

《数据结构与算法分析》课程设计报告

课题名称： 哈夫曼编码

课题设计人（学号）： 李紫萱 2017141461185

指导教师： 朱宏

评阅成绩： _____

评阅意见： _____

提交报告时间：2018 年 12 月 17 日

哈夫曼编码

计算机科学与技术 专业

学生

李紫莹

指导老师

朱宏

【摘要】 在计算机数据处理中，哈夫曼编码使用变长编码表对源符号（如文件中的一个字母）进行编码，其中变长编码表是通过一种评估来源符号出现机率的方法得到的，出现机率高的字母使用较短的编码，反之出现机率低的则使用较长的编码，这便使编码之后的字符串的平均长度、期望值降低，从而达到无损压缩数据的目的。哈夫曼树是带权路径长度最短的树，权值较大的结点离根较近。

关键词：哈夫曼编码 二叉树应用 编码 解码

一.实验名称：**Huffman** 编码（二叉树应用）

二.实验目的和要求：

1. 要求对文件进行 **Huffman** 编码的算法,以及对一编码文件进行解码的算法。
2. 熟练掌握二叉树的应用。

三.实验环境：

1. 硬件环境：16G 内存+250GSSD+1T 机械+校园网

2. 软件环境:

操作系统: windows10

编译系统: Dev-C++

四. 算法描述:

1. 定义哈夫曼树的结构体

```
typedef struct {  
    char ch;  
    float weight;    //权值  
    unsigned int parent,lchild,rchild;  
}HTNode,*activeHuffmanTree; //动态分配数组存储哈夫曼树
```

2. 接下来构造一颗哈夫曼树

```
void      createHuffmanTree(activeHuffmanTree      &HT,  
HuffmanCode HC[], float *w, char *ch,int n)  
//构造哈夫曼树 HT，哈夫曼数的编码存放在 HC 中，w 为 n  
//个字符的权值  
{  
    int i;  
  
    if (n<=1) return;  
    m = 2 * n - 1;  
    HT = (activeHuffmanTree)malloc((m+1) * sizeof(HTNode));  
    for (i=1; i<=n; i++,w++)    //初始化 n 个叶子结点  
    {  
        HT[i].weight=*w;  
        HT[i].ch=ch[i-1];  
        HT[i].parent=0;  
        HT[i].lchild=0;  
        HT[i].rchild=0;
```

```
    }
    for (i=n+1; i<=m; i++) //初始化其余的结点
    {
        HT[i].weight=0;
        HT[i].parent=0;
        HT[i].lchild=0;
        HT[i].rchild=0;
    }

    for (i=n+1; i<=m; i++) //构造哈弗曼树
    {
        Select(&HT, i-1,&s1,&s2); //查找树中权值最小的
        //两个结点
        HT[s1].parent = i; HT[s2].parent = i; //其序号分别为 s1
        //和 s2
        HT[i].lchild = s1; HT[i].rchild = s2;
        HT[i].weight = HT[s1].weight + HT[s2].weight;
    }
}
```

3. 构造完哈夫曼树后，求出从叶子节点到根节点的每个字符的哈夫曼树编码。

```
void      activeHuffmanCoding(activeHuffmanTree      &HT,
HuffmanCode HC[], float *w,char *ch ,int n)
{
    int i;
    unsigned int c,f;
    char *cd;
    int start;
```

```
createHuffmanTree(HT,HC, w, ch,n);
*HC=(HuffmanCode)malloc((n+1)*sizeof(char*));//分配 n 个
//字符编码 的头指针项链
    cd=(char*)malloc(n*sizeof(char));        //为哈弗曼编码动
//态分配空间
    cd[n-1]='\0'; //编码 结束符
    //求 n 个叶子结点的哈弗曼编码
    for(i=1;i<=n;i++)
    {
        start=n-1;          //编码结束符位置
        for(c=i,f=HT[i].parent;f!=0;c=f,f=HT[f].parent)
            //从叶子结点到根结点求编码
            if(HT[f].lchild==c)
                cd[--start]='0';
            else
                cd[--start]='1';
        HC[i]=(char*)malloc((n-start)*sizeof(char));
        //为第 i 个字符编码分配空间
        strcpy(HC[i],&cd[start]);
        //将当前求出结点的哈弗曼编码复制到 HC
    }
    free(cd);//释放空间
}
```

