编

程

知

识

日

常

积

累

**龚 涛**

2015/11/20

目录

[前言 3](#_Toc435804454)

[第一章 常用计算函数 4](#_Toc435804455)

[1.1 求绝对值 4](#_Toc435804456)

[1.2 屋面活荷载 4](#_Toc435804457)

[1.3 雪荷载 4](#_Toc435804458)

[1.4 风荷载 5](#_Toc435804459)

[1.5 地震荷载 8](#_Toc435804460)

[1.6 作用效应组合 8](#_Toc435804461)

# 前言

本手册《幕墙易结构计算与设计原理》说明了幕墙易的主要编制原理和计算依据，展示了软件各模块计算中所采用的理论、假设及公式，力求让用户在使用软件的过程中能够比较清楚的了解其内部工作过程。

# 常用计算函数

本章主要记录常用并容易忘记的C#自带的数学函数。

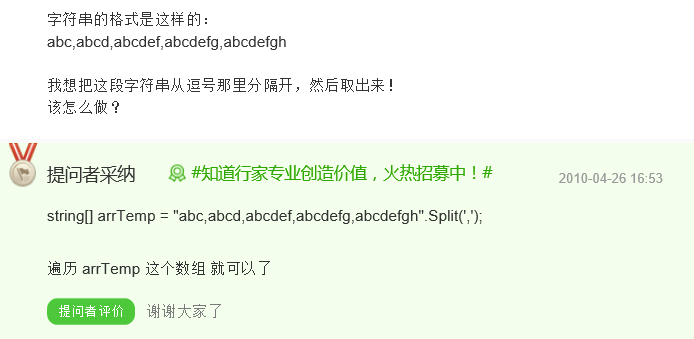
2011-05-30 15:06:35|  分类： [C#](http://blog.163.com/w_jingli/blog/#m=0&t=1&c=fks_084065092094081074087094087095087094083070081085080074) |  标签：[c#](http://blog.163.com/w_jingli/blog/#m=0&t=3&c=c#)  |字号 订阅

