本实例只有一个文件

copybtree.c

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  /\*  二叉树的复制  \*/  //定义二叉树节点结构体  typedef struct BinaryNode  {  char ch;  struct BinaryNode\* lChild;  struct BinaryNode\* rChild;  }BinaryNode;  //遍历二叉树  void Recursion(BinaryNode\* root)  {  //判断二叉树 是否为空  if (root == NULL)  {  return;  }  printf("%c", root->ch);  Recursion(root->lChild);  Recursion(root->rChild);  }  //拷贝功能函数  BinaryNode\* CopyBtree(BinaryNode\* root)  {  if (root == NULL)  {  return NULL;  }  //拷贝左子树  BinaryNode\* lchild = CopyBtree(root->lChild);  //拷贝右子树  BinaryNode\* rchild = CopyBtree(root->rChild);  //创建节点  BinaryNode\* newroot = (BinaryNode\*)malloc(sizeof(BinaryNode));  //实现拷贝  newroot->ch = root->ch;  newroot->lChild = lchild;  newroot->rChild = rchild;  return newroot;  }  //释放二叉树内存  void FreeBinaryTree(BinaryNode\* root)  {  if (root == NULL)  {  return;  }  //释放左子树  FreeBinaryTree(root->lChild);  //释放右子树  FreeBinaryTree(root->rChild);  free(root);  }  void testcopybtree()  {  //创建所有节点  BinaryNode node1 = { 'A', NULL, NULL };  BinaryNode node2 = { 'B', NULL, NULL };  BinaryNode node3 = { 'C', NULL, NULL };  BinaryNode node4 = { 'D', NULL, NULL };  BinaryNode node5 = { 'E', NULL, NULL };  BinaryNode node6 = { 'F', NULL, NULL };  BinaryNode node7 = { 'G', NULL, NULL };  BinaryNode node8 = { 'H', NULL, NULL };  //开始创建二叉树  //A节点是跟节点，它的左子节点是B,右子节点是F  node1.lChild = &node2;  node1.rChild = &node6;  //B节点，没有左子节点，右子节点是C（node3）  node2.rChild = &node3;  //C节点，它的左子节点是D（node4），右子节点是E（node5）  node3.lChild = &node4;  node3.rChild = &node5;  //F节点，它没有子左节点，它的右子节点是G节点（node7）  node6.rChild = &node7;  //G节点，它只有左子节点是H节点(node8)  node7.lChild = &node8;  Recursion(&node1);  printf("\n");  //拷贝二叉树  BinaryNode\* root = CopyBtree(&node1);  //输出复制结果  Recursion(root);  printf("\n");  FreeBinaryTree(root);  }  int main(int argc, char const \*argv[])  {  testcopybtree();  system("pause");  return 0;  } |

结果

|  |
| --- |
|  |