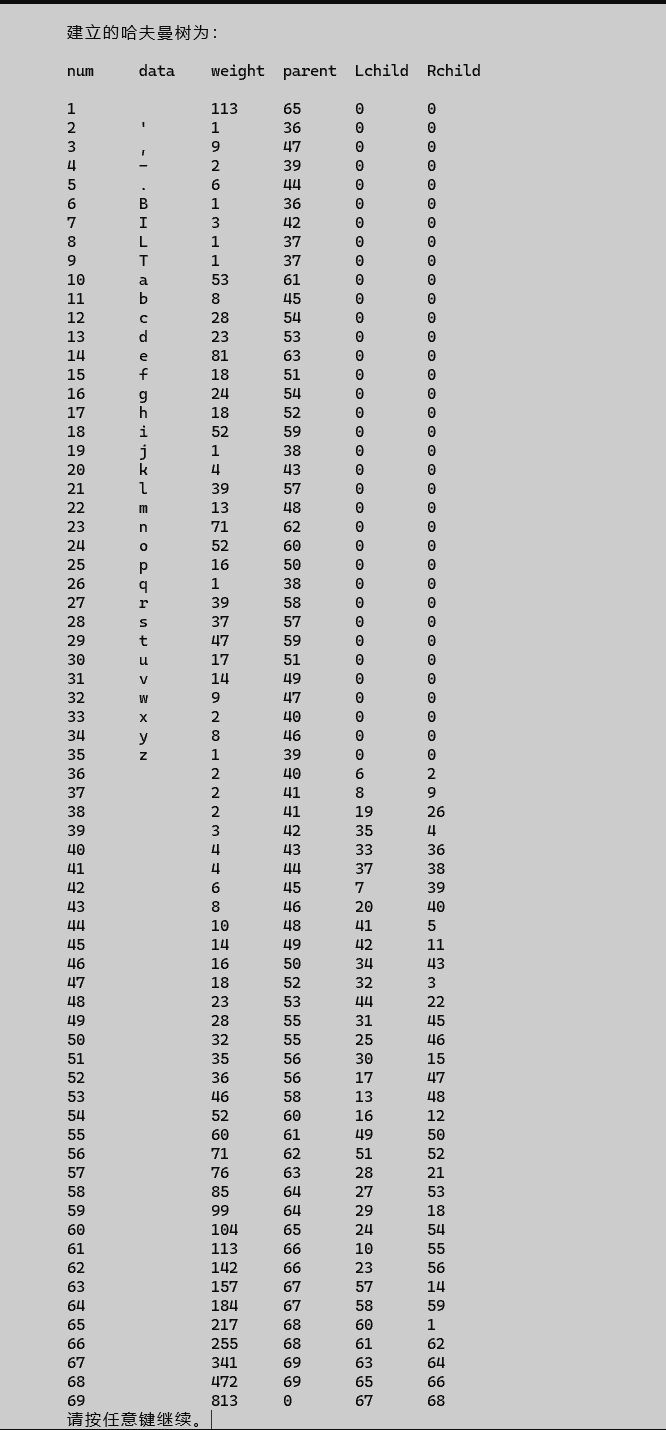
//哈夫曼编码   
void CrtHuffmanCode(HuffmanTree ht,HuffmanCode hc)   
{   
 int c,p;   
 char \*cd;   
 int start;   
   
 cd = (char \*)malloc(count\*sizeof(char));   
 cd[count-1] = '\0';   
   
 for(int i=1;i<=count;i++){   
 start = count-1;   
 c=i;   
 p=ht[i].parent;   
   
 hc[i].data = ht[i].data;   
   
 while(p!=0){   
 --start;   
 if(ht[p].lchild == c) cd[start]='0';   
 else cd[start] = '1';   
 c = p;   
 p = ht[p].parent;   
 }   
 strcpy(hc[i].bits,&cd[start]);   
 }   
 printf("\n\t\t每个字符对应的编码如下:\n");   
 printf("\t\t序号 符号 编码\n");   
 for(int i=1;i<=count;i++)   
 printf("\t\t%-3d %c %s \n",i,hc[i].data,hc[i].bits);   
}





