测量程序设计

C#–WPF

作者：朱学军

xazhuxj@qq.com

西安科技大学

测绘科学与技术学院

二零一八年三月

本文档仅适用于西安科技大学测绘专业本科三年级学生教学。

也可用于其他专业人员参考。

教学计划及课程目标

教学计划

教学课时

该课程标准学时为 64 学时。本学期为 60 学时，上课 15 次 30 学时，上机 15 次 30 学时。
必修课，考试。笔试（60%）+ 上机实验与作业（40%）

学习内容

学习面向对象的程序设计与编写；
学习基本的测量算法程序编写，学习软件界面的编写。

编程预言的选择

注意：该课程不是再次学习某种编程语言，而是运用某种编程语言进行测量数据处理及测量算法的编写。

可能大家只学习了 C 语言，C 语言是面向过程的功能强大的语言，执行效率高，但界面程序编写较复杂。现代编程语言更多的是面向对象的编程语言，高效而且兼容 C 语言的是 C++，但界面的编写仍较复杂。因此我们选择更加现代化的编程语言 C#，它语法优美，界面编写方式有 WinForm 与 WPF 方式，虽然执行效率没有 C++ 高，但在 Windows7、Windows8/8.1、Windows10 中都几乎预装了运行库.Net Frame，并且得到了 Microsoft 的大力推广，Microsoft 的许多软件都在使用 C# 开发，学习与开发成本相比与 C++ 显著减少，效率显著提升。

C# 的语法与 JAVA 的许多语法是极其相似的，现代软件工程的许多方法也可以用 C# 去实现。因此，在本课程中我们将学习 C# 编程语言，并且会学习如何运用 C# 语言进行测量程序设计及测量数据处理。

现代软件基本上都具有简洁易用的界面，我们还将学习 WPF 的界面编写技术，为我们的程序设计处美观简洁的界面，这些都包含在 C# 之中。

编程环境与工具

本课程的基本编程工具为 Visual Studio，版本理论上为 2010 及更新版，推荐大家用 Visual Studio 2015 或 2017 版（目前的最新版）。

参考教材

C# 语言及 WPF 界面编写参考马骏主编，人民邮电出版社出版，《C# 程序设计及应用教程》第 3 版。

测量算法参考本教程（一直更新中.....）。

目录

第一章　从 C 走向 C#

基本上我们都学习过 C 语言，有过一定的软件设计基础。C 语言是功能强大的高效程序设计语言，但在现代的软件开发中特别是 Window 界面程序中开发的效率太低了。Microsoft 推出的.Net Framwork 在新版 Windows 中越来越多的进行了预安装，C# 语言面向对象、语法优美、功能强大、开发效率高，应用也越来越广泛。因此我们将从 C 语言程序设计转向 C# 程序设计，学习更多的现代程序设计方法，满足现代测绘大数据的处理与算法编写需求。

1.1　Client/Server 程序设计模式

现代软件往往是多人协作开发的成果，单个人进行大型软件的开发是比较少的。在软件开发中需要遵循软件工程的组织原则，寻求代码的可复用性、可测试性与可阅读性。在学习编程时我们首先应该改掉以下三个不良编码习惯：

改变 C 语言中将代码写入 main 函数中的习惯；

改变 C# 语言中将代码写入 Main 函数中的习惯；

改变在 UI(WinForm 或 WPF 或其它的界面) 中直接写入逻辑算法代码的习惯。

以上三种形式的代码书写处我们可以称为 Client,我们编写的逻辑算法代码可以称为 Server。试想如果我们去商场买东西，我们就是 Client，提供需求；商场就是 Server，提供服务。如果我们要买一只圆珠笔，也许我们只需简单的告诉商场人员，然后付钱拿笔走人就行了。但如果商场要让我们自己去仓库里找笔、查价钱、再在商场登记簿上登记库存等等一些动作，你想想会是怎么样的一种结果？买只笔都要把我们累死了。

