华北电力大学

**实 验 报 告**

|

|

**实验名称** 树

**课程名称**  数据结构综合实践

|

|

专业班级： 信息2202 学生姓名：徐梓乔

学 号： 220221100327 成 绩：

|  |
| --- |
| 一、实验内容  对下图所示的二叉树进行取双亲结点、左兄弟结点、右兄弟结点、统计叶子结点个数、交换子树、前序遍历、 中序遍历、后序遍历、层次遍历等操作。  二、实验方法与步骤  实验思路：  该问题涉及到了树的方方面面的代码实现，只要构造树并将其一些功能实现即可。  代码如下：  构建二叉树:  private void frmMain\_Load(object sender, EventArgs e)  {  \_a = new BinTreeNode<string>("A");  \_b = new BinTreeNode<string>("B");  \_c = new BinTreeNode<string>("C");  \_d = new BinTreeNode<string>("D");  \_e = new BinTreeNode<string>("E");  \_f = new BinTreeNode<string>("F");  \_g = new BinTreeNode<string>("G");  \_h = new BinTreeNode<string>("H");  \_i = new BinTreeNode<string>("I");  \_tree = new BinTree<string>(\_a);  \_tree.Insert(\_a, \_b, \_c);  \_tree.Insert(\_b, \_d, null);  \_tree.Insert(\_c, \_e, \_f);  \_tree.Insert(\_e, null, \_g);  \_tree.Insert(\_f, \_h, \_i);  cboNode.Items.Clear();  cboNode.Items.Add("A");  cboNode.Items.Add("B");  cboNode.Items.Add("C");  cboNode.Items.Add("D");  cboNode.Items.Add("E");  cboNode.Items.Add("F");  cboNode.Items.Add("G");  cboNode.Items.Add("H");  cboNode.Items.Add("I");  cboNode.SelectedIndex = 0;  }  通过switch-cases进行结点选取:  BinTreeNode<string> result = null;  switch (str)  {  case "A":  result = \_a;  break;  case "B":  result = \_b;  break;  case "C":  result = \_c;  break;  case "D":  result = \_d;  break;  case "E":  result = \_e;  break;  case "F":  result = \_f;  break;  case "G":  result = \_g;  break;  case "H":  result = \_h;  break;  case "I":  result = \_i;  break;  }  return result;  双亲结点:  private void cmdParent\_Click(object sender, EventArgs e)  {  string str = cboNode.Text;  BinTreeNode<string> binTreeNode = \_tree.GetParent(GetBinTreeNode(str));  txtParent.Text = ((binTreeNode != null) ? binTreeNode.Data : "没有双亲结点");  }  左兄弟:  private void cmdLeftSibling\_Click(object sender, EventArgs e)  {  string str = cboNode.Text;  BinTreeNode<string> leftSibling = \_tree.GetLeftSibling(GetBinTreeNode(str));  txtLeftSibling.Text = ((leftSibling != null) ? leftSibling.Data : "没有左兄弟");  }  右兄弟:  private void cmdRightSibling\_Click(object sender, EventArgs e)  {  string str = cboNode.Text;  BinTreeNode<string> rightSibling = \_tree.GetRightSibling(GetBinTreeNode(str));  txtRightSibling.Text = ((rightSibling != null) ? rightSibling.Data : "没有右兄弟");  }  叶结点数:  private void cmdLeafCount\_Click(object sender, EventArgs e)  {  txtLeafCount.Text = \_tree.GetLeafCount().ToString();  }  交换子树:  private void cmdExchangeSubTree\_Click(object sender, EventArgs e)  {  if (!\_flag)  {  picTreeImage.Load(".\\Tree2.bmp");  picTreeImage.Image = Resources.Tree2;  \_flag = true;  }  else  {  picTreeImage.Load(".