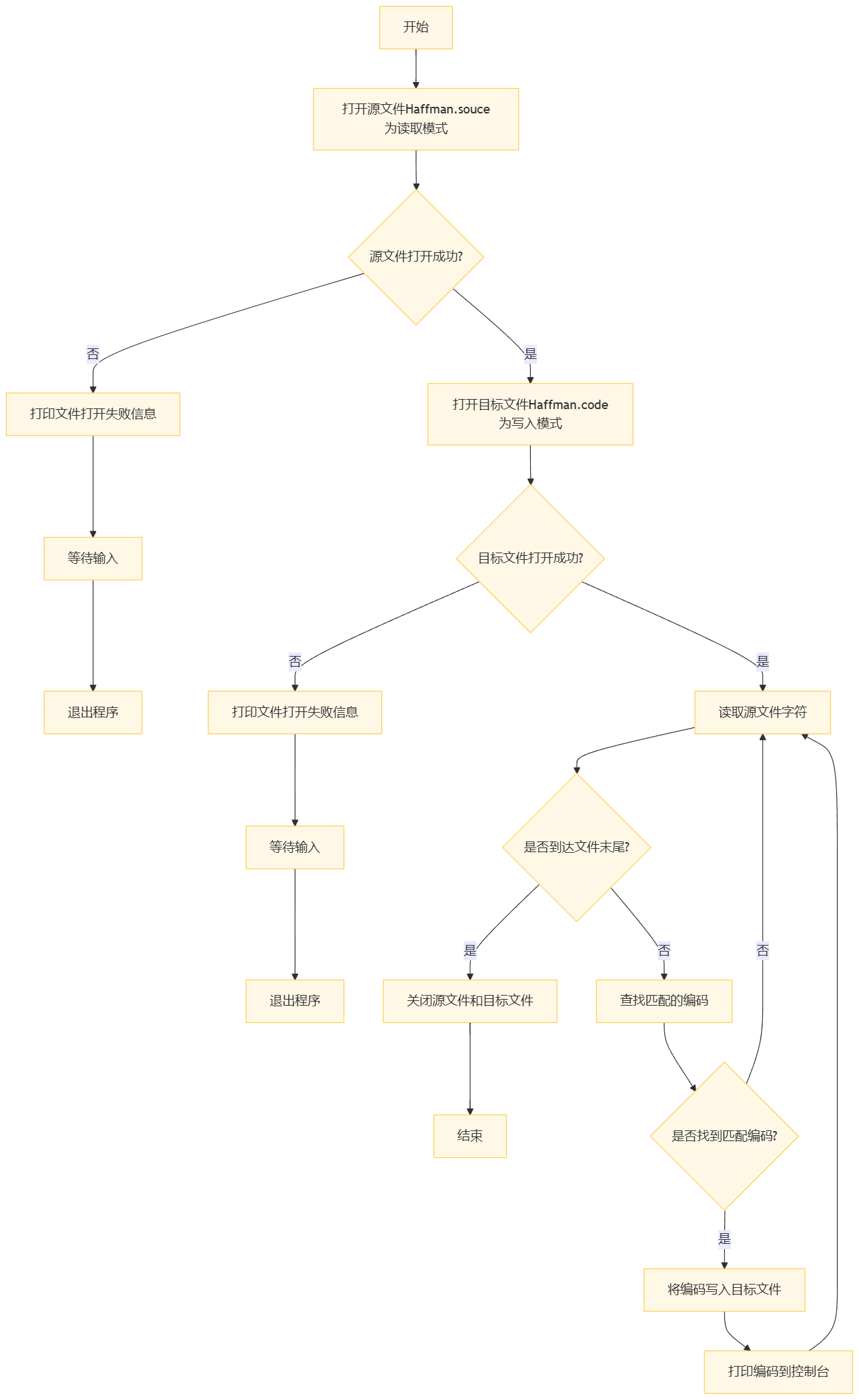
#### 8、打印树

void Print()   
{   
 system("CLS");   
 printf("\n\t\t建立的哈夫曼树为：\n");   
 printf("\n\t\tnum\tdata\tweight\tparent\tLchild\tRchild\n");   
 for(int i=1;i<=2\*count-1;i++)   
 {   
 printf("\n\t\t%d\t%c\t%d\t%d\t%d\t%d",i,ht[i].data,ht[i].weight,ht[i].parent,ht[i].lchild,ht[i].rchild);   
 }   
 printf("\n");   
}