这说明了什么呢？提供服务功能的商场应该对有功能需求的客户简化和屏蔽各种复杂的中间环节。在软件开发时同样如此，我们的 main 或 Main 函数或 UI 处的代码应该简洁，基本上只是调用各个功能算法而已。

如果我们像以上这三种方式组织代码，将会带来一系列的问题，尤其在团队开发与多人协作时代码不能复用，不能进行 unittest(单元测试)、不能用 git 工具（著名的源代码管理工具）进行源代码的自动合并。软件系统稍微复杂一点，我们的开发就会面临失败的危险。

万丈高楼从地起，因此，在学习编程时首先我们要有 Client/Server 模式意识，要遵循界面与算法相分离的原则进行程序设计。

1.2　从面向过程走向面向对象的程序设计

良好的面向过程设计程序设计程序是可以很好的转向面向对象的程序设计的，我们将从一个简单的 C 程序开始设计结构较为良好的 C 代码，再将其用面向对象的 C# 进行实现。从中体会面向过程与面向对象的程序设计方法的不同。

1.2.1 面向过程的 C 代码示例

在测绘专业中我们经常与测量点打交道，因此我们定义一个点 Point（测点除了它的坐标 x,y 和高程 z 之外，还需至少有点名），另外我们定义一个简单的为大家所熟知的圆 Circle，来使用点 Point 定义它的圆心，并实现判断两圆是否相交，计算圆的面积和周长等功能。代码如下所示：

```
1  // Ch01Ex01.cpp : Defines the entry point for the console application.
2  //
3
4  #include "stdafx.h"
5
6  #define _USE_MATH_DEFINES
7  #include <math.h>
8  #include <string.h>
9  #define PI M_PI
10
11 typedef struct _point {
12     char name[11];
13     double x, y, z;
14 }Point;
15
16 typedef struct _circle {
17     Point center;
18     double r;
19     double area;
20     double length;
21 }Circle;
22
23 //计算圆的面积
24 void Area(Circle * c) {
25     c->area = PI * c->r * c->r;
26 }
27
28 //计算两点的距离
29 double Distance(const Point * p1, const Point * p2) {
30     double dx = p2->x - p1->x;
31     double dy = p2->y - p1->y;
32     return sqrt(dx*dx + dy * dy);
33 }
34
35 //判断两圆是否相交
36 bool IsIntersectWithCircle(const Circle * c1, const Circle * c2) {
37     double d = Distance(&c1->center, &c2->center);
38     return d <= (c1->r + c2->r);
39 }
```

1.2.2 对 C 代码的解释

代码中的 11 和 16 行定义结构体时使用了 typedef，这样在标准 C 中再定义 Point 类型或 Circle 类型时可以避免在其前面加关键词 struct。

第 24-26 行计算圆的面积，传入了圆的指针，这是 C 及 C++ 中传递自定义数据类型的常用的高效方式。由于计算的圆面积会保存在结构体 Circle 中的 area 中，所以函数不需要返回值，返回值定义为 void。

第 29-33 行为计算两点的距离，由于该函数不会修改 p1、p2 两个 Point 内的任何成员，故应该将这两个指针定义为 const 类型，防止函数内部无意或故意的修改。

同理第 36-39 行判断两圆是否相交，由于该函数不会修改 c1、c2 两个圆内的任何成员，也应将这两个指针定义为 const 类型。

第 37 行由于 Distance 函数的参数需要两个 Point 指针类型的参数，尽管 c1 与 c2 均为指针，但 c1->center 与 c2->center 不是指针，所以需要在其前用取地址符 & 。

