# 实验-3-树-参考程序

王 新 宇
计算机科学与技术系

```
//实验题第1题第（1）小题
//BinaryTree.h
template <class DataType>
void BinaryTree<DataType>::CreateBinaryTree(BTNode<DataType> * & r) {
    DataType ch;
    cin >> ch;
    if (ch == '#')    r = NULL;
    else {
        r = new BTNode<DataType>(ch);
        CreateBinaryTree(r->lChild);
        CreateBinaryTree(r->rChild);
    }
}
template <class DataType>
void BinaryTree<DataType>::CreateBinaryTree() {
    CreateBinaryTree(root);
}
```

```
//实验题第1题第（2）小题
//BinaryTree.h
template <class DataType>
int  BinaryTree<DataType>::Width()  {
        if (!root)    return 0;
        int treeheight = Height();
        int levelwidth, maxwidth = Width(1);
        for (int i = 2; i <= treeheight; i++)
        {
                levelwidth = Width(i);
                if (levelwidth > maxwidth)    maxwidth = levelwidth;
        }
        return maxwidth;
}
```

```
template <class DataType>
int  BinaryTree<DataType>::Width(int level)  {
      LinkQueue<BTNode<DataType> *> nodequeue;
      BTNode<DataType> *p;
      int curlevel, count;
      if (!root)    return 0;
      nodequeue.EnQueue(root);
      curlevel = 0;
      while (curlevel < level)  {
            curlevel++;
            count = nodequeue.GetLength();
            for (int i = count; i; i--)  {
                  nodequeue.DelQueue(p);
                  if (p->lChild != NULL)    nodequeue.EnQueue(p->lChild);
                  if (p->rChild != NULL)    nodequeue.EnQueue(p->rChild);
            }
      }
      return count;
}
```

```
//实验题第1题第（3）小题
//BinaryTree.h
template <class DataType>
int  BinaryTree<DataType>::NodeCount(const BTNode<DataType> *r)  {
        if(!r)    return 0;
        else
                return NodeCount(r->lChild) + NodeCount(r->rChild) + 1;
}
template <class DataType>
int  BinaryTree<DataType>::NodeCount()  {
        return NodeCount(root);
}
```

```
//实验题第2题第（1）小题
//Tree.h
template <class DataType>
int Tree<DataType>::Height(const TreeNode<DataType> *r) const  {
        if(r == NULL)    return 0;
        else  {
                int lHeight, rHeight;
                lHeight = Height(r->firstChild);
                rHeight = Height(r->nextSibling);
                return lHeight+1 > rHeight ? lHeight+1 : rHeight;
        }
}
template <class DataType>
int  Tree<DataType>::Height() const  {
        return Height(root);
}
```

```
//实验题第2题第（2）小题
//Tree.h
template <class DataType>
int Tree<DataType>::Degree() const  {
        return Degree(root);
}
template <class DataType>
int Tree<DataType>::Degree(TreeNode<DataType> *r) const  {
        LinkQueue<TreeNode<DataType> *> queue;
        TreeNode<DataType> *t;
        int nChild, degree;
        if(!r)    return 0;
        queue.EnQueue(r);
        degree = 0;
        while(!queue.IsEmpty())  {
                queue.DelQueue(t);
                nChild = 0;
```

```
            if(t->firstChild)  {
                  queue.EnQueue(t->firstChild);
                  nChild++;
                  t = t->firstChild;
                  while(t->nextSibling)  {
                        queue.EnQueue(t->nextSibling);
                        nChild++;
                        t = t->nextSibling;
                  }
            }
            if(nChild > degree)    degree = nChild;
      }
      return degree;
}
```