4. 本程序有三部分功能构成，第一部分是动态编码，即输入一串字符，程序给出各个字符的编码结果，实现函数为 activeTest 代码如下：

```
void activeTest(activeHuffmanTree &HT, HuffmanCode HC[])
{
    int n;
```

```
int i;
char *c;
float *w;

puts("\nn=====");
printf("请输入编码字符个数: ");
scanf("%d",&n);
fflush(stdin);
w=(float *)malloc(n*sizeof(float));
c=(char *)malloc(n*sizeof(char));

printf("\n 请依次输入编码的字符串（无需空格）: ");
gets(c);
fflush(stdin);
printf("\n 请依次输入字符权值（权值之间空格隔开）: ");
for(i=0;i<n;i++)
    scanf("%f",&w[i]);
fflush(stdin);
activeHuffmanCoding(HT,HC,w,c,n);
printf("\n 编码完成，结果如下: \n");
puts("\nn=====");
puts("      字符      编码      权 值");
for(i = 0;i <n;i++)
{
printf("%8c      :%-12s      (%-3f)\n",c[i],HC[i+1],w[i]);
}
puts("=====");
}
```

5. 第二个功能是根据给出的字符集频度，对一篇英文文章进行哈

夫曼编码，实现函数为 FileCoding 代码如下：

```
int FileCoding(activeHuffmanTree HT, HuffmanCode HC[])
{
    FILE *fp,*fp1;
    char ch;
    char name[20];
    char c[27]={'
    'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w',
    'x','y','z'};
    float
    w[27]={ 186,64,13,22,32,103,21,15,47,57,1,5,32,20,57,63,15,1,
    48,51,80,23,8,18,1,16,1};
    printf("\n 请输入文件名（测试文件： test.txt）: ");
    scanf("%s",name);
    fflush(stdin);
    if((fp=fopen(name,"r"))==NULL)
    {
        printf(" 文件打开失败！ \n");
        exit(1);
    }
    printf("原文如下： \n");
    puts("=====
    =====\n");
    ch=fgetc(fp);
    while(ch!=EOF)
    {
        putchar(ch);
        ch=fgetc(fp);
    }
    puts("\n=====
```

```
=====\\n");
fclose(fp);

activeHuffmanCoding(HT,HC,w,c,27);

if((fp=fopen(name,"r"))==NULL)
{
    printf(" 文件打开失败！ \\n");
    exit(1);
}
if((fp1=fopen("coding.cod","w"))==NULL)
{
    printf(" 文件打开失败！ \\n");
    exit(1);
}
printf("编码如下： \\n");

puts("=====\\n");
=====\\n");

    ch= tolower(fgetc(fp));
while(ch!=EOF)
{

    if(ch==10)
    {
        putchar(10);
        fputc(10,fp1);
    }
    else
        if(ch==' ')
```



```
        {
            printf("%s",HC[1]);
            fprintf(fp1,"%s",HC[1]);
        }

    else
        if(ch>96&&ch<123)
        {
            printf("%s",HC[ch-95]);
            fprintf(fp1,"%s",HC[ch-95]);
        }

    else
    {
        putchar(ch);
        fputc(ch,fp1);
    }

    ch=tolower(fgetc(fp));
}

fclose(fp1);
puts("\n=====
=====\\n");
puts("文件已保存至\\coding.cod\\\\n");
fclose(fp);
return 0;
}
```

