华北电力大学

**实 验 报 告**

|

|

**实验名称** 字符串

**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  利用我们介绍的串结构（SeqString:IString），输入一个串，可对这个串做取子串、插入子串、删除子串、模 式匹配、克隆串、连接串等操作。  二、实验方法与步骤  取子串:  int startIndex = (int)numUpDownStartSubString.Value;  int count = (int)numUpDownCountSubString.Value;  txtboxSubString.Text = @string.SubString(startIndex, count).ToString();  break;  插入子串:  int startIndex = (int)numericUpDownInsert.Value;  IString s = new SeqString(textBoxSub.Text);  textBoxInsert.Text = @string.Insert(startIndex, s).ToString();  break;  删除子串:  int startIndex = (int)numericUpDownDelete.Value;  int count = (int)numericUpDownCountDelete.Value;  textBoxRemove.Text = @string.Remove(startIndex, count).ToString();  break;  模式匹配:  IString string2 = new SeqString(textBoxPat.Text);  if (@string.Length == 0)  {  MessageBox.Show("主串为Empty，不能进行匹配！", "系统提示", MessageBoxButtons.OK, MessageBoxIcon.Hand);  txtboxMain.Focus();  }  else if (string2.Length == 0)  {  MessageBox.Show("子串为Empty，不能进行匹配！", "系统提示", MessageBoxButtons.OK, MessageBoxIcon.Hand);  textBoxPat.Focus();  }  else  {  textBoxPatResult.Text = @string.IndexOf(string2).ToString();  }  break;  串拷贝:  textBoxCopy.Text = @string.Clone().ToString();  break;  连接串:  SeqString seqString = new SeqString(textBoxLY.Text);  textBoxJoin.Text = ((SeqString)@string + seqString).ToString();  break;  三、实验结果    四、结论与总结  结论:可以成功实行取子串、插入子串、删除子串、串匹配、串拷贝、连接串.  总结:串的基本操作:   1. 获取串的长度 2. 获取或设置指定索引处的字符 3. 在指定位置插入子串 4. 在指定位置移除给定的长度的子串 5. 在指定的位置取子串 6. 串的拷贝 7. 串连接 8. 串的匹配 9. 填充字符串 |
| 另：使用KMP算法在主串中查找子串  using System;  class KMPAlgorithm  {  static void Main()  {  string mainString = "ABABDABACDABABCABAB";  string pattern = "ABABCABAB";  int index = KMPSearch(mainString, pattern);  if (index != -1)  Console.WriteLine($"在主串中找到子串，起始位置为 {index}。");  else  Console.WriteLine("在主串中未找到子串。");  }  static int KMPSearch(string mainString, string pattern)  {  int[] lps = ComputeLPSArray(pattern);  int m = pattern.Length;  int n = mainString.Length;  int i = 0; // 索引用于主串  int j = 0; // 索引用于子串  while (i < n)  {  if (pattern[j] == mainString[i])  {  j++;  i++;  }  if (j == m)  {  // 找到子串在主串中的位置  return i - j;  }  else if (i < n && pattern[j] != mainString[i])  {  if (j != 0)  j = lps[j - 1];  else  i++;  }  }  return -1;  }  static int[] ComputeLPSArray(string pattern)  {  int m = pattern.Length;  int[] lps = new int[m];  int len = 0;  int i = 1;  while (i < m)  {  if (pattern[i] == pattern[len])  {  len++;  lps[i] = len;  i++;  }  else  {  if (len != 0)  len = lps[len - 1];  else  {  lps[i] = 0;  i++;  }  }  }  return lps;  }  } |