**1 创建二叉搜索树**

void InsertBinarySearchTree(TreeNode \*root, int key)

{

TreeNode \*p = root;

while (root)

{

p = root;

key < root->value ? root = root->lchild : root = root->rchild;

}

key < p->value ? p->lchild = NewTreeNode(key) : p->rchild = NewTreeNode(key);

}

TreeNode \*CreateBinarySearchTree(int \*num, int n)

{

if (!num || n <= 0) return NULL;

TreeNode \*root = NewTreeNode(num[0]);

for (int i = 1; i < n; i++)

{

InsertBinarySearchTree(root, num[i]);

}

return root;

}

**2 先序创建二叉树**

**3 中序创建二叉树**

**4 后序创建二叉树**

**5 层次创建二叉树**

TreeNode\* LevelCreateBTree(int \*num, int n)

{

if (!num || n <= 0 || num[0] == -1) return NULL;

TreeNode \*root = NewTreeNode(num[0]);

queue<TreeNode \*> queue;

queue.push(root);

int start = 0;

while (!queue.empty())

{

TreeNode \*p = queue.front();

queue.pop();

if (++start < n)

{

if (num[start] == -1) p->lchild = NULL;

else

{

p->lchild = NewTreeNode(num[start]);

queue.push(p->lchild);

}

}

if (++start < n)

{

if (num[start] == -1) p->rchild = NULL;

else

{

p->rchild = NewTreeNode(num[start]);

queue.push(p->rchild);

}

}

}

return root;

}

**6 先序+中序创建二叉树**