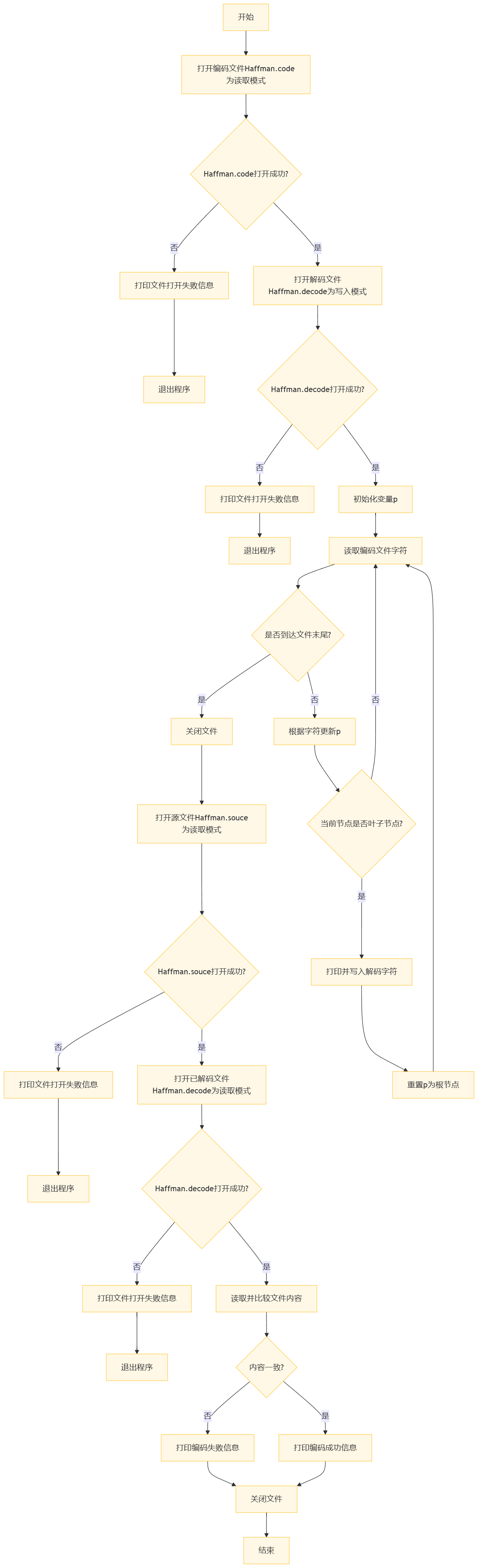
#### 9、编码写入文件

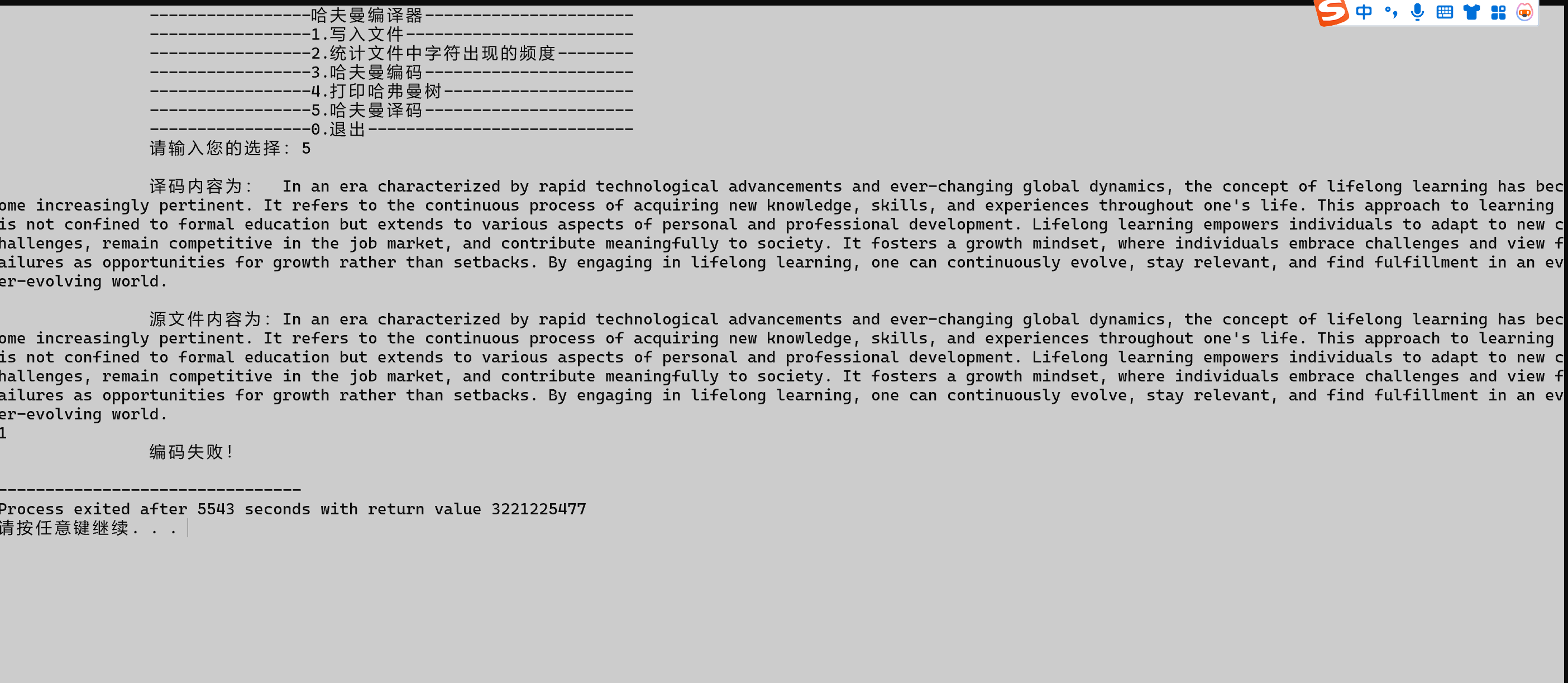
//把编码结果写入文件   
void CodeToFile(HuffmanCode hc,int n)   
{   
 FILE \*fp1,\*fp2;   
 char ch;   
   
 if((fp1 = fopen("Haffman.souce","r")) == NULL){   
 printf("文件打开失败");   
 getchar();   
 exit(1);   
 }   
 if((fp2 = fopen("Haffman.code","w")) == NULL){   
 printf("文件打开失败");   
 getchar();   
 exit(1);   
 }   
   
 ch = fgetc(fp1);   
 printf("\t\t编码内容为：");   
 while(ch!=EOF){   
 for(int i=1;i<=count;i++){   
 if(ch == hc[i].data){   
 fputs(hc[i].bits,fp2);   
 printf("%s",hc[i].bits);   
 }   
 }   
 ch = fgetc(fp1);   
 }   
 printf("\n");   
 fclose(fp1);   
 fclose(fp2);   
 }



#### 10、译码

void Decode()   
{   
   
 FILE \*fp1,\*fp2;   
 int p;   
 char ch;   
   
 if((fp1 = fopen("Haffman.code","r")) == NULL){   
 printf("文件打开失败");   
 exit(1);   
 }   
 if((fp2 = fopen("Haffman.decode","w")) == NULL){   
 printf("文件打开失败");   
 exit(1);   
 }   
   
 p = 2\*count-1;   
 printf("\n\t\t译码内容为： ");   
 ch = fgetc(fp1);   
 while(!feof(fp1)){   
 if(ch == '0')   
 p = ht[p].lchild;   
 else if(ch == '1')   
 p = ht[p].rchild;   
 if(ht[p].data != 0){   
 printf("%c",ht[p].data);   
 fputc(ht[p].data,fp2);   
 p = 2\*count-1;   
 }   
 ch = fgetc(fp1);   
 }   
 printf("\n");   
 fclose(fp1);   
 fclose(fp2);   
 //验证解码正确性   
 int i,j=1;   
 char s1[N],s2[N];   
   
 if((fp1 = fopen("Haffman.souce","r")) == NULL){   
 printf("\n打开文件失败！\n");   
 exit(1);   
 }   
 if((fp2 = fopen("Haffman.decode","r")) == NULL){   
 printf("\n打开文件失败！\n");   
 exit(1);   
 }   
 for(i = 1; (s2[i]=fgetc(fp2)) != EOF;i++){   
 }   
 printf("\n\t\t源文件内容为：");   
 for(i = 1; (s1[i]=fgetc(fp1)) != EOF;i++){   
 printf("%c",s1[i]);   
 }   
 printf("\n");   
 while( j < count){   
 if(s1[j] == s2[j]){   
 j++;   
 }   
 else{   
 printf("%d",j);   
 printf("\n\t\t编码失败！\n");   
 break;   
 }   
 }   
 if(j == count){   
 printf("\n\t\t编码内容一致，编码成功。\n");   
 }   
 fclose(fp1);   
 fclose(fp2);   
}





#### 11、测试哈夫曼编码算法的几个关键步骤，test()函数：

void test() {   
 FILE \*fp;   
 char ch;   
 str s1[128],s2[128];   
   
 for(int i=0;i<128;i++){   
 s1[i].data = 0;   
 s1[i].num = 0;   
 s2[i].data = 0;   
 s2[i].num = 0;   
 }   
 if((fp = fopen("Haffman.souce","r")) == NULL){   
 exit(1);   
 }   
   
 ch = fgetc(fp);   
 while(!feof(fp)){   
 s1[ch].data = ch;   
 s1[ch].num ++;   
 ch = fgetc(fp);   
 }   
 fclose(fp);   
 count=1;   
 for (int i=1;i<128;i++){   
 if(s1[i].num!=0){   
 s2[count].data = s1[i].data;   
 s2[count].num = s1[i].num;   
 count++;   
 }   
 }   
 count-=1;   
 for(int i=1;i<=count;i++){   
 ht[i].data = s2[i].data;   
 ht[i].weight = s2[i].num;   
 ht[i].parent = 0;   
 ht[i].lchild = 0;   
 ht[i].rchild = 0;   
 }   
 printf("\n\n");   
 //初始化后n-1个空元素   
 int m = 2\*count-1;   
 for(int i=count+1;i<=m;i++){   
 ht[i].data = 0;   
 ht[i].weight = 0;   
 ht[i].parent = 0;   
 ht[i].lchild = 0;   
 ht[i].rchild = 0;   
 }   
 //从第n+1个元素开始构造新结点   
 int s3,s4;   
 for(int i=count+1;i<=m;i++){   
 //在ht的前i-1项中选双亲为0且权值最小的两个节点s1,s2   
 select(ht,i-1,&s3,&s4);   
 //建立新结点，赋权值   
 ht[i].weight = ht[s3].weight + ht[s4].weight;   
 ht[i].lchild = s3;   
 ht[i].rchild = s4;   
 ht[s3].parent = i;   
 ht[s4].parent = i;   
 }   
}

