从语句到方法

北京理工大学计算机学院
金旭亮

背景

在实际开发中，我们经常会发现某些功能在很多程序中都需要。当然，你可以直接地在不同程序中“Copy & Paste”代码，但这么干，麻烦很多：

当你发现了这些代码中有错误时，你必须找出它们被复制过的所有地方，一一更改，这实在太烦人了.....

能不能把这些需要重复使用的代码“归作一堆”，给它起个名字，然后在需要调用它们时，只需指定一个名字即可？

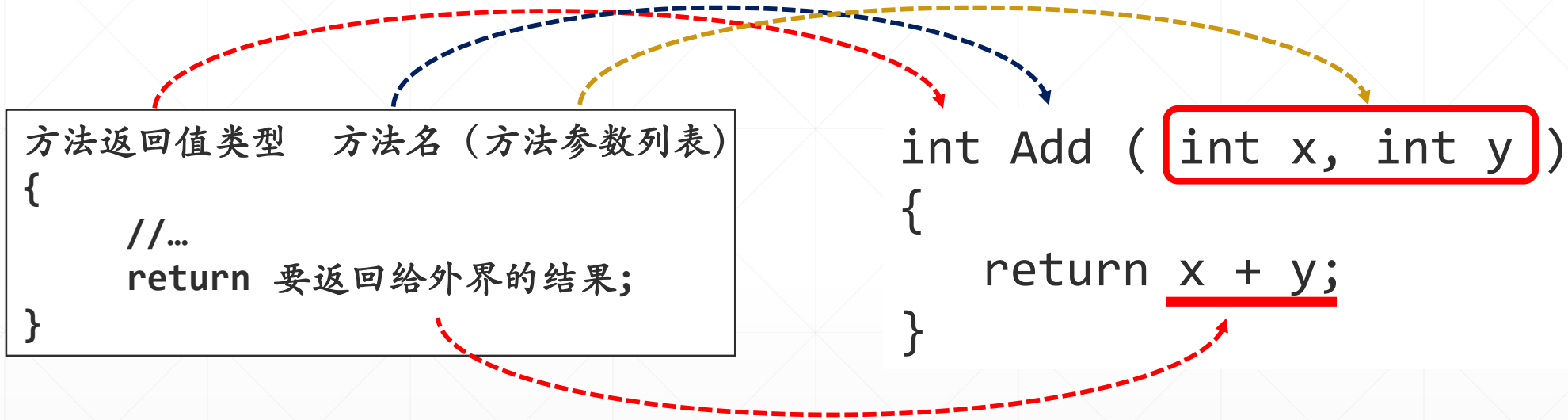
我们的解决方案

把多个语句组合在一起，共同完成一个功能，向外界返回一个结果，再给它起个名字，这样的一个“代码集合”，在面向对象编程领域，称之为“**方法（method）**”。

在结构化编程领域，面向对象中的“**方法（mothod）**”被称为“**函数（function）**”，这两个术语经常混用，可以看成是一回事（虽然有细微的差别）。

在C#中，所有方法都必须放到一个“**类（class）**”中，不存在完全独立的方法。

方法示例：设计一个计算两数之和的方法



定义方法时指明的参数，称为“形参（即形式参数）”。

“return” 关键字之后的表达式，代表要返回给外界的结果，称为“（函数或方法的）返回值”

Add方法调用实例

方法调用：因为Add方法返回一个整数，所以，它可以用在一个整数可以出现的地方。

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Console.WriteLine("{0}+{1}={2}",100,200,Add(100,200));
    }

    1 reference
    static int Add(int x, int y)
    {
        return x + y;
    }
}
```

方法的定义

调用Add()方法时传入的“100”和“200”，称为方法的“**实参（实际参数）**”。

静态方法

```
static int Add ( int x, int y)
{
    return x + y;
}
```

注意上面代码的static关键字，它表明这个方法是一个“静态方法（static method）”。

C#中，位于同一个类的静态方法可以通过方法名直接调用，其它类要调用时，需要加上此方法所在的类名，比如：Program.Add（100,200）；

如果定义方法时没有加上static关键字，它表明这个方法是一个“实例方法（instance method）”，这种方法依附于特定的对象，外界需要通过对象变量来调用。这部分内容，留待后面课程介绍。

试一试

仿照前一个示例，自己动手编写一个方法，它接收两个数值类型（比如int、long、float、double）的参数，向外界返回其中较大的一个。

方法的重载

在同一个类中，我们可以定义名字一样的方法，只要它们的参数列表不一样就行了，这种语法特性，叫作“**方法的重载（method overload）**”

三个重载的Add()方法