\\Tree1.bmp");  \_flag = false;  }  \_tree.Exchange();  txtPreOrder.Clear();  txtMidOrder.Clear();  txtPostOrder.Clear();  txtLevel.Clear();  txtParent.Clear();  txtLeftSibling.Clear();  txtRightSibling.Clear();  txtLeafCount.Clear();  }  前序遍历:  private void cmdPreOrder\_Click(object sender, EventArgs e)  {  txtPreOrder.Text = \_tree.PreOrderTraversal();  }  private void PreOrder(BinTreeNode<T> current)  {  if (current != null)  {  \_orderString = string.Concat(\_orderString, current.Data, " ");  PreOrder(current.LeftChild);  PreOrder(current.RightChild);  }  }  public string PreOrderTraversal()  {  \_orderString = string.Empty;  PreOrder(Root);  return \_orderString.Trim();  }  中序遍历:  private void cmdMidOrder\_Click(object sender, EventArgs e)  {  txtMidOrder.Text = \_tree.MidOrderTraversal();  }  private void MidOrder(BinTreeNode<T> current)  {  if (current != null)  {  MidOrder(current.LeftChild);  \_orderString = string.Concat(\_orderString, current.Data, " ");  MidOrder(current.RightChild);  }  }  public string MidOrderTraversal()  {  \_orderString = string.Empty;  MidOrder(Root);  return \_orderString.Trim();  }  后序遍历:  private void cmdPostOrder\_Click(object sender, EventArgs e)  {  txtPostOrder.Text = \_tree.PostOrderTraversal();  }  private void PostOrder(BinTreeNode<T> current)  {  if (current != null)  {  PostOrder(current.LeftChild);  PostOrder(current.RightChild);  \_orderString = string.Concat(\_orderString, current.Data, " ");  }  }  public string PostOrderTraversal()  {  \_orderString = string.Empty;  PostOrder(Root);  return \_orderString.Trim();  }  层次遍历:  private void cmdLevel\_Click(object sender, EventArgs e)  {  txtLevel.Text = \_tree.LevelTraversal();  }  public string LevelTraversal()  {  \_orderString = string.Empty;  if (Root != null)  {  LinkQueue<BinTreeNode<T>> val = (LinkQueue<BinTreeNode<T>>)(object)new LinkQueue<BinTreeNode<BinTreeNode<T>>>();  ((LinkQueue<BinTreeNode<BinTreeNode<T>>>)(object)val).EnQueue((BinTreeNode<BinTreeNode<T>>)(object)Root);  while (!((LinkQueue<BinTreeNode<BinTreeNode<T>>>)(object)val).IsEmpty())  {  BinTreeNode<T> queueFront = ((LinkQueue<BinTreeNode<BinTreeNode<T>>>)(object)val).get\_QueueFront();  ((LinkQueue<BinTreeNode<BinTreeNode<T>>>)(object)val).DeQueue();  \_orderString = string.Concat(\_orderString, queueFront.Data, " ");  if (queueFront.LeftChild != null)  {  ((LinkQueue<BinTreeNode<BinTreeNode<T>>>)(object)val).EnQueue((BinTreeNode<BinTreeNode<T>>)(object)queueFront.LeftChild);  }  if (queueFront.RightChild != null)  {  ((LinkQueue<BinTreeNode<BinTreeNode<T>>>)(object)val).EnQueue((BinTreeNode<BinTreeNode<T>>)(object)queueFront.RightChild);  }  }  }  三、实验结果    四、结论与总结  结论:成功建立二叉树并进行取双亲结点、左兄弟结点、右兄弟结点、统计叶子结点个数、交换子树、前序遍历、 中序遍历、后序遍历、层次遍历等操作.  总结:二叉树操作:获取或设置二叉树结点、向二叉树中插入节点、获取前序遍历序列、获取中序遍历序列、获取后序遍历序列、获取层次遍历序列、获取给定结点的双亲结点、获取给定结点左右兄弟的结点、删除以给定结点为根的子树、根据给定数据查找其在二叉树中的结点、获取叶子结点的个数、交换二叉树的左右子树. |
|  |