第 38 行的关系运算符本身的运算结果就为 bool 型，故不需要进行 if、else 类型的判断。

相应的 main 函数测试代码如下：

```
1  int main()
2  {
3      Point pt1;
4      strcpy_s(pt1.name, 11, "pt1");
5      pt1.x = 100; pt1.y = 100; pt1.z = 425.324;
6
7      Point pt2;
8      strcpy_s(pt2.name, 11, "pt2");
9      pt2.x = 200; pt2.y = 200; pt2.z = 417.626;
10
11     Circle c1;
12     c1.center = pt1; c1.r = 80;
13
14     Circle c2;
15     c2.center = pt2; c2.r = 110;
16
17     Area(&c1) ; //计算圆c1的面积
18     printf("Circle1 的面积 = %lf \n", c1.area );
19
20     bool yes = IsIntersectWithCircle(&c1, &c2);
21     printf("Circle1 与 Circle2 是否相交 : %s\n", (yes ? "是" : "否") );
22
23     return 0;
24 }
```

程序的运行结果如下：

```
Circle1 的面积 = 20106.192983
Circle1 与 Circle2 是否相交 : 是
```

1.2.3 面向对象的 C# 代码

将以上代码相对应的 C 代码直接翻译为 C# 代码为：

```
1  using System;
2
3  namespace Ch01Ex02
4  {
5      class Point
6      {
7          public string name;
8          public double x;
9          public double y;
10         public double z;
11
12         public double Distance(Point p2)
13         {
14             double dx = x - p2.x;
```

```
            double dy = y - p2.y;
            return Math.Sqrt(dx * dx + dy * dy);
        }
    }

    class Circle
    {
        public Point center;
        public double r;
        public double area;
        public double length;

        public void Area()
        {
            area = Math.PI * r * r;
        }

        public bool IsIntersectWithCircle(Circle c2)
        {
            double d = this.center.Distance(c2.center);
            return d <= (r + c2.r);
        }
    }
}
```

相应 C# 的 Main 函数测试代码如下：

```
using System;

namespace Ch01Ex02
{
    class Program
    {
        static void Main(string[] args)
        {
            Point p1 = new Point();
            p1.name = "p1";
            p1.x = 100;
            p1.y = 100;
            p1.z = 425.324;


            Circle c1 = new Circle();
            c1.center = p1;
            c1.r = 80;

            Point p2 = new Point();
            p2.name = "p2";
            p2.x = 200;
            p2.y = 200;
            p2.z = 417.626;

            Circle c2 = new Circle();
            c2.center = p2;
            c2.r = 110;

            c1.Area(); //计算圆c1的面积
            Console.WriteLine("Circle1的面积={0}", c1.area );
```

```
32
33              c2.Area(); //计算圆c2的面积
34              Console.WriteLine("Circle2的面积={0}", c2.area);
35
36              bool yes = c1.IsIntersectWithCircle(c2);
37              Console.WriteLine("Circle1与Circle2是否相交:{0}",
38                  yes ? "是" : "否");
39          }
40      }
41  }
```

程序的运行结果如下:

```
Circle1的面积=20106.1929829747
Circle2的面积=38013.2711084365
Circle1与Circle2是否相交:是
```

1.2.4 优化的面向对象 C# 代码

上面的 C# 不符合面向对象的软件设计原则，且在 Main 函数中，如果由于疏漏忘记第 30 行或 33 行，圆 c1 或 c2 将不会有正确的面积。

因此，我们将其优化。首先对 Point 类进行封装，将其字段定义为 private，用属性方法将其向外简单的暴露接口，相应的 C# 代码为:

```
1           private string name;
2           public string Name
3           {
4               get { return name; }
5               set { name = value; }
6           }
7
8           private double x;
9           public double X
10          {
11              get { return x; }
12              set { x = value; }
13          }
14          private double y;
15          public double Y
16          {
17              get { return y; }
18              set { y = value; }
19          }
20          private double z;
21          public double Z
22          {
23              get { return z; }
24              set { z = value; }
25          }
26
27          public Point(string name, double x, double y, double z)
28          {
29              this.name = name;
30              this.x = x;
31              this.y = y;
32              this.z = z;
```

```
33          }
```