```
1 个引用
static int Add(int x, int y)
{
    return x + y;
}

1 个引用
static double Add(double x, double y)
{
    return x + y;
}

1 个引用
static double Add(string x, string y)
{
    double dx = double.Parse(x);
    double dy = double.Parse(y);
    return dx + dy;
}
```

什么叫“参数列表不一样”？

（1）参数个数不一样
（2）参数个数相同，但相同位置的参数，其类型不一样

注意：
返回值类型不作为方法重载判断的依据。

重载方法的调用

对于重载方法，到底调用的是哪个，是由其参数决定的。

```
var intResult = Add(100, 200);
var doubleResult = Add(100.5d, 200.5d);
var doubleResult2 = Add("100", "200");
```

调用

调用

调用

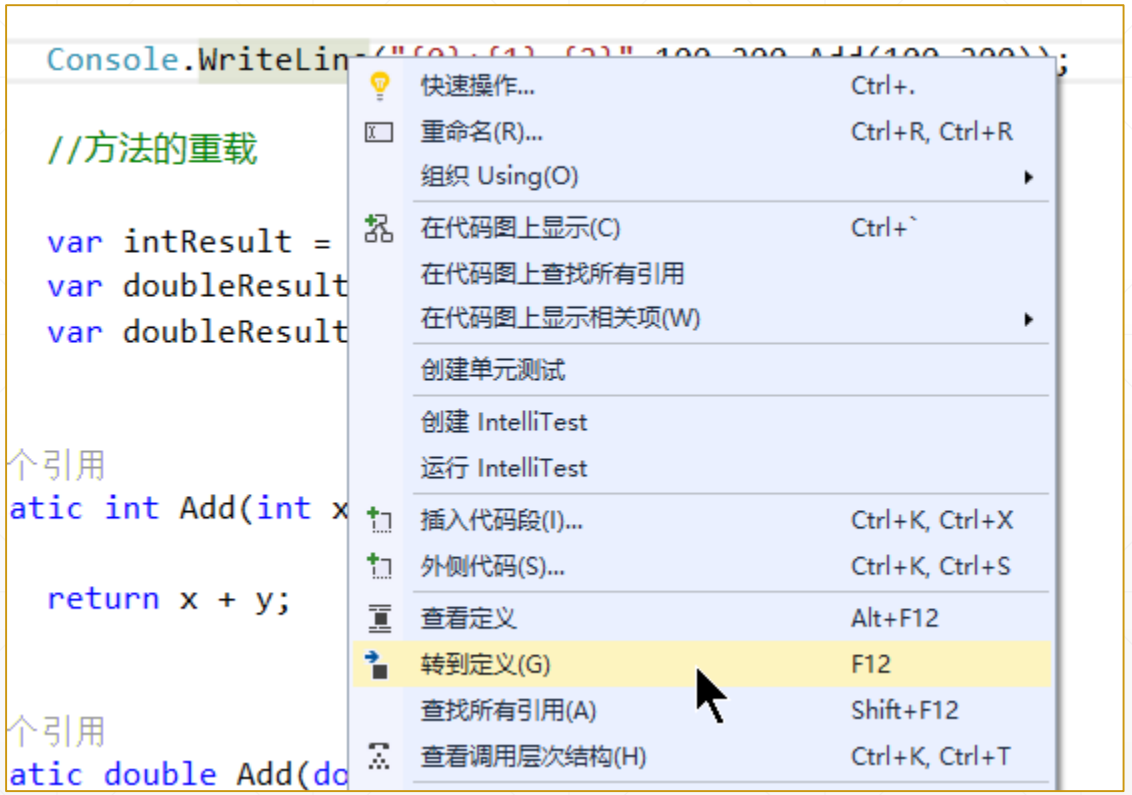
```
1 个引用
static int Add(int x, int y)
{
    return x + y;
}