6. 第三个功能为针对哈夫曼编码后的文章进行译码,采用简单的大小写转换功能,实现函数为 FileDecoding,代码如下:

```
int FileDecoding(activeHuffmanTree HT, HuffmanCode HC[])
{
    int p=2*N-1;
    FILE *fp;
    char ch;
    char name[20];
    char c[27]={'
','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w',
'x','y','z'};
    float
w[27]={ 186,64,13,22,32,103,21,15,47,57,1,5,32,20,57,63,15,1,
48,51,80,23,8,18,1,16,1};
    int change=0;
    printf("\n 请输入文件名（测试文件：coding.cod）: ");
    scanf("%s",name);
    fflush(stdin);
    if((fp=fopen(name,"r"))==NULL)
    {
        printf(" 文件打开失败！ \n");
        exit(1);
    }
    printf("原码如下： \n");
    puts("=====
=====\\n");
    ch=fgetc(fp);
    while(ch!=EOF)
    {
        putchar(ch);
        ch=fgetc(fp);
    }
```

```
puts("\n=====
=====\\n");
fclose(fp);
createHuffmanTree(HT,HC,w,c,27);
if((fp=fopen(name,"r"))==NULL)
{
    printf(" 文件打开失败! \\n");
    exit(1);
}
printf("译文如下: \\n");

puts("=====
=====\\n");

    ch= fgetc(fp);
while(ch!=EOF)
{
    if(ch=='0')
        p=HT[p].lchild;
    else
        if(ch=='1')
            p=HT[p].rchild;
    else
        if(ch==10||ch=='.'||ch=='?')
        {
            putchar(ch);
            change=1;
        }
    else
        putchar(ch);
        if(p<=N)
```

```

        {
            if(change)
            {
                printf("%c",toupper(HT[p].ch));
                change=0;
            }
            else
                printf("%c",HT[p].ch);

            p=2*N-1;
        }
        ch=fgetc(fp);
    }
    fclose(fp);
    puts("\n=====
=====\\n");

    return 0;
}

```

7. 为方便操作，我添加了用户菜单功能，实现函数为 `Meum`，代码如下：

```

int Meum()
{
    int choice;
    puts("
\\n\\t=====
=====\\n");

    puts("                === 哈夫曼编/译码器
===\\n ");
}

```



```

        case 1:activeTest(HT,HC); break;
        case 2:FileCoding(HT,HC); break;
        case 3:FileDecoding(HT,HC);break;
        default:exit(0);
    }

    printf("\n\n 任意键继续...");
    getchar();
}

return 0;
}

```

五. 源程序清单

由两部分构成，一个为头文件 huffman.h，另一个为 main.cpp

1. 头文件 huffman.h，代码如下：

```

#include<string.h>
#include<stdlib.h>
#include<stdio.h>
#include<malloc.h>
#include<bits/stdc++.h>
#define infinity 32768      //定义一个无限大的值
#define MAXSIZE 100
#define N 27
typedef struct {           //哈夫曼树的结构体
    char ch;
    float weight;         //权值
    unsigned int parent,lchild,rchild;
}HTNode,*activeHuffmanTree; //动态分配数组存储哈夫曼树

typedef char *HuffmanCode; //动态分配数组存储哈夫曼编码表

```

int m,s1,s2; //定义全局变量总结点 m, 最小结点序号 s1,次小
结点序号 s2

/******

/*函数名: Min */

/*函数功能: 返回树中 n 个结点中权值最小的结点序号 */

/*形参: activeHuffmanTree t,int n */

/*返回值: flag */

/******

int Min(activeHuffmanTree t,int n) //返回树中 n 个结
点中权值最小的结点序号

{

int i,flag;

float f=infinity; //f 为一个无限大的值

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

if(t[i].weight<f&& t[i].parent==0)

{

f=t[i].weight;flag=i;

}

t[flag].parent=1; //给选中的结点的双亲结点赋值

1, 避免再次查找该结点

return flag;