|  |  |
| --- | --- |
| 名称 | 说明 |
| [E](http://msdn2.microsoft.com/zh-cn/library/system.math.e(VS.80).aspx) | 表示自然对数的底，它由常数 e 指定。 |
| [PI](http://msdn2.microsoft.com/zh-cn/library/system.math.pi(VS.80).aspx) | 表示圆的周长与其直径的比值，它通过常数 π 指定。 |
| 名称 | 说明 |
| [Abs](http://msdn2.microsoft.com/zh-cn/library/system.math.abs(VS.80).aspx) | 已重载。 返回指定数字的绝对值。 |
| [Acos](http://msdn2.microsoft.com/zh-cn/library/system.math.acos(VS.80).aspx) | 返回余弦值为指定数字的角度。 |
| [Asin](http://msdn2.microsoft.com/zh-cn/library/system.math.asin(VS.80).aspx) | 返回正弦值为指定数字的角度。 |
| [Atan](http://msdn2.microsoft.com/zh-cn/library/system.math.atan(VS.80).aspx) | 返回正切值为指定数字的角度。 |
| [Atan2](http://msdn2.microsoft.com/zh-cn/library/system.math.atan2(VS.80).aspx) | 返回正切值为两个指定数字的商的角度。 |
| [BigMul](http://msdn2.microsoft.com/zh-cn/library/system.math.bigmul(VS.80).aspx) | 生成两个 32 位数字的完整乘积。 |
| [Ceiling](http://msdn2.microsoft.com/zh-cn/library/system.math.ceiling(VS.80).aspx) | 已重载。 返回大于或等于指定数字的最小整数。 |
| [Cos](http://msdn2.microsoft.com/zh-cn/library/system.math.cos(VS.80).aspx) | 返回指定角度的余弦值。 |
| [Cosh](http://msdn2.microsoft.com/zh-cn/library/system.math.cosh(VS.80).aspx) | 返回指定角度的双曲余弦值。 |
| [DivRem](http://msdn2.microsoft.com/zh-cn/library/system.math.divrem(VS.80).aspx) | 已重载。 计算两个数字的商，并在输出参数中返回余数。 |
| [Equals](http://msdn2.microsoft.com/zh-cn/library/system.object.equals(VS.80).aspx) | 已重载。 确定两个 [Object](http://msdn2.microsoft.com/zh-cn/library/system.object(VS.80).aspx;) 实例是否相等。 （从 [Object](http://msdn2.microsoft.com/zh-cn/library/system.object(VS.80).aspx;) 继承。） |
| [Exp](http://msdn2.microsoft.com/zh-cn/library/system.math.exp(VS.80).aspx) | 返回 e 的指定次幂。 |
| [Floor](http://msdn2.microsoft.com/zh-cn/library/system.math.floor(VS.80).aspx) | 已重载。 返回小于或等于指定数字的最大整数。 |
| [GetHashCode](http://msdn2.microsoft.com/zh-cn/library/system.object.gethashcode(VS.80).aspx) | 用作特定类型的哈希函数。[GetHashCode](http://msdn2.microsoft.com/zh-cn/library/system.object.gethashcode(VS.80).aspx) 适合在哈希算法和数据结构（如哈希表）中使用。 （从 [Object](http://msdn2.microsoft.com/zh-cn/library/system.object(VS.80).aspx;) 继承。） |
| [GetType](http://msdn2.microsoft.com/zh-cn/library/system.object.gettype(VS.80).aspx) | 获取当前实例的 [Type](http://msdn2.microsoft.com/zh-cn/library/system.type(VS.80).aspx;)。 （从 [Object](http://msdn2.microsoft.com/zh-cn/library/system.object(VS.80).aspx;) 继承。） |
| [IEEERemainder](http://msdn2.microsoft.com/zh-cn/library/system.math.ieeeremainder(VS.80).aspx) | 返回一指定数字被另一指定数字相除的余数。 |
| [Log](http://msdn2.microsoft.com/zh-cn/library/system.math.log(VS.80).aspx) | 已重载。 返回指定数字的对数。 |
| [Log10](http://msdn2.microsoft.com/zh-cn/library/system.math.log10(VS.80).aspx) | 返回指定数字以 10 为底的对数。 |
| [Max](http://msdn2.microsoft.com/zh-cn/library/system.math.max(VS.80).aspx) | 已重载。 返回两个指定数字中较大的一个。 |
| [Min](http://msdn2.microsoft.com/zh-cn/library/system.math.min(VS.80).aspx) | 已重载。 返回两个数字中较小的一个。 |
| [Pow](http://msdn2.microsoft.com/zh-cn/library/system.math.pow(VS.80).aspx) | 返回指定数字的指定次幂。 |
| [ReferenceEquals](http://msdn2.microsoft.com/zh-cn/library/system.object.referenceequals(VS.80).aspx) | 确定指定的 Object 实例是否是相同的实例。 （从 [Object](http://msdn2.microsoft.com/zh-cn/library/system.object(VS.80).aspx;) 继承。） |
| [Round](http://msdn2.microsoft.com/zh-cn/library/system.math.round(VS.80).aspx) | 已重载。 将值舍入到最接近的整数或指定的小数位数。 |
| [Sign](http://msdn2.microsoft.com/zh-cn/library/system.math.sign(VS.80).aspx) | 已重载。 返回表示数字符号的值。 |
| [Sin](http://msdn2.microsoft.com/zh-cn/library/system.math.sin(VS.80).aspx) | 返回指定角度的正弦值。 |
| [Sinh](http://msdn2.microsoft.com/zh-cn/library/system.math.sinh(VS.80).aspx) | 返回指定角度的双曲正弦值。 |
| [Sqrt](http://msdn2.microsoft.com/zh-cn/library/system.math.sqrt(VS.80).aspx) | 返回指定数字的平方根。 |
| [Tan](http://msdn2.microsoft.com/zh-cn/library/system.math.tan(VS.80).aspx) | 返回指定角度的正切值。 |
| [Tanh](http://msdn2.microsoft.com/zh-cn/library/system.math.tanh(VS.80).aspx) | 返回指定角度的双曲正切值。 |
| [ToString](http://msdn2.microsoft.com/zh-cn/library/system.object.tostring(VS.80).aspx) | 返回表示当前 Object 的 [String](http://msdn2.microsoft.com/zh-cn/library/system.string(VS.80).aspx;)。 （从 [Object](http://msdn2.microsoft.com/zh-cn/library/system.object(VS.80).aspx;) 继承。） |
| [Truncate](http://msdn2.microsoft.com/zh-cn/library/system.math.truncate(VS.80).aspx) | 已重载。 计算一个数字的整数部分。 |

