二叉树的高度定义为任何叶节点距根节点的最大深度。 也就是说，它是从根节点到任何叶节点的最长路径的长度。

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include<math.h>  /\*  求二叉树的高  \*/  //定义二叉树节点结构体  typedef struct BinaryNode  {  char ch;  struct BinaryNode\* lChild;  struct BinaryNode\* rChild;  }BinaryNode;  //定义一个求二叉树高度的函数  int tree\_height(BinaryNode\* root) {  if (root == NULL)  return 0;  else {  // 获取左子树的高度  int left\_height = tree\_height(root->lChild);  // 获取右子树的高度  int right\_height = tree\_height(root->rChild);  // 二叉树的高度=两个子树中最高的那一个的高度+1  return max(left\_height, right\_height) + 1;  }  } |

|  |  |
| --- | --- |
| //高度为3的二叉树  void TestGetBinaryTreeHeight()  {  //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 };    //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 = &node3;  //B节点，没有子节点  //C节点，它的左子节点是D（node4），右子节点是E（node5）  node2.lChild = &node4;  node2.rChild = &node5;    int height = 0;  //GetBinaryTreeHeight(&node1, &height);  height = tree\_height(&node1);  printf("the height is :%d\n", height);  } |  |
| //高度为4的二叉树  void TestGetBinaryTreeHeight2()  { //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;  int height = 0;  height = tree\_height(&node1);  //GetBinaryTreeHeight(&node1, &height);  printf("the height is :%d\n", height);  } |  |
| //高度为5的二叉树  void TestGetBinaryTreeHeight3()  {  //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 };  BinaryNode node9 = { 'I', NULL, NULL };  BinaryNode node10 = { 'J', 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;  //H节点增加两个字节点，I(node9)，J(node10)  node8.lChild = &node9;  node8.rChild = &node10;  int height = 0;  height = tree\_height(&node1);  //GetBinaryTreeHeight(&node1, &height);  printf("the height is :%d\n", height);  } |  |
| int main(int argc, char const \*argv[])  {  TestGetBinaryTreeHeight3();  system("pause");  return 0;  } | 结果：  TestGetBinaryTreeHeight()==》the height is：3  TestGetBinaryTreeHeight2()==》the height is：4  TestGetBinaryTreeHeight3()==》the height is：5 |