完整代码：

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<windows.h>

#include<string.h>

#define N 600 //最大字符

#define M 2\*N-1 //最大数量的节点

int count = 0; //记录不同字符的数量

typedef struct Huffmantree//哈夫曼树

{

char data;

int weight;

int parent;

int lchild;

int rchild;

}HuffmanTree[M];

typedef struct Huffmancode //哈夫曼编码

{

char data;

char bits[N];

}HuffmanCode[N];

typedef struct str//字符数组

{

char data;

int num;

}str;

HuffmanTree ht;

HuffmanCode hc;

void setColor(); //设置颜色

void menu(); //选项

void write(); //写入文件

void statistics(); //写出文件

void Code(); //哈夫曼编码

void Print(); //打印哈弗曼树

void Decode(); //哈夫曼译码

void CrtHuffmanTree(HuffmanTree ht,str \*s);//创建哈夫曼树

void CrtHuffmanCode(HuffmanTree ht,HuffmanCode hc);//创建哈夫曼编码

void select(HuffmanTree ht,int i,int \*s1,int \*s2);//选择两个权重最小的节点

void CodeToFile(HuffmanCode hc,int n);//将编码写入文件

void setColor()

{

int c;

printf("\t\t以下为可选择背景色及字体颜色：\n\n");

printf("\t\t\t1.系统默认\n");

printf("\t\t\t2.白底黑字\n");

printf("\t\t\t3.灰底黑字\n");

printf("\t\t\t4.蓝底黑字\n\n");

printf("\t\t请输入您选择的序号：");

scanf("%d",&c);

switch(c)

{

case 1:break;

case 2:system("color F0"); break;

case 3:system("color 70"); break;

case 4:system("color B0"); break;

default:printf("\n\t\t输入错误，请您重新输入!\n\n");

setColor();

}

system("cls");//清空控制台屏幕

menu();

}

void menu(void)

{

printf("\t\t-----------------哈夫曼编译器----------------------\n");

printf("\t\t-----------------1.写入文件------------------------\n");

printf("\t\t-----------------2.统计文件中字符出现的频度--------\n");

printf("\t\t-----------------3.哈夫曼编码----------------------\n");

printf("\t\t-----------------4.打印哈弗曼树--------------------\n");

printf("\t\t-----------------5.哈夫曼译码----------------------\n");

printf("\t\t-----------------0.退出----------------------------\n");

printf("\t\t请输入您的选择：");

int choose;

scanf("%d",&choose);

switch(choose)

{

case 0:exit(0);

case 1:write();break;

case 2:statistics();break;

case 3:Code();break;

case 4:Print();break;

case 5:Decode();break;

default:printf("\n\t\t输入错误，请您重新输入！\n");

printf("\t\t请按任意键继续！\n\n");getchar();getchar();

system("cls");

menu();

}

printf("\t\t请按任意键继续。") ;

getchar();getchar();

system("cls");

menu();

}

void write()

{

FILE \*fp;

char ch;

//尝试以写入模式('w')打开文件

if((fp = fopen("Haffman.souce","w")) == NULL){

printf("发生错误！\n");

exit(1);

}

ch = getchar();

printf("\t\t请输入要操作的字符串(以#结束)：");

ch = getchar();

while(ch != '#'){

//将字符写入文件中

fputc(ch,fp);

ch = getchar();

}

fclose(fp);

printf("\n");

}

void statistics()

{

FILE \*fp;

char ch;

str s1[128],s2[128]; //ASCII字符集有128个可能的字符

for(int i=0;i<128;i++){

s1[i].data = 0;

s1[i].num = 0;

s2[i].data = 0;

s2[i].num = 0;

}

//以读取模式('r')打开文件

if((fp = fopen("Haffman.souce","r")) == NULL){

printf("\n文件打开失败！\n");

exit(1);

}

printf("\n\t\t读取到的字符串为：");

ch = fgetc(fp);

while(!feof(fp)){ //直到文件末尾

printf("%c",ch);

s1[ch].data = ch;

s1[ch].num ++;

ch = fgetc(fp);

}

fclose(fp);

count=1;

for (int i=1;i<128;i++){

if(s1[i].num!=0){

s2[count].data = s1[i].data;

s2[count].num = s1[i].num;

count++;

}

}

count-=1;

printf("\n\n\t\t文件中字符出现的频度:\n\n");

for(int i=1;i<=count;i++)

{

printf("\t\t\t%c : %d\n",s2[i].data,s2[i].num);

}

printf("\t\t共有%d中不同的字符：",count);

CrtHuffmanTree(ht,s2);

}

void Code()

{

CrtHuffmanCode(ht,hc);

CodeToFile(hc,count);

}

//建立哈夫曼树

void CrtHuffmanTree(HuffmanTree ht,str \*s)

{

//初始化前n个元素成为根节点

for(int i=1;i<=count;i++){

ht[i].data = s[i].data;

printf("%c ",ht[i].data);

ht[i].weight = s[i].num;

ht[i].parent = 0;

ht[i].lchild = 0;

ht[i].rchild = 0;

}

printf("\n\n");

//初始化后n-1个空元素

int m = 2\*count-1;

for(int i=count+1;i<=m;i++){

ht[i].data = 0;

ht[i].weight = 0;

ht[i].parent = 0;

ht[i].lchild = 0;

ht[i].rchild = 0;

}

//从第n+1个元素开始构造新结点

int s1,s2;

for(int i=count+1;i<=m;i++){

//在ht的前i-1项中选双亲为0且权值最小的两个节点s1,s2

select(ht,i-1,&s1,&s2);

//建立新结点，赋权值

ht[i].weight = ht[s1].weight + ht[s2].weight;

ht[i].lchild = s1;

ht[i].rchild = s2;

ht[s1].parent = i;

ht[s2].parent = i;

}

}

void select(HuffmanTree ht,int n,int \*s1,int \*s2)

{

int i,j;

for(i=1;i<=n;i++){

if(ht[i].parent == 0){

\*s1 = i;

break;

}

}

for(j=i+1;j<=n;j++){

if(ht[j].parent == 0){

\*s2 = j;

break;

