#include<stdio.h>

#include<malloc.h>

#define MaxSize 100

typedef struct BTBNode

{

char data;

struct BTBNode\* left, \* right;

}BTBNode;

typedef struct BTNode

{

BTBNode\* node;

int parent;

}BTNode;

//创建二叉树

void Create(BTBNode\*& s, char\* p)

{

BTBNode\* str[MaxSize];

BTBNode\* r;

r = (BTBNode\*)malloc(sizeof(BTBNode));

int top = -1;

int k = 0;

s = NULL;

while (\*p != '\0')

{

switch (\*p)

{

case '(':

str[++top] = r;

k = 1;

break;

case ')':

top--;

break;

case ',':

k = 2;

break;

default:

r = (BTBNode\*)malloc(sizeof(BTBNode));

r->data = \*p;

r->left = r->right = NULL;

if (s == NULL)

{

s = r;

}

switch (k)

{

case 1:

str[top]->left = r;

break;

case 2:

str[top]->right = r;

break;

}

}

p++;

}

}

//输出结点个数(先序遍历)

int yezi(BTBNode\* s)

{

BTBNode\* str[MaxSize];

BTBNode\* p;

int sum = 0;

int top = -1;

str[++top] = s;

while (top > -1)

{

p = str[top--];

sum++;

if (p->right != NULL)

{

str[++top] = p->right;

}

if (p->left != NULL)

{

str[++top] = p->left;

}

}

return sum;

}

//输出叶子结点个数(中序遍历)

int yezidian(BTBNode\* s)

{

BTBNode\* str[MaxSize];

int top = -1;

int sum=0;

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

{

while (s != NULL)

{

str[++top] = s;

s = s->left;

}

s = str[top--];

if (s->right == NULL&&s->left==NULL)

{

sum++;

}

s = s->right;

}

return sum;

}

//指点字符的层数(用层次遍历)

int cengsu(BTBNode\* s, char e)

{

BTNode str[MaxSize];

BTBNode\* p;

int front, rear;

front = rear = -1;

str[++rear].node = s;

str[rear].parent = -1;

int sum=0;

int k;

while (front != rear)

{

p = str[++front].node;

k = front;

if (p->data == e)

{

break;

}

if (p->left != NULL)

{

str[++rear].node = p->left;

str[rear].parent = k;

}

if (p->right != NULL)

{

str[++rear].node = p->right;

str[rear].parent = k;

}

}

while (str[k].parent != -1)

{

sum++;

k = str[k].parent;

}

sum++;

return sum;

}

//求宽度(用层次遍历)

int length(BTBNode\* s)

{

BTNode str[MaxSize];

BTBNode\* p;

int front, rear;

front = rear = -1;

str[++rear].node = s;

str[rear].parent =0;

int sum = 0;

int k;

while (front != rear)

{

p = str[++front].node;

k = front;

if (p->left != NULL)

{

str[++rear].node = p->left;

str[rear].parent = str[front].parent - 1;

}

if (p->right != NULL)

{

str[++rear].node = p->right;

str[rear].parent = str[front].parent + 1;

}

}

int min = 0, max = 0;

for (int t = 0;t < rear;t++)

{

if (str[t].parent >= max)

{

max = str[t].parent;

}

if (str[t].parent <= min)

{

min = str[t].parent;

}

}

return max - min+1;

}

int main()

{

char a[] = "A(B(D,E(H(J,K(L,M(,N))))),C(F,G(,I)))";

int sum,sum\_1,sum\_2,sum\_3;

BTBNode\* s;

Create(s, a);

sum = yezi(s);

printf("结点个数:%d\n", sum);

sum\_1=yezidian(s);

printf("叶子结点个数:%d\n", sum\_1);

sum\_2 = cengsu(s, 'B');

printf("结点的层数:%d\n", sum\_2);

sum\_3 = length(s);

printf("树的宽度:%d\n", sum\_3);

return 0;

}