以上是用 C# 的属性对 private 字段的简单封装，也可以写成如下形式：

```
1           public string Name { get ; set ; }
2           public double X { get ; set ;}
3           public double Y { get ; set ; }
4           public double Z {get ; set ;}
```

为了简化 Main 函数对 Point 类的调用及对其各字段值的初始化，我们定义一个构造函数如下：

```
1           public Point(string name, double x, double y, double z)
2           {
3               this.name = name;
4               this.x = x;
5               this.y = y;
6               this.z = z;
7           }
```

同样道理，也应对 Circle 进行封装，将其各个字段定义为 private，并用属性方法将其向外简单的暴露接口，相应的 C# 代码为：

```
1           public Point Center { get; set; }
2
3           private double r;
4           public double R
5           {
6               get { return r; }
7               set
8               {
9                   if(value != r)
10                  {
11                      r = value;
12                      CalArea(); //r的值发生改变，重新计算圆的面积
13                  }
14              }
15          }
```

圆心 Center 只是简单的封装，圆的半径则不同。由圆的特性知，当圆的半径确定，其面积与周长也就确定了。为了计算的高效，封装时，当圆的半径值发生改变时，就需要重新计算圆的面积，确保面积的正确性。

由于圆的面积只与圆的半径有关，我们不需要在其他的情况对圆的面积进行修改，因此封装的 C# 代码为：

```
1           private double area;
2           public double Area
3           {
4               get { return area; }
5           }
```

相应 C# 的 Main 函数测试代码如下：

```
1   using System;
2
3   namespace Ch01Ex03
4   {
5       class Program
```

```
6     {
7         static void Main(string[] args)
8         {
9             Circle c1 = new Circle("pt1", 100, 100, 425.324, 80);
10            Circle c2 = new Circle("pt2", 200, 200, 417.626, 110);
11
12            Console.WriteLine("Circle1的面积={0}", c1.Area );
13            Console.WriteLine("Circle2的面积={0}", c2.Area);
14
15            bool yes = c1.IsIntersectWithCircle(c2);
16            Console.WriteLine("Circle1与Circle2是否相交:{0}",
17                yes ? "是" : "否" );
18        }
19    }
20 }
```

1.3 C# 与面向对象基础

C 语言中有 struct，在 C++ 中对 struct 进行了扩展，其实质就是一 class。C 与 C++ 中又特别是 C 中，struct 与指针是其根本。

类 (class) 是绝大多数的面向对象程序设计语言的关键词。

程序或软件的基本概念是：程序 = 数据结构 + 算法

从程序设计语言的语法角度分析，class 是数据与函数的复合体，是符合上述程序设计原则的。

1.4 C# 程序的组织结构

从以上 C# 示例代码可以看出，C# 是一纯面向对象语言，其组织程序的基本单位是类 (class)。在 class 之外是 namespace，在 class 内可以定义类成员 (class member) 和类方法 (class method)。在 method 中可以定义语句。

Visual Studio 用 Solution(.sln)、Project(.csproj) 与 File(.cs) 及文件夹进行组织。其形式如下：

Solution –1..m–Project – 1..m–File

一个 Project 对应一个 Exe 文件或 DLL 文件或其它模块，是 class 等的功能组合。

第二章　C# 语言基础

由于我们都学习过 C 语言，在此我们主要讲解 C# 中不同于 C 的一些基本语法。

2.1　C# 中的数据类型

C# 中的数据类型可分为两类，一类为值类型，一类为引用类型。

2.1.1　值类型

2.1.2　引用类型

2.1.3　装箱与拆箱

2.1.4　struct 与 class

struct 是值类型数据，class 是引用类型数据。

2.1.5　C# 中的 string

C# 中的 string 不是值类型数据，而是引用类型数据。

2.2　C# 中的数组

2.3　C# 中的条件语句

2.3.1　if 语句

2.3.2　switch 语句

2.4　C# 中的循环语句

2.4.1　for 语句

2.4.2　foreach 语句

2.4.3　while 与 do while 语句