}

}

for(i=1;i<=n;i++){

if((ht[i].weight<ht[\*s1].weight)&&(ht[i].parent==0)&&(\*s2!=i))

\*s1 = i;

}

for(j=1;j<=n;j++){

if((ht[j].weight<ht[\*s2].weight)&&(ht[j].parent==0)&&(\*s1!=j))

\*s2 = j;

}

if(ht[\*s1].weight>ht[\*s2].weight){

int temp = \*s1;

\*s1 = \*s2;

\*s2 = temp;

}

}

//哈夫曼编码

void CrtHuffmanCode(HuffmanTree ht,HuffmanCode hc)

{

int c,p;

char \*cd;

int start;

cd = (char \*)malloc(count\*sizeof(char));

cd[count-1] = '\0';

for(int i=1;i<=count;i++){

start = count-1;

c=i;

p=ht[i].parent;

hc[i].data = ht[i].data;

while(p!=0){

--start;

if(ht[p].lchild == c) cd[start]='0';

else cd[start] = '1';

c = p;

p = ht[p].parent;

}

strcpy(hc[i].bits,&cd[start]);

}

printf("\n\t\t每个字符对应的编码如下:\n");

printf("\t\t序号 符号 编码\n");

for(int i=1;i<=count;i++)

printf("\t\t%-3d %c %s \n",i,hc[i].data,hc[i].bits);

}

//把编码结果写入文件

void CodeToFile(HuffmanCode hc,int n)

{

FILE \*fp1,\*fp2;

char ch;

if((fp1 = fopen("Haffman.souce","r")) == NULL){

printf("文件打开失败");

getchar();

exit(1);

}

if((fp2 = fopen("Haffman.code","w")) == NULL){

printf("文件打开失败");

getchar();

exit(1);

}

ch = fgetc(fp1);

printf("\t\t编码内容为：");

while(ch!=EOF){

for(int i=1;i<=count;i++){

if(ch == hc[i].data){

fputs(hc[i].bits,fp2);

printf("%s",hc[i].bits);

}

}

ch = fgetc(fp1);

}

printf("\n");

fclose(fp1);

fclose(fp2);

}

void Print()

{

system("CLS");

printf("\n\t\t建立的哈夫曼树为：\n");

printf("\n\t\tnum\tdata\tweight\tparent\tLchild\tRchild\n");

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

{

printf("\n\t\t%d\t%c\t%d\t%d\t%d\t%d",i,ht[i].data,ht[i].weight,ht[i].parent,ht[i].lchild,ht[i].rchild);

}

printf("\n");

}

void Decode()

{

FILE \*fp1,\*fp2;

int p;

char ch;

if((fp1 = fopen("Haffman.code","r")) == NULL){

printf("文件打开失败");

exit(1);

}

if((fp2 = fopen("Haffman.decode","w")) == NULL){

printf("文件打开失败");

exit(1);

}

p = 2\*count-1;

printf("\n\t\t译码内容为： ");

ch = fgetc(fp1);

while(!feof(fp1)){

if(ch == '0')

p = ht[p].lchild;

else if(ch == '1')

p = ht[p].rchild;

if(ht[p].data != 0){

printf("%c",ht[p].data);

fputc(ht[p].data,fp2);

p = 2\*count-1;

}

ch = fgetc(fp1);

}

printf("\n");

fclose(fp1);

fclose(fp2);

//验证解码结果

int i,j=1;

char s1[N],s2[N];

if((fp1 = fopen("Haffman.souce","r")) == NULL){

printf("\n打开文件失败！\n");

exit(1);

}

if((fp2 = fopen("Haffman.decode","r")) == NULL){

printf("\n打开文件失败！\n");

exit(1);

}

for(i = 1; (s2[i]=fgetc(fp2)) != EOF;i++){

}

printf("\n\t\t源文件内容为：");

for(i = 1; (s1[i]=fgetc(fp1)) != EOF;i++){

printf("%c",s1[i]);

}

printf("\n");

while( j < count){

if(s1[j] == s2[j]){

j++;

}

else{

printf("%d",j);

printf("\n\t\t编码失败！\n");

break;

}

}

if(j == count){

printf("\n\t\t编码内容一致，编码成功。\n");

}

fclose(fp1);

fclose(fp2);

}

void test() {

FILE \*fp;

char ch;

str s1[128],s2[128];

for(int i=0;i<128;i++){

s1[i].data = 0;

s1[i].num = 0;

s2[i].data = 0;

s2[i].num = 0;

}

if((fp = fopen("Haffman.souce","r")) == NULL){

exit(1);

}

ch = fgetc(fp);

while(!feof(fp)){

s1[ch].data = ch;

s1[ch].num ++;

ch = fgetc(fp);

}

fclose(fp);

count=1;

for (int i=1;i<128;i++){

if(s1[i].num!=0){

s2[count].data = s1[i].data;

s2[count].num = s1[i].num;

count++;

}

}

count-=1;

for(int i=1;i<=count;i++){

ht[i].data = s2[i].data;

ht[i].weight = s2[i].num;

ht[i].parent = 0;

ht[i].lchild = 0;

ht[i].rchild = 0;

}

printf("\n\n");

//初始化后n-1个空元素

int m = 2\*count-1;

for(int i=count+1;i<=m;i++){

ht[i].data = 0;

ht[i].weight = 0;

ht[i].parent = 0;

ht[i].lchild = 0;

ht[i].rchild = 0;

}

//从第n+1个元素开始构造新结点

int s3,s4;

for(int i=count+1;i<=m;i++){

//在ht的前i-1项中选双亲为0且权值最小的两个节点s1,s2

select(ht,i-1,&s3,&s4);

//建立新结点，赋权值

ht[i].weight = ht[s3].weight + ht[s4].weight;

ht[i].lchild = s3;

ht[i].rchild = s4;

ht[s3].parent = i;

ht[s4].parent = i;

}

}

int main(void)

