#include<stdio.h>

#include<malloc.h>

#define MaxSize 100

typedef struct Lnode {

char data;

struct Lnode\* lchile, \* rchile;

}Lnode;

typedef struct {

int h;

Lnode\* bb;

}getvalue;

//创建二叉树

void CreateLnode(Lnode\*& s, char\* p)

{

Lnode\* str[MaxSize], \* r;

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

int top = -1, k = 0;

s = NULL;

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

{

switch (\*p)

{

case '(':

k = 1;

top++;

str[top] = r;

break;

case ',':

k = 2;

break;

case ')':

top--;

break;

default:

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

r->lchile = r->rchile = NULL;

r->data = \*p;

if (s == NULL)

{

s = r;

}

switch (k)

{

case 1:

str[top]->lchile = r;

break;

case 2:

str[top]->rchile = r;

}

}

p++;

}

}

//求树的高度

int BTHeight(Lnode\*& s)

{

int height\_1, height\_2;

if (s == NULL)

{

return 0;

}

else

{

height\_1 = BTHeight(s->lchile);

height\_2 = BTHeight(s->rchile);

return (height\_1 > height\_2 ? height\_2+1:height\_1+1);

}

}

//改进的求树高度

getvalue BTHeight\_t(Lnode\* s)

{

getvalue a, b,c;

if (s == NULL)

{

c.h = 0;

return c;

}

else

{

a = BTHeight\_t(s->lchile);

a.bb = s->lchile;

b = BTHeight\_t(s->rchile);

b.bb = s->rchile;

int t = (a.h > b.h) ? b.h + 1 : a.h + 1;

if (t == a.h + 1)

{

a.h = t;

return a;

}

else

{

b.h = t;

return b;

}

}

}

int main() {

char p[] = "A(B(D,E),C(,F))";

char a[] = "1(2,3)";

Lnode\* b1,\*b2;

CreateLnode(b1, p);

CreateLnode(b2, a);

getvalue h;

h = BTHeight\_t(b1);

if (h.bb->lchile == NULL)

{

h.bb->lchile = b2;

}

else

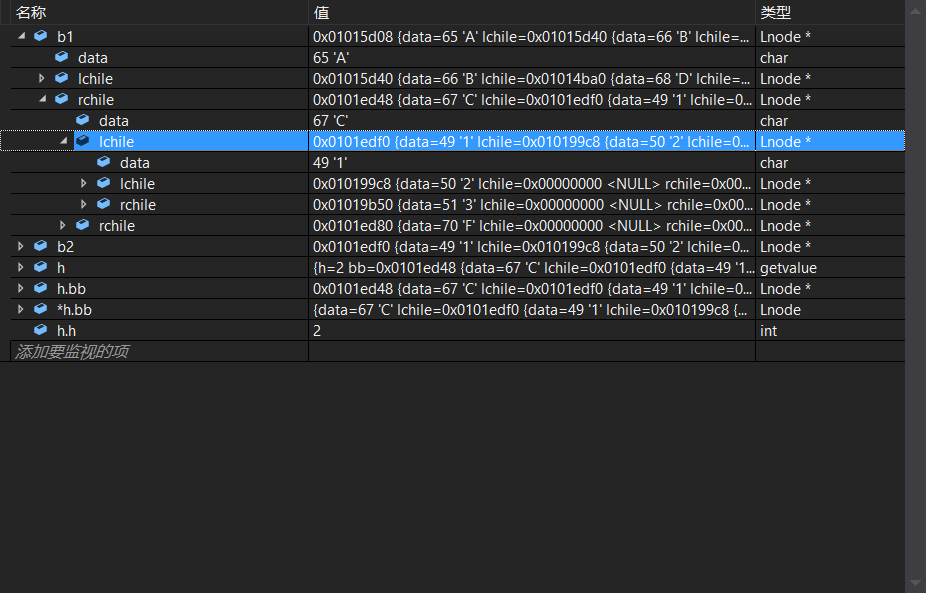
{

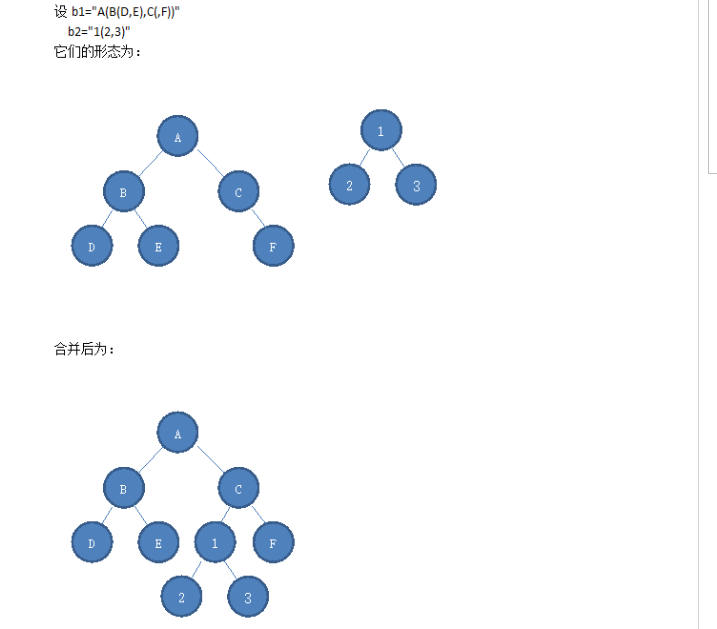
h.bb->rchile = b2;

}

return 0;

}





调试结果显示，c的左孩子跟b2连起来了，连接成功。