1 个引用
static double Add(double x, double y)
{
    return x + y;
}

1 个引用
static double Add(string x, string y)
{
    double dx = double.Parse(x);
    double dy = double.Parse(y);
    return dx + dy;
}
```

动手动脑

在代码编辑器中选中Console.WriteLine(...)中的“WriteLine”这个单词，然后按F12键（或者是选中之后鼠标右击，从弹出菜单中选“转到定义”），你发现了什么？



下面来看几个复杂一点的方法示例.....

编程从模仿开始

一个可以显示图片的小程序：ShowPicForm

打开Visual Studio，跟着教师，一步步地完成示例，务必弄懂其每一行代码的作用

示例技术要点：

1. 读取文件并显示的功能被封装成一个方法，
2. 这里面用到了按钮、PictureBox和OpenFileDialog三个控件，请注意记住它们的使用方法和典型代码。

数学中用于生成随机数（pseudorandom)的公式：

后一个随机数　　前一个随机数

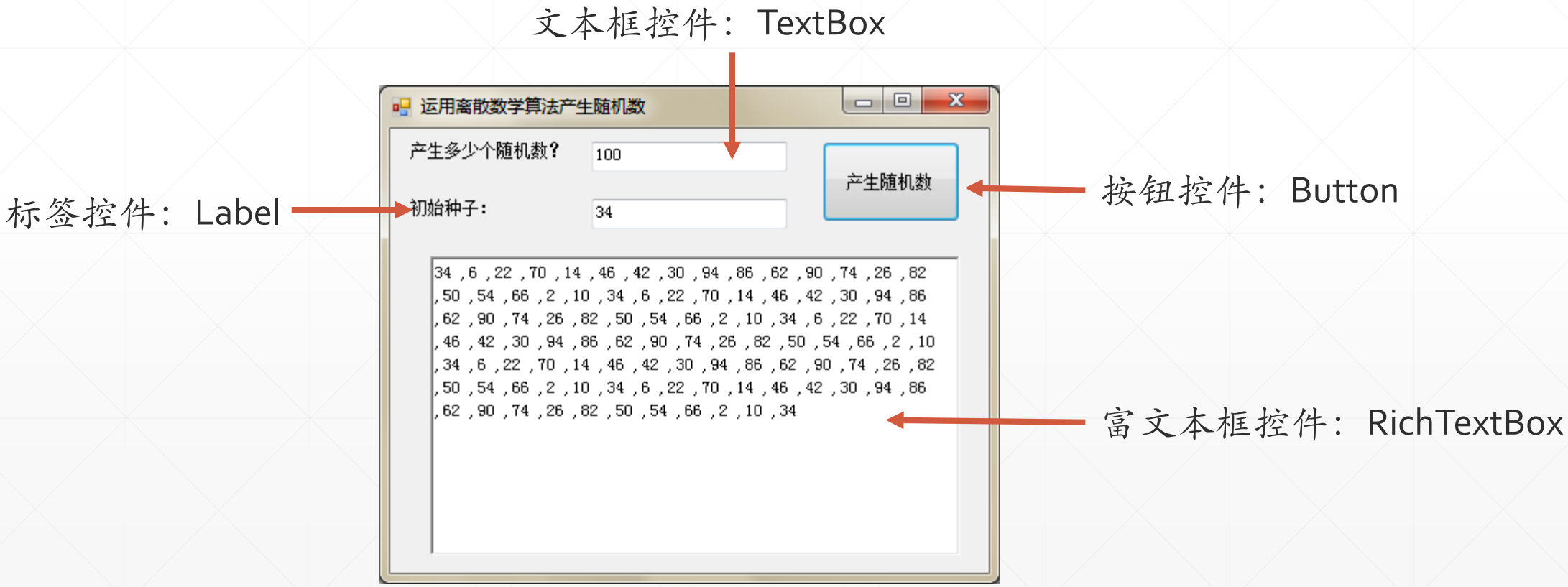
$$x_{n+1} = (ax_n + c) \bmod m$$

Multiplier（乘数）　Increment（增量）　Modulus（模）

Seed（种子，即第一个随机数）：x_0

代码阅读训练

自己阅读pseduorandom示例代码，此示例采用前页的数学公式生成随机数。你能看得懂吗?



卷起袖子动动手……

给前面PPT所展示的数学公式以具体的数值，得到一个随机数生成器算法，如下所示：

$$x_{n+1} = (ax_n + c) \bmod m$$

Multiplier=7^5=16807　　C=0　　Modulus= 2^{31}-1 = int.MaxValue

使用这一随机数生成算法，当显示过$2^{31}-2$个数之后，才可能重复。

你需要完成什么任务?

请编写一个示例程序，使用以上算法生成指定数目（比如1000个）的随机整数

偷懒的方法：使用.NET类库中的类生成随机数

GenerateRandomNumber示例中使用以下代码调用.NET基类库中的Random类生成随机数：

参看实例：GenerateRandomNumber

代表当前计算机内部时钟的“滴答”数

```
Random ran = new Random(System.Environment.TickCount);
for (int i = 0; i < 100; i++)
     Console.Write(" {0}", ran.Next(1, 100));
```

生成1~100之间的随机整数

动手重构示例：

重写前两页中所展示的示例pseduorandom，用Random类换掉手工编写生成随机数的那些代码。