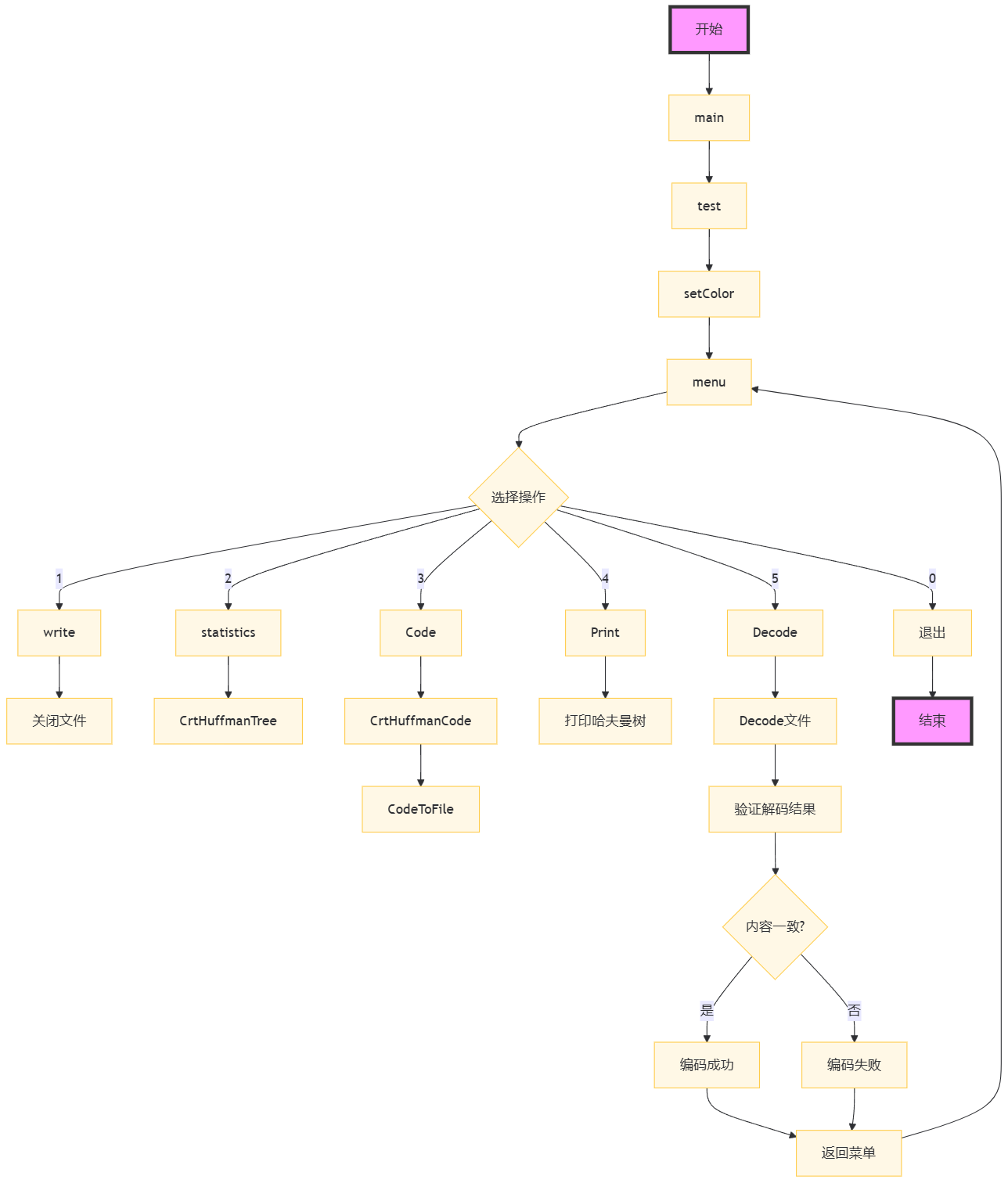
{

test();

setColor();