}

/******

/* 函 数 名 : Select

*/

/*函数功能: 在 n 个结点中选择两个权值最小的结点序号,
其中 s1 最小, s2 次小 */

```

/* 形参： activeHuffmanTree *t,int n,int *s1,int *s2
*/
/*      返      回      值      :      s1,s2
*/
/*****
*****/

void Select(activeHuffmanTree *t,int n,int *s1,int *s2)
//在 n 个结点中选择两个权值最小的结点序号,其中 s1 最小,
s2 次小
{
    int x;
    *s1=Min(*t,n);
    *s2=Min(*t,n);
    if((*t)[*s1].weight>(*t)[*s2].weight) //如果序号 s1 的权值
大于序号 s2 的权值,将两者交换,使 s1 最小,s2 次小
    {
        x=*s1;
        *s1=*s2;
        *s2=x;
    }
}

/*****
*****/

/*      函      数      名      :      createHuffmanTree
*/
/*函数功能：构造哈弗曼树 HT,哈弗曼数的编码存放在 HC
中,w 为 n 个字符的权值 */
/*形参： activeHuffmanTree &HT, HuffmanCode HC[], float
*w, char *ch,int n */

```



```
/*      返      回      值      :      无
*/

/*****
*****/

void      createHuffmanTree(activeHuffmanTree      &HT,
HuffmanCode HC[], float *w, char *ch,int n)
//构造哈弗曼树 HT，哈弗曼数的编码存放在 HC 中，w 为 n
个字符的权值
{
int i;

if (n<=1) return;
m = 2 * n - 1;
HT = (activeHuffmanTree)malloc((m+1) * sizeof(HTNode));
for (i=1; i<=n; i++,w++)      //初始化 n 个叶子结点
{
HT[i].weight=*w;
HT[i].ch=ch[i-1];
HT[i].parent=0;
HT[i].lchild=0;
HT[i].rchild=0;
}
for (i=n+1; i<=m; i++)      //初始化其余的结点
{
HT[i].weight=0;
HT[i].parent=0;
HT[i].lchild=0;
HT[i].rchild=0;
}
```

```

for (i=n+1; i<=m; i++)    //构造哈弗曼树
{
    Select(&HT, i-1,&s1,&s2);    //查找树中权值最小的两个
    结点
    HT[s1].parent = i; HT[s2].parent = i; //其序号分别为 s1 和
    s2
    HT[i].lchild = s1; HT[i].rchild = s2;
    HT[i].weight = HT[s1].weight + HT[s2].weight;

}
}

/*****
*****/

/*    函    数    名    :    activeHuffmanCoding
*/

/*函数功能: 从叶子结点到根结点求每个字符的哈弗曼树编
码
*/

/*形参: activeHuffmanTree &HT, HuffmanCode HC[], float
*w,char *ch ,int n */

/*    返    回    值    :    无
*/

/*****
*****/

void    activeHuffmanCoding(activeHuffmanTree    &HT,
HuffmanCode HC[], float *w,char *ch ,int n)
{
    int i;
    unsigned int c,f;
    char *cd;
    int start;

```

```

createHuffmanTree(HT,HC, w, ch,n);

//从叶子结点到根结点求每个字符的哈弗曼树编码
*HC=(HuffmanCode)malloc((n+1)*sizeof(char*)); //分配 n 个
字符编码 的头指针项链
cd=(char*)malloc(n*sizeof(char));           //为哈 弗 曼 编 码 动
态分配空间
cd[n-1]='\0'; //编码 结束符
//求 n 个叶子结点的哈弗曼编码
for(i=1;i<=n;i++)
{
    start=n-1;           //编码结束符位置
    for(c=i,f=HT[i].parent;f!=0;c=f,f=HT[f].parent)
        //从叶子结点到根结点求编码
        if(HT[f].lchild==c)
            cd[--start]='0';
        else
            cd[--start]='1';
    HC[i]=(char*)malloc((n-start)*sizeof(char));
    //为第 i 个字符编码分配空间
    strcpy(HC[i],&cd[start]);
    //将当前求出结点的哈弗曼编码复制到 HC
}
free(cd); //释放空间
}

/*****
****/

/*      函      数      名      :      activeTest
*/

