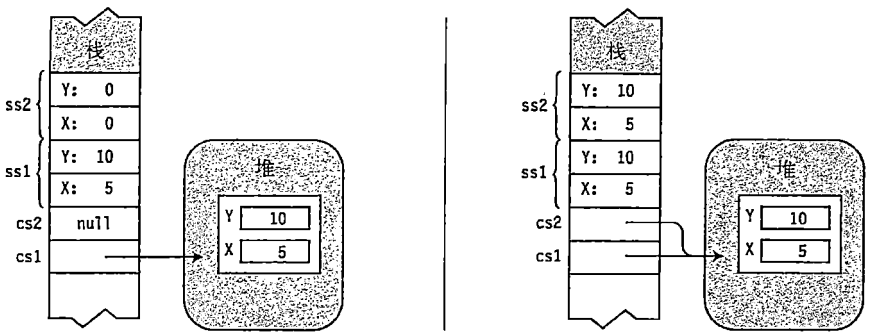

在实例赋值之前　　在实例赋值之后

图 11-2　类变量赋值和结构变量赋值

11.4　构造函数和析构函数

结构可以有实例构造函数和静态构造函数，但不允许有析构函数。

11.4.1　实例构造函数

语言隐式地为每个结构提供一个无参数的构造函数。这个构造函数把结构的每个成员设置为该类型的默认值。值成员设置成它们的默认值，引用成员设置成 null。

对于每个结构，都存在预定义的无参数构造函数，而且不能删除或重定义。但是，可以创建另外的构造函数，只要它们有参数。注意，这和类不同。对于类，编译器只在没有声明其他构造函数时提供隐式的无参数构造函数。

调用一个构造函数，包括隐式无参数构造函数，要使用 new 运算符。注意，即使不从堆中分配内存，也要使用 new 运算符。

例如，下面的代码声明了一个简单的结构，它有一个带两个 int 参数的构造函数。Main 创建该结构的两个实例，一个使用隐式无参数构造函数，另一个使用带两个参数的构造函数。

```
struct Simple
{
   public int X;
   public int Y;

   public Simple(int a, int b)            //带有参数的构造函数
   {
      X = a;
      Y = b;
   }
}

class Program
```

```
{
   static void Main()
   {                调用隐式构造函数
                        ↓
      Simple s1 = new Simple();
      Simple s2 = new Simple(5, 10);
                          ↑
                      调用构造函数
      Console.WriteLine($"{ s1.X },{ s1.Y }");
      Console.WriteLine($"{ s2.X },{ s2.Y }");
   }
}
```

也可以不使用 new 运算符创建结构的实例。然而，如果这样做，有一些限制，如下：

- 在显式设置数据成员之后，才能使用它们的值；
- 在对所有数据成员赋值之后，才能调用结构的函数成员。

例如，下面的代码展示了结构 Simple 的两个实例，它们没有使用 new 运算符创建。当企图访问 s1 而没有显式地设置该数据成员的值时，编译器产生一条错误消息。对 s2 的成员赋值之后，读取 s2 就没有问题了。

```
struct Simple
{
   public int X;
   public int Y;
}

class Program
{
   static void Main()
   {
        没有构造函数的调用
          ↓   ↓
      Simple s1, s2;
      Console.WriteLine("{0},{1}", s1.X, s1.Y);              //编译错误
                                    ↑     ↑
      s2.X = 5;                   还未被赋值
      s2.Y = 10;
      Console.WriteLine($"{ s2.X },{ s2.Y }");              //没问题
   }
}
```

11.4.2　静态构造函数

与类相似，结构的静态构造函数创建并初始化静态数据成员，而且不能引用实例成员。结构的静态构造函数遵从与类的静态构造函数一样的规则，但允许有不带参数的静态构造函数。

以下两种行为，任意一种发生之前，将会调用静态构造函数。

- 调用显式声明的构造函数。
- 引用结构的静态成员。