}

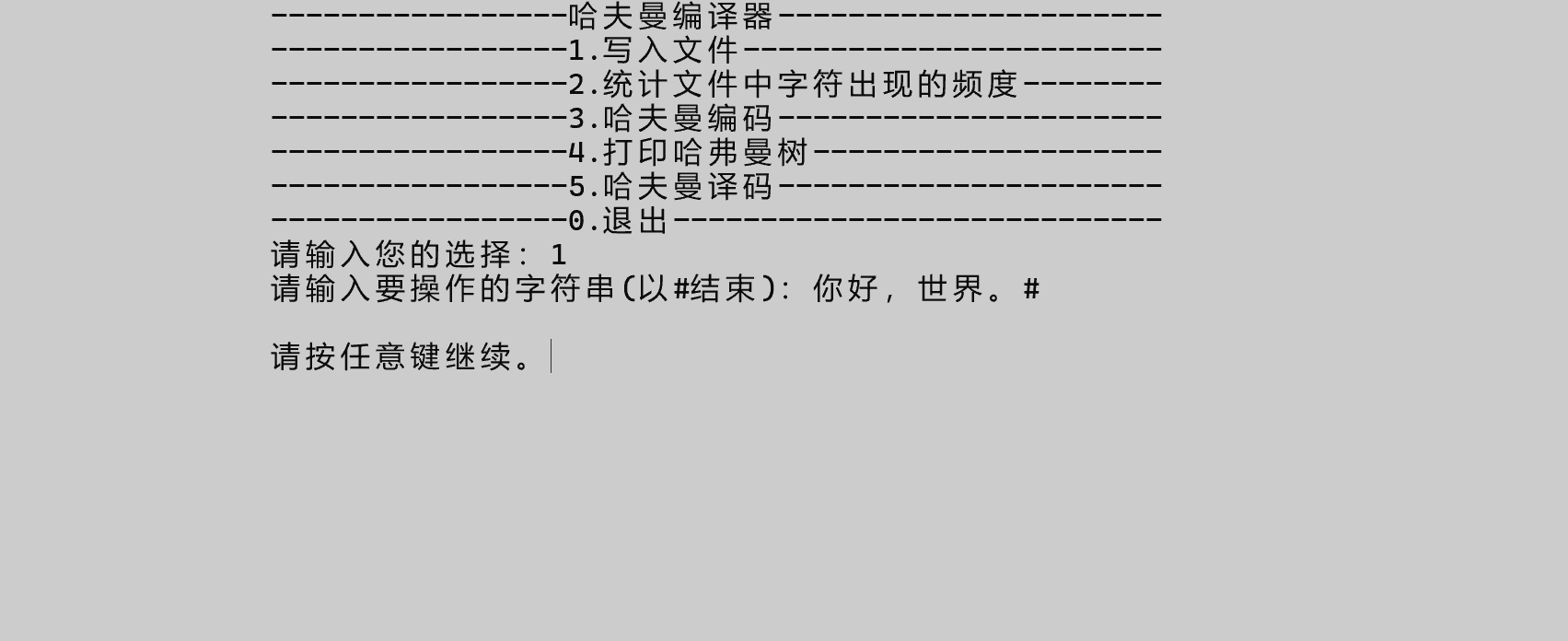
整个程序的流程图：

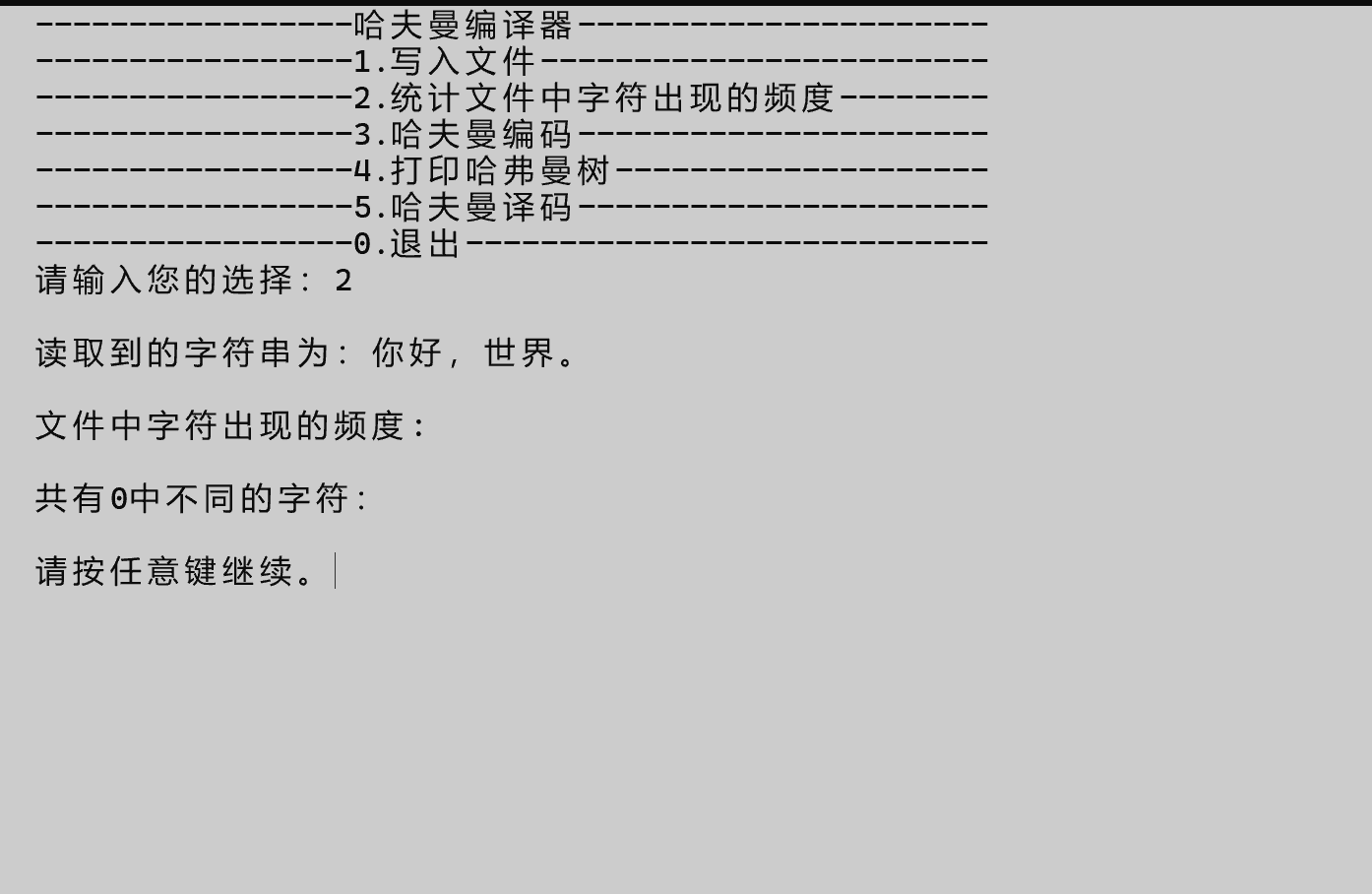


### 2．异常测试数据及运行结果

要求提供2组异常测试数据和运行结果

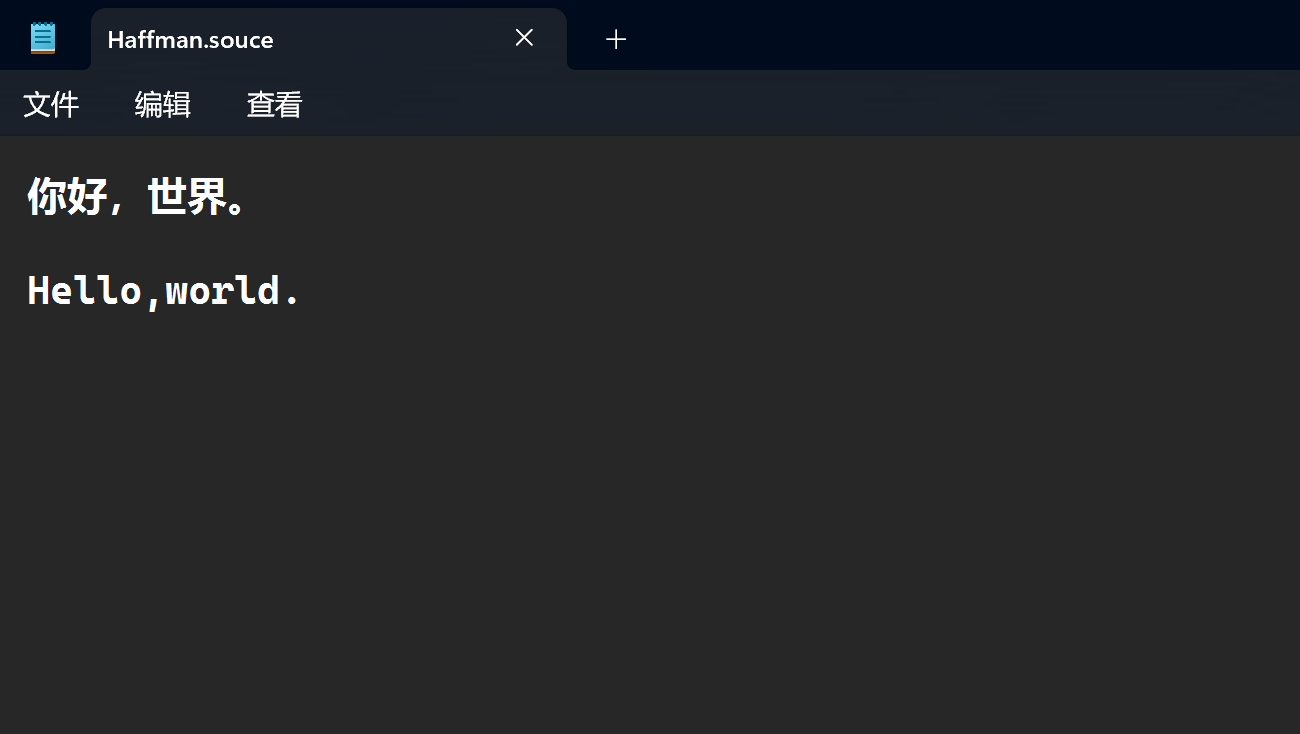
（1）输入汉字无法正确统计：



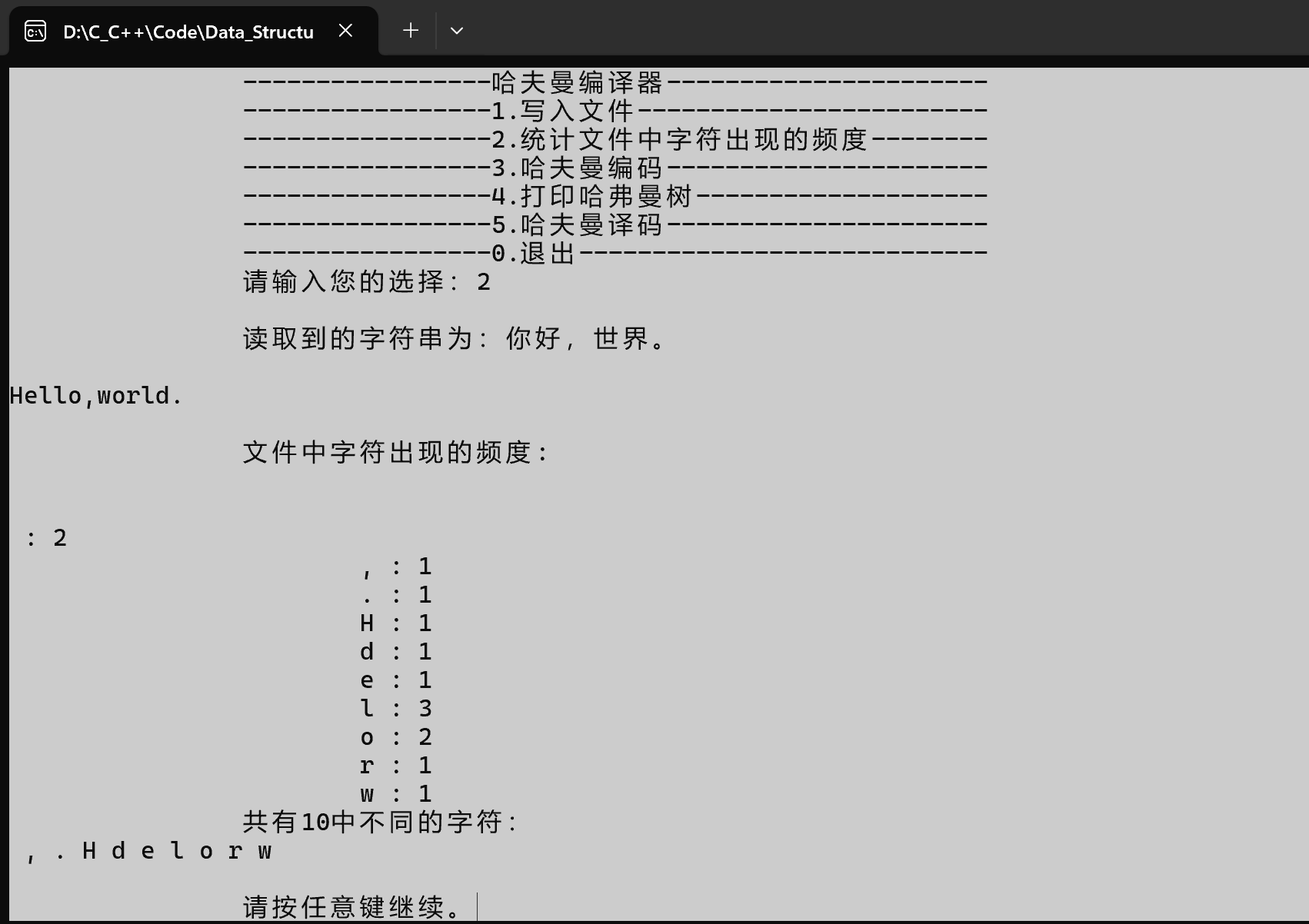


（2）文本中有回车，显示会破坏界面美感：

文件文本内容：



统计字符频度界面：



|  |
| --- |
| 1. 实验总结：   这此数据结构课程设计主要是通过使用哈夫曼树的知识来实现，通过此次课程设计我学习和收获到了很多、也积累了更多的知识以及解决问题的方法。在实验的第一天我就决定了要完成哈夫曼树编译码的任务，我觉得这是个很锻炼人能力的项目，对于压缩解压缩也是非常感兴趣。但是在后来不断实践的过程中我遇见了很多的问题，例如编写代码的思路都正确但是却无法实现功能、编码成功但是无法译码成功等等，但在后面的不断实践下也一一解决，但是仍然有考虑不周全的地方，例如无法正确读取汉字，包括对于回车的处理还不够等等问题，这些是我在此次课设中没有考虑到的地方,也是此次课设的不足之处。包括在此次实验中没有成功实现压缩解压缩的功能 |

说明：

1. **课程名称、课程设计目的、课程设计内容、课程设计要求**由教师确定，实验前由教师事先填好，然后作为实验报告模版供学生使用；
2. **实验条件**由学生在实验或上机之前填写，教师应该在实验前检查并指导；
3. **实验过程**由学生记录实验的过程，包括操作过程、遇到哪些问题以及如何解决等；
4. **实验总结**由学生在实验后填写，总结本次实验的收获、未解决的问题以及体会和建议等；
5. 源程序、代码、具体语句等，若表格空间不足时可作为**附录**另外附页。