## 1.1 求绝对值

Math.Abs(value);

## 用逗号分隔的字符串，分别取出来的方法



## 重新设置数组的大小

Array.Resize(ref lianxulcs, temp.Length);//修正数组的大小

# 事件

## 2.1 委托

网上视频：<http://v.youku.com/v_show/id_XODkyMzE0NzY0.html?from=s1.8-1-1.2>

委托是个引用类型，将方法委托给委托对象，进行传递。

方法的返回值类型和参数类型与个数都必须与定义的委托相同，才能进行委托。

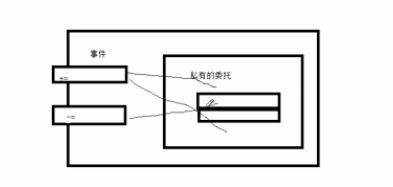
窗体之间参数的传递可以用委托来实现，

比如讲子窗体的参数传递到主窗口，可以在子窗体中声明委托和实例化委托对象，然后在主窗体中，将方法委托给委托。

## 2.2 事件

网上视频：<http://v.youku.com/v_show/id_XODk0OTQzMDQ4.html?from=s1.8-1-1.2>

<http://v.youku.com/v_show/id_XODk0NTc5Nzc2.html?from=s1.8-1-1.2>



事件可以将两个类中的方法绑定起来，一个发布者，一个订阅者，通过事件绑定。

为啥要使用事件？

使用步骤

1. 定义一个委托
2. 申明一个事件（发布者）（class）
3. 订阅者中创建回调方法(class)
4. 将发布者与订阅者绑定
5. 触发事件（必须在发布者类的内部触发事件，用方法来触发事件）

实例

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace event1

{

/// <summary>

/// MainWindow.xaml 的交互逻辑

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

BankCard icbc = new BankCard(){BankName="建设银行",Balance=10000};

Phone iphone = new Phone();

icbc.MyEvent += iphone.Display;

icbc.Pisplay(2000);//调用刷卡方法

}

}

internal delegate void MyDEL2(string BankName, int balance,int amount);//声明一个委托

class BankCard //发布者

{

public string BankName { get; set; }

public int Balance { get; set; }

public event MyDEL2 MyEvent;//声明一个事件

public void Pisplay(int amount)//刷卡方法触发事件

{

Balance -= amount;

if (MyEvent != null)//确认事件被订阅了

{

MyEvent(BankName,Balance,amount);//触发事件

}

}

}

class Phone //订阅者

{

public void Display(string name,int balance,int amount)//回调方法（事件处理程序）

{

MessageBox.Show("您的"+name+"卡消费了"+amount+",余额为："+balance);

}

}

}

## 2.3 在一个事件中调用另一个事件

void btn1\_Click(object sender,EventArgs e)   
{  
}  
void btn2\_Click(object sender,EventArgs e)   
{  
}  
  
