## 5-14

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#define N 10

#define M 2\*N-1

#define EPS 1e-5

typedef char datatype;

typedef struct{

float weight;

datatype data;

int lchild,rchild,parent;

}hufmtree;

hufmtree tree[M];

void Huffman(hufmtree[]){

int i,j,p1,p2;

int m=8+21+37+24+6+18+23+41+56+14;

char ch;

float small1,small2,f;

for(i=0;i<M;i++){

tree[i].parent=-1;

tree[i].lchild=-1;

tree[i].rchild=-1;

tree[i].weight=0.0;

tree[i].data='0';

}

/\* for(i=0;i<N;i++){

scanf("%f ",&f);

tree[i].weight=f;

scanf("%c",&ch);

tree[i].data=ch;

}

\*/

tree[0].weight=8.0/m;

tree[0].data='a';

tree[1].weight=21.0/m;

tree[1].data='b';

tree[2].weight=37.0/m;

tree[2].data='c';

tree[3].weight=24.0/m;

tree[3].data='d';

tree[4].weight=6.0/m;

tree[4].data='e';

tree[5].weight=18.0/m;

tree[5].data='f';

tree[6].weight=23.0/m;

tree[6].data='g';

tree[7].weight=41.0/m;

tree[7].data='h';

tree[8].weight=56.0/m;

tree[8].data='i';

tree[9].weight=14.0/m;

tree[9].data='j';

for(i=N;i<M;i++){

p1=p2=0;

small1=small2=100;

for(j=0;j<i;j++){

if(tree[j].parent==-1){

if(tree[j].weight-small1<EPS){

small2=small1;

small1=tree[j].weight;

p2=p1;

p1=j;

}

else if(tree[j].weight-small2<EPS){

small2=tree[j].weight;

p2=j;

}

}

}

tree[p1].parent=i;

tree[p2].parent=i;

tree[i].lchild=p1;

tree[i].rchild=p2;

tree[i].parent=-1;

tree[i].weight=tree[p1].weight+tree[p2].weight;

}

}

typedef struct{

char bits[N];

int start;

datatype data;

}codetype;

codetype code[N];

//哈夫曼编码

void HuffmanCode(codetype code[],hufmtree tree[]){

int i,c,p;

codetype cd;

for(i=0;i<N;i++){

cd.start=N;

c=i;

p=tree[c].parent;

cd.data=tree[c].data;

while(p!=-1){

cd.start--;

if(tree[p].lchild==c)

cd.bits[cd.start]='0';

else

cd.bits[cd.start]='1';

c=p;

p=tree[c].parent;

}

code[i]=cd;

}

}

void main(){

int i,j;

Huffman(tree);

HuffmanCode(code,tree);

for(i=0;i<N;i++){

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

for(j=code[i].start;j<N;j++){

printf("%c",code[i].bits[j]);

}

printf("\n");

}

}

结果为：a:10001 b:1101 c:101 d:001 e:10000 f:1100 g:000 h:111 i:01 j:1001

## 5-15见尾页

## 5-17

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#define MAXSIZE 1024

typedef char datatype;

typedef struct node{

datatype data;

struct node \*lchild,\*rchild;

}bitree;

bitree \*root;

//建立二叉树

bitree\* CreateTree(){

char ch;

bitree \*Q[MAXSIZE];

int front,rear;

bitree \*root,\*s;

root=NULL;

front=1,rear=0;

while((ch=getchar())!='#'){

s=NULL;

if(ch != '@'){

s=(bitree\*)malloc(sizeof(bitree));

s->data=ch;

s->lchild=NULL;

s->rchild=NULL;

}

rear++;

Q[rear]=s;

if(rear==1)

root=s;

else{

if(s && Q[front]){

if(rear%2==0)

Q[front]->lchild=s;

else

Q[front]->rchild=s;

}

if(rear%2==1)

front++;

}

}

return root;

}

//非递归先序遍历

void PreOrder(bitree \*p){

bitree\* stack[MAXSIZE];

bitree \*s=p;

int top=-1;

while(top!=-1 || s!=NULL){

while(s!=NULL){

if(top == MAXSIZE-1){

printf("overflow");

return;

}

else{

top++;

stack[top]=s;

printf("%c",s->data);

s=s->lchild;

}

}

s=stack[top]->rchild;

top--;

}

}

void main(){

bitree \*L;

L=CreateTree();

PreOrder(L);

}

## 5-21

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#define MAXSIZE 1024

typedef char datatype;

typedef struct node{

datatype data;

struct node \*lchild,\*rchild;

}bitree;

bitree \*root;

//建立二叉树

bitree\* CreateTree(){

char ch;

bitree \*Q[MAXSIZE];

int front,rear;

bitree \*root,\*s;

root=NULL;

front=1,rear=0;

while((ch=getchar())!='#'){

s=NULL;

if(ch != '@'){

s=(bitree\*)malloc(sizeof(bitree));

s->data=ch;

s->lchild=NULL;

s->rchild=NULL;

}

rear++;

Q[rear]=s;

if(rear==1)

root=s;

else{

if(s && Q[front]){

if(rear%2==0)

Q[front]->lchild=s;

else

Q[front]->rchild=s;

}

if(rear%2==1)

front++;

}

}

return root;

}

//先序遍历

void PreOrder(bitree \*p){

if(p!=NULL){

printf("%c",p->data);

PreOrder(p->lchild);

PreOrder(p->rchild);

}

}

//交换左右子树

void Exchange(bitree \*p){

bitree \*s=p;

bitree \*r;

if(s!=NULL){

if(s->lchild !=NULL || s->rchild !=NULL){

r=s->lchild;

s->lchild=s->rchild;

s->rchild=r;

Exchange(s->lchild);

Exchange(s->rchild);

}

return;

}

}

void main(){

bitree \*L;

L=CreateTree();

PreOrder(L);

Exchange(L);

printf("\n");

PreOrder(L);

}

## 5-15