```

```
/*函数功能：动态实现哈弗曼树以及字符和权值的输入输出
*/
/* 形 参： activeHuffmanTree  &HT, HuffmanCode  HC[]
*/
/*          返          回          值          :          无
*/
/*****
*****/
void activeTest(activeHuffmanTree &HT, HuffmanCode HC[])
{
    int n;
    int i;
    char *c;
    float *w;

    puts("\n=====");
    printf("请输入编码字符个数: ");
    scanf("%d",&n);
    fflush(stdin);
    w=(float *)malloc(n*sizeof(float));
    c=(char *)malloc(n*sizeof(char));

    printf("\n 请依次输入编码的字符串（无需空格）: ");
    gets(c);
    fflush(stdin);
    printf("\n 请依次输入字符权值（权值之间空格隔开）: ");
    for(i=0;i<n;i++)
        scanf("%f",&w[i]);
    fflush(stdin);
    activeHuffmanCoding(HT,HC,w,c,n);
}
```

```

printf("\n 编码完成，结果如下： \n");
puts("\n=====");
puts("      字符      编码      权 值");
    for(i = 0;i <n;i++)
    {

printf("%8c      :%-12s      (%-3f)\n",c[i],HC[i+1],w[i]);
    }
puts("=====");
}

/*****
*****/

/*      函      数      名      :      FileCoding
*/

/*函数功能：根据给出的字符集频度，对一篇英文文章进行
哈夫曼编码；*/

/* 形 参 :   activeHuffmanTree  HT,  HuffmanCode  HC[]
*/

/*      返      回      值      :      无
*/

/*****
*****/

int FileCoding(activeHuffmanTree HT, HuffmanCode HC[])
{
    FILE *fp,*fp1;
    char ch;
    char name[20];
    char                                     c[27]='
', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w

```

```
'x','y','z'};
float
w[27]={ 186,64,13,22,32,103,21,15,47,57,1,5,32,20,57,63,15,1
,48,51,80,23,8,18,1,16,1};
printf("\n 请输入文件名 (测试文件: test.txt): ");
scanf("%s",name);
fflush(stdin);
if((fp=fopen(name,"r"))==NULL)
{
    printf(" 文件打开失败! \n");
    exit(1);
}
printf("原文如下: \n");
puts("=====
=====\\n");
    ch=fgetc(fp);
while(ch!=EOF)
{
    putchar(ch);
    ch=fgetc(fp);
}
puts("\\n=====
=====\\n");
fclose(fp);

activeHuffmanCoding(HT,HC,w,c,27);

if((fp=fopen(name,"r"))==NULL)
{
    printf(" 文件打开失败! \n");
```

```
        exit(1);
    }
    if((fp1=fopen("coding.cod","w"))==NULL)
    {
        printf(" 文件打开失败! \n");
        exit(1);
    }
    printf("编码如下: \n");

    puts("=====  

=====\\n");
    ch= tolower(fgetc(fp));
    while(ch!=EOF)
    {

        if(ch==10)
        {
            putchar(10);
            fputc(10,fp1);
        }
        else
            if(ch==' ')
            {
                printf("%s",HC[1]);
                fprintf(fp1,"%s",HC[1]);
            }

        else
            if(ch>96&&ch<123)
            {
```

```

        printf("%s",HC[ch-95]);
        fprintf(fp1,"%s",HC[ch-95]);
    }

    else
    {
        putchar(ch);
        fputc(ch,fp1);
    }

    ch=tolower(fgetc(fp));
}

fclose(fp1);
puts("\n=====
=====\\n");

puts("文件已保存至\\coding.cod\\n");
fclose(fp);
return 0;
}

/*****
*****/

/*      函      数      名      :      FileDecoding
*/

/*函数功能：针对哈夫曼编码后的文章进行译码，采用简单
的大小写转换功能 */

/* 形 参 :   activeHuffmanTree  HT,  HuffmanCode  HC[]
*/

/*      返      回      值      :      无
*/

/*****
*****/

