```
#include<string.h>
#include<stdlib.h>
#include<stdio.h>

int m,s1,s2;

typedef struct {
unsigned int weight;
unsigned int parent,lchild,rchild;
}HTNode,*HuffmanTree; //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

void Select(HuffmanTree HT,int n) {
int i,j,tmp;
for(i = 1;i <= n;i++)
if(!HT[i].parent){s1 = i;break;}
for(j = i+1;j <= n;j++)
if(!HT[j].parent){s2 = j;break;}
for(i = 1;i <= n;i++)
if((HT[s1].weight>HT[i].weight)&&(!HT[i].parent)&&(s2!=i))s1=i;
for(j = 1;j <= n;j++)
if((HT[s2].weight>HT[j].weight)&&(!HT[j].parent)&&(s1!=j))s2=j;
if (HT[s1].weight>HT[s2].weight) {tmp=s1;s1=s2;s2=tmp;}
}

void HuffmanCoding(HuffmanTree &HT, HuffmanCode HC[], int *w, int n) {
// w 存放 n 个字符的权值(均>0)，构造哈夫曼树 HT，
//  并求出 n 个字符的哈夫曼编码 HC
int i, j;
char *cd;
int p;
int cdlen;

if (n<=1) return;
m = 2 * n - 1;
HT = (HuffmanTree)malloc((m+1) * sizeof(HTNode)); // 0 号单元未用
```

```
for (i=1; i<=n; i++) { //初始化
HT[i].weight=w[i-1];
HT[i].parent=0;
HT[i].lchild=0;
HT[i].rchild=0;
}
for (i=n+1; i<=m; i++) { //初始化
HT[i].weight=0;
HT[i].parent=0;
HT[i].lchild=0;
HT[i].rchild=0;
}
puts("\n 哈夫曼树的构造过程如下所示：");
printf("HT 初态:\n  结点  weight   parent   lchild     rchild");
for (i=1; i<=m; i++)
printf("\n%4d%8d%8d%8d%8d",i,HT[i].weight,
HT[i].parent,HT[i].lchild, HT[i].rchild);
for (i=n+1; i<=m; i++) { //  建哈夫曼树
//  在 HT[1..i-1]中选择 parent 为 0 且 weight 最小的两个结点，
//  其序号分别为 s1 和 s2。
Select(HT, i-1);
HT[s1].parent = i; HT[s2].parent = i;
HT[i].lchild = s1; HT[i].rchild = s2;
HT[i].weight = HT[s1].weight + HT[s2].weight;
printf("\nselect: s1=%d s2=%d\n", s1, s2);
printf("  结点      weight   parent   lchild     rchild");
for (j=1; j<=i; j++)
printf("\n%4d%8d%8d%8d%8d",j,HT[j].weight,
HT[j].parent,HT[j].lchild, HT[j].rchild);

}

//------无栈非递归遍历哈夫曼树，求哈夫曼编码
cd = (char *)malloc(n*sizeof(char)); //  分配求编码的工作空间
p = m; cdlen = 0;
for (i=1; i<=m; ++i) //  遍历哈夫曼树时用作结点状态标志
```

```
HT[i].weight = 0;
while (p) {
if (HT[p].weight==0) { //  向左
HT[p].weight = 1;
if (HT[p].lchild != 0) { p = HT[p].lchild; cd[cdlen++] ='0'; }
else if (HT[p].rchild == 0) { //  登记叶子结点的字符的编码
HC[p] = (char *)malloc((cdlen+1) * sizeof(char));
cd[cdlen] ='\0'; strcpy(HC[p], cd); //  复制编码(串)
}
} else if (HT[p].weight==1) { //  向右
HT[p].weight = 2;
if (HT[p].rchild != 0) { p = HT[p].rchild; cd[cdlen++] ='1'; }
} else { // HT[p].weight==2，退回退到父结点，编码长度减 1
HT[p].weight = 0; p = HT[p].parent; --cdlen;
}
}
} // HuffmanCoding
int main() {
HuffmanTree HT;HuffmanCode *HC;int *w,n,i;
puts("输入结点数:");
scanf("%d",&n);
HC = (HuffmanCode *)malloc(n*sizeof(HuffmanCode));
w = (int *)malloc(n*sizeof(int));
printf("输入%d 个结点的权值\n",n);
for(i = 0;i <n;i++)
scanf("%d",&w[i]);
HuffmanCoding(HT,HC,w,n);
puts("\n 各结点的哈夫曼编码:");
for(i = 1;i <= n;i++)
printf("%2d(%4d):%s\n",i,w[i-1],HC[i]);
return 0;
//getchar();
}
```