如果在按钮1里触发按钮2的事件  
void btn1\_Click(object sender,EventArgs e)   
{  
btn2\_Click(sender,e) ;  
  
}就可以了

## 2.4让控件获取焦点事件

文本框的名称.Focus();

# 属性

## 4.1 将combobox控件的数据源绑定到数据库

# LinQ

## 4.1 简单的LINQ查询

private void Button\_Click(object sender, RoutedEventArgs e)

{

int[] nums = {1, 2, -3, 5, -7, 9, -12, 65, 15, 0, 62, 70};

//隐式类型,类型由编译器确定

//使用Linq查询

//1 创建查询

//2 执行查询

//所有的Linq查询都是以from 开始，以select 或者是group结束

//n为范围变量，nums为数据源

var posNums = from n in nums//创建查询

where n > 0 && n<20 //为查询条件，必须为bool值

select n;

//第一次查询

StringBuilder sbResult = new StringBuilder();

foreach (var item in posNums)//执行查询

{

sbResult.AppendFormat("{0},", item);

}

MessageBox.Show(sbResult.ToString());

//第二次查询

nums[2] = 3;

StringBuilder sbResult1 = new StringBuilder();

foreach (var item in posNums)//执行查询

{

sbResult1.AppendFormat("{0},", item);

}

MessageBox.Show(sbResult1.ToString());

}

//

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

string[] wwbadddr =

{

".com", "www.baidu.com", "www.google.com", "www.bing.com", "www.net",

"www.bb.org", "www.ccc.net"

};

var adds = from addr in wwbadddr

where addr.Length > 4 && addr.EndsWith(".com")

select addr;

StringBuilder sbResult = new StringBuilder();

foreach (var item in adds)

{

sbResult.AppendFormat("{0},", item);

}

MessageBox.Show(sbResult.ToString());

}

//

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

int[] nums = { 1, 2, -3, 5, -7, 9, -12, 65, 15, 0, 62, 70 };

//隐式类型,类型由编译器确定

//使用Linq查询

//1 创建查询

//2 执行查询

//所有的Linq查询都是以from 开始，以select 或者是group结束

//n为范围变量，nums为数据源

var posNums = from n in nums//创建查询

where n > 0 && n < 20 //为查询条件，必须为bool值

orderby n descending//排序 -降序

select n;

//第一次查询

StringBuilder sbResult = new StringBuilder();

foreach (var item in posNums)//执行查询

{

sbResult.AppendFormat("{0},", item);

}

MessageBox.Show(sbResult.ToString());

}

private void Button\_Click\_4(object sender, RoutedEventArgs e)

{

//选择每组中网址数量等于3个的

string[] wwbadddr =

{

".com", "www.baidu.com", "www.google.com", "www.bing.com", "www.net",

"www.bb.org", "www.ccc.net"

};

var addrs = from addr in wwbadddr

where addr.LastIndexOf(".") != -1

group addr by addr.Substring(addr.LastIndexOf("."));

StringBuilder sbResult = new StringBuilder();

foreach (var sites in addrs)

{

sbResult.AppendFormat("Key:{0}", sites.Key);

sbResult.AppendLine();

foreach (var item in sites)

{

sbResult.AppendFormat("{0}",item);

}

sbResult.AppendLine();

}

MessageBox.Show(sbResult.ToString());

}

private void Button\_Click\_5(object sender, RoutedEventArgs e)

{

//使用select 或者 group 时，有时候想生成一个临时结果，查询的后续

//部分需要使用临时结果来生成最终结果，这就称为查询继续，或者

//选择每组中网址数量等于3个的

string[] wwbadddr =

{

".com", "www.baidu.com", "www.google.com", "www.bing.com", "www.net",

"www.bb.org", "www.ccc.net"

};

var addrs = from addr in wwbadddr

let indx = addr.LastIndexOf(".")//用let在中间定义一个变量

where indx != -1

group addr by addr.Substring(indx)

into ws

where ws.Count() >= 3

select ws;