```



```
*****/

int FileDecoding(activeHuffmanTree HT, HuffmanCode HC[])
{
    int p=2*N-1;
    FILE *fp;
    char ch;
    char name[20];
    char c[27]={'
','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w',
','x','y','z'};
    float
w[27]={ 186,64,13,22,32,103,21,15,47,57,1,5,32,20,57,63,15,1
,48,51,80,23,8,18,1,16,1};
    int change=0;
    printf("\n 请输入文件名（测试文件：coding.cod）: ");
    scanf("%s",name);
    fflush(stdin);
    if((fp=fopen(name,"r"))==NULL)
    {
        printf(" 文件打开失败！ \n");
        exit(1);
    }
    printf("原码如下： \n");
    puts("=====
=====\\n");

    ch=fgetc(fp);
    while(ch!=EOF)
    {
        putchar(ch);
        ch=fgetc(fp);
    }
}
```

```
    }
    puts("\n=====
=====\\n");
    fclose(fp);

    createHuffmanTree(HT,HC,w,c,27);

    if((fp=fopen(name,"r"))==NULL)
    {
        printf(" 文件打开失败! \\n");
        exit(1);
    }
    printf("译文如下: \\n");

    puts("=====
=====\\n");

    ch= fgetc(fp);

    while(ch!=EOF)
    {
        if(ch=='0')
            p=HT[p].lchild;
        else
            if(ch=='1')
                p=HT[p].rchild;
        else
            if(ch==10||ch=='.'||ch=='?')
            {
                putchar(ch);
                change=1;
            }
    }
```

```
        }
    else
        putchar(ch);

        if(p<=N)
        {
            if(change)
            {
                printf("%c",toupper(HT[p].ch));
                change=0;
            }
            else
                printf("%c",HT[p].ch);

            p=2*N-1;
        }
        ch=fgetc(fp);
    }

    fclose(fp);

    puts("\n=====
=====\\n");

    return 0;

}

/*****
/*      函      数      名      :      Meum
*/
*/
```

```

/*函数功能：系统功能菜单 */
/*形参：无 */
/*返回值：choice */
/******/

int Meum()
{
    int choice;

    puts("
\n\t=====
=====\\n");

    puts("          === 哈夫曼编/译码器
===\\n   ");

    puts("
\t=====
=====\\n");

    printf("      \t\t\t1-----动态哈夫曼编码\\n\\n");
    printf("      \t\t\t2-----哈夫曼编码       \\n\\n");
    printf("      \t\t\t3-----哈夫曼译码       \\n\\n");
    printf("      \t\t\t4-----退出           \\n\\n");
    puts("
\t=====
=====\\n");

    printf("\\t\t\t\t请选择: ");
    scanf("%d",&choice);
    fflush(stdin);
    return choice;
}

/*****/
/*函数名：SystemInfo */

```

```

/*函数功能：显示函数功能系统信息 */
/*形参：无 */
/*返回值：无 */
/*****/

void SystemInfo()
{
    FILE *fp;
    if(!(fp=fopen("系统信息.txt","r")))
    {
        puts("说明文件丢失！");
        return ;
    }
    puts("\n\n\n\n");
    puts("-----");
    puts("==== 系统信息 ===");
    puts("\n");
    puts("-----");
    puts("-----");
    while(!feof(fp))
        putchar(fgetc(fp));

    putchar(10);
    puts("-----");
    puts("-----");
}

```

2. main.cpp

```

#include"huffman.h"

/*****

```

```

*****/

/*      函      数      名      :      main
*/

/*函数功能：针对提供的系统功能菜单中用户的选择调用各
个函数 */

/*      形      参      :      无
*/

/*      返      回      值      :      无
*/

/*****
*****/

int main()
{
    int flag=1;
    activeHuffmanTree HT;
    HuffmanCode *HC;
    HC = (HuffmanCode
*)malloc(MAXSIZE*sizeof(HuffmanCode));
    while((flag=Meum())<4)
    {
        switch(flag)
        {
            case 0:SystemInfo( ); break;
            case 1:activeTest(HT,HC); break;
            case 2:FileCoding(HT,HC); break;
            case 3:FileDecoding(HT,HC);break;
            default:exit(0);
        }

        printf("\n\n 任意键继续...");
        getchar();
    }
}

```

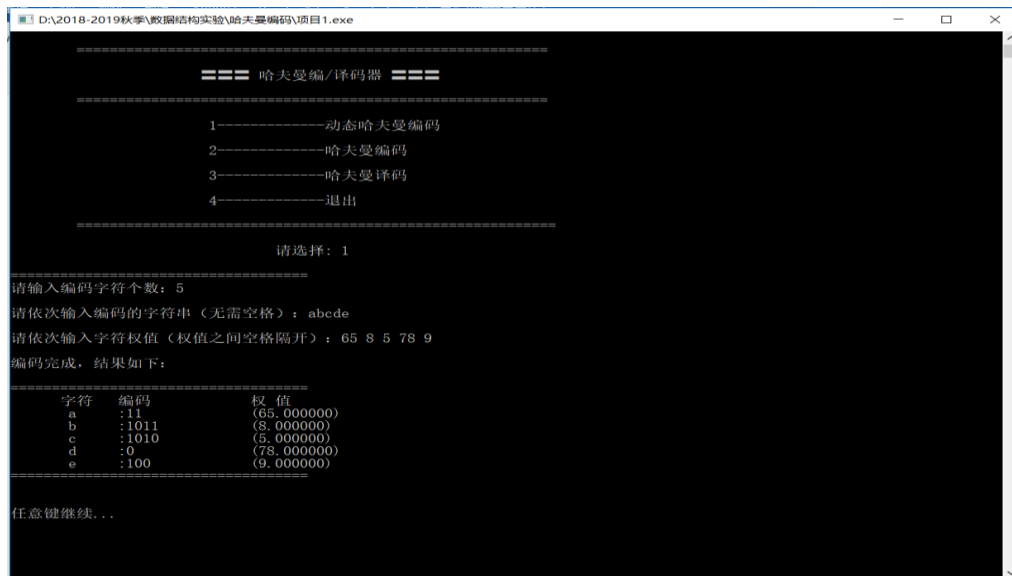
```

    }
    return 0;
}

```

六. 运行结果

1. 动态编码:



```

===== 哈夫曼编/译码器 =====
1-----动态哈夫曼编码
2-----哈夫曼编码
3-----哈夫曼译码
4-----退出

请选择: 1

请输入编码字符个数: 5
请依次输入编码的字符串(无需空格): abcde
请依次输入字符权值(权值之间空格隔开): 65 8 5 78 9
编码完成, 结果如下:

=====
字符  编码      权  值
a     :11      (65.000000)
b     :1011    (8.000000)
c     :1010    (5.000000)
d     :0       (78.000000)
e     :100     (9.000000)
=====

任意键继续...

```

2. 从文件读入编码, 并将编码结果储存于 coding.cod



```

=====
1-----动态哈夫曼编码
2-----哈夫曼编码
3-----哈夫曼译码
4-----退出

请选择: 2

请输入文件名(测试文件: test.txt): test.txt
原文如下:
=====
aadataaraefrtaafter
=====

编码如下:
=====
101010101010101010101010100010100010110011001100110101010101001111010100010
=====

文件已保存至"coding.cod"

```

3. 从 coding.cod 中译码

```
Da\2018-2019秋季\数据结构实验\哈夫曼编码\项目1.exe
1010101010110101011011010001010100101100110010110110101011001111010100010
文件已保存至"coding.cod"
任意键继续...

===== 哈夫曼编/译码器 =====
1-----动态哈夫曼编码
2-----哈夫曼编码
3-----哈夫曼译码
4-----退出

请选择: 3
请输入文件名(测试文件: coding.cod): coding.cod
原码如下:
1010101010110101011011010001010100101100110010110110101011001111010100010
译文如下:
aadataaefrtaafter
任意键继续...
```

七. 实验运行情况分析

1. 动态编码：输入字符和各字符权值，输出各个字符的编码和权值。经检验，结果正确。
2. 从 text.txt 读入文本，输出编码，并将编码文件保存在 coding.cod 中。
3. 从 coding.cod 的编码中译码，与之前文本对应相符。

八. 程序不足分析

动态编码时，需要用户输入字符个数，灵活性比较低，输入错误后也不会报错，比较不人性化。希望接下来可以改进的更好。

参考文献

- [1] 唐宁九, 游洪跃, 孙界平、朱宏、杨秋辉. 数据结构与算法教程实验和课程设计 (C++版), 2012