StringBuilder sbResult = new StringBuilder();

foreach (var sites in addrs)

{

sbResult.AppendFormat("Key:{0}", sites.Key);

sbResult.AppendLine();

foreach (var item in sites)

{

sbResult.AppendFormat("{0},", item);

}

sbResult.AppendLine();

}

MessageBox.Show(sbResult.ToString());

}

private void Button\_Click\_6(object sender, RoutedEventArgs e)

{

// join //表的连接等

Item[] items =

{

new Item {Numb = 1342, Name = "aa"},

new Item {Numb = 1343, Name = "bb"},

new Item {Numb = 1344, Name = "cc"},

new Item {Numb = 1345, Name = "dd"},

new Item {Numb = 1346, Name = "ee"}

};

Stored[] stores =

{

new Stored {ItemNumb = 1342, Instore = true},

new Stored {ItemNumb = 1343, Instore = true},

new Stored {ItemNumb = 1344, Instore = false},

new Stored {ItemNumb = 1345, Instore = true},

new Stored {ItemNumb = 1346, Instore = false}

};

var instorelist = from item in items

join entry in stores //两个表连接

on item.Numb equals entry.ItemNumb //连接条件

select new Temp(item.Name, entry.Instore);

StringBuilder sbResult = new StringBuilder();

foreach (var item in instorelist)

{

sbResult.AppendFormat("ItemName:{0} InStoreStatus:{1}", item.Name, item.Instore);

sbResult.AppendLine();

}

MessageBox.Show(sbResult.ToString());

}

private void Button\_Click\_7(object sender, RoutedEventArgs e)

{

Item[] items =

{

new Item {Numb = 1342, Name = "aa"},

new Item {Numb = 1343, Name = "bb"},

new Item {Numb = 1344, Name = "cc"},

new Item {Numb = 1345, Name = "dd"},

new Item {Numb = 1346, Name = "ee"}

};

Stored[] stores =

{

new Stored {ItemNumb = 1342, Instore = true},

new Stored {ItemNumb = 1343, Instore = true},

new Stored {ItemNumb = 1344, Instore = false},

new Stored {ItemNumb = 1345, Instore = true},

new Stored {ItemNumb = 1346, Instore = false}

};

var instorelist = from item in items

join entry in stores //两个表连接

on item.Numb equals entry.ItemNumb //连接条件

select new {item.Name, entry.Instore};//匿名类型

StringBuilder sbResult = new StringBuilder();

foreach (var item in instorelist)

{

sbResult.AppendFormat("ItemName:{0} InStoreStatus:{1}", item.Name, item.Instore);

sbResult.AppendLine();

}

MessageBox.Show(sbResult.ToString());

}

//查询方法和拉姆达表达式

# 常用类

## 5.1 StringBuilder学习

C#   
String 对象是不可改变的。每次使用 System.String 类中的方法之一时，都要在内存中创建一个新的字符串对象，这就需要为该新对象分配新的空间。在需要对字符串执行重复修改的情况下，与创建新的 String 对象相关的系统开销可能会非常昂贵。如果要修改字符串而不创建新的对象，则可以使用 System.Text.StringBuilder 类。例如，当在一个循环中将许多字符串连接在一起时，使用 StringBuilder 类可以提升性能。

通过用一个重载的构造函数方法初始化变量，可以创建 StringBuilder 类的新实例，正如以下示例中所阐释的那样。

设置容量和长度   
虽然 StringBuilder 对象是动态对象，允许扩充它所封装的字符串中字符的数量，但是您可以为它可容纳的最大字符数指定一个值。此值称为该对象的容量，不应将它与当前 StringBuilder 对象容纳的字符串长度混淆在一起。例如，可以创建 StringBuilder 类的带有字符串“Hello”（长度为 5）的一个新实例，同时可以指定该对象的最大容量为 25。当修改 StringBuilder 时，在达到容量之前，它不会为其自己重新分配空间。当达到容量时，将自动分配新的空间且容量翻倍。可以使用重载的构造函数之一来指定 StringBuilder 类的容量。以下代码示例指定可以将 MyStringBuilder 对象扩充到最大 25 个空白。      
MyStringBuilder.Capacity = 25;   
  
EnsureCapacity 方法可用来检查当前 StringBuilder 的容量。如果容量大于传递的值，则不进行任何更改；但是，如果容量小于传递的值，则会更改当前的容量以使其与传递的值匹配。   
  
也可以查看或设置 Length 属性。如果将 Length 属性设置为大于 Capacity 属性的值，则自动将 Capacity 属性更改为与 Length 属性相同的值。如果将 Length 属性设置为小于当前 StringBuilder 对象内的字符串长度的值，则会缩短该字符串。   
  
修改 StringBuilder 字符串   
下表列出了可以用来修改 StringBuilder 的内容的方法。   
  
StringBuilder.Append 将信息追加到当前 StringBuilder 的结尾。   
StringBuilder.AppendFormat 用带格式文本替换字符串中传递的格式说明符。   
StringBuilder.Insert 将字符串或对象插入到当前 StringBuilder 对象的指定索引处。   
StringBuilder.Remove 从当前 StringBuilder 对象中移除指定数量的字符。   
StringBuilder.Replace 替换指定索引处的指定字符。   
  
Append   
Append 方法可用来将文本或对象的字符串表示形式添加到由当前 StringBuilder 对象表示的字符串的结尾处。以下示例将一个 StringBuilder 对象初始化为“Hello World”，然后将一些文本追加到该对象的结尾处。将根据需要自动分配空间。   
  
[C#]   
StringBuilder MyStringBuilder = new StringBuilder("Hello World!");   
MyStringBuilder.Append(" What a beautiful day.");   
Console.WriteLine(MyStringBuilder);   
  
  
AppendFormat   
AppendFormat 方法将文本添加到 StringBuilder 的结尾处，而且实现了 IFormattable 接口，因此可接受格式化部分中描述的标准格式字符串。可以使用此方法来自定义变量的格式并将这些值追加到 StringBuilder 的后面。以下示例使用 AppendFormat 方法将一个设置为货币值格式的整数值放置到 StringBuilder 的结尾。   
  
[C#]   
int MyInt = 25;   
StringBuilder MyStringBuilder = new StringBuilder("Your total is ");   
MyStringBuilder.AppendFormat("{0:C} ", MyInt);   
Console.WriteLine(MyStringBuilder);   
  
Insert   
Insert 方法将字符串或对象添加到当前 StringBuilder 中的指定位置。以下示例使用此方法将一个单词插入到 StringBuilder 的第六个位置。   
  
[C#]   
StringBuilder MyStringBuilder = new StringBuilder("Hello World!");   
MyStringBuilder.Insert(6,"Beautiful ");   
Console.WriteLine(MyStringBuilder);   
  
Remove   
可以使用 Remove 方法从当前 StringBuilder 中移除指定数量的字符，移除过程从指定的从零开始的索引处开始。以下示例使用 Remove 方法缩短 StringBuilder。   
  
[C#]   
StringBuilder MyStringBuilder = new StringBuilder("Hello World!");   
MyStringBuilder.Remove(5,7);   
Console.WriteLine(MyStringBuilder);   
  
Replace   
使用 Replace 方法，可以用另一个指定的字符来替换 StringBuilder 对象内的字符。以下示例使用 Replace 方法来搜索 StringBuilder 对象，查找所有的感叹号字符 (!)，并用问号字符 (?) 来替换它们。   
  
[C#]   
StringBuilder MyStringBuilder = new StringBuilder("Hello World!");   
MyStringBuilder.Replace('!', '?');   
Console.WriteLine(MyStringBuilder);