变量、数据类型与表达式

北京理工大学计算机学院
金旭亮

1 理解“变量”

什么是“变量”？

程序运行所需要处理的数据，通常都需要放到“变量（variable）”中。

变量可以看作是一种数据容器。

不同类型的“容器”，适合放置不同类型的数据，这种“类型”，我们称它为“变量的数据类型”。

如何在C#中定义变量？

语法格式：

数据类型名称 变量名 = 变量初始值;

示例：

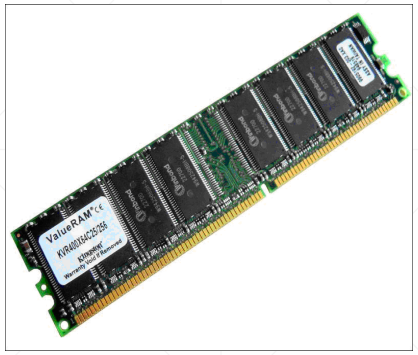
```
int value = 100;
```

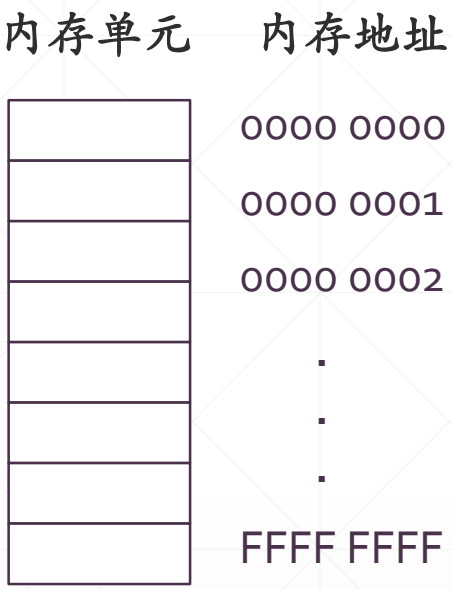
C#语句以“分号”结束，因此，这个分号不能省，省略了之后，代码将无法编译。

上述语句定义了一变量，它的类型是“int（整型）”，可以放置一个整数，它最初“装入”的数是“100”。

变量“生活”在“内存（memory）”中……

聊聊内存、内存单元和内存地址





应用程序中能使用的内存，是由操作系统提供的“虚拟内存”，其物理载体通常是我们在前一单元视频中介绍过的内存条和硬盘。

- 内存由多个内存单元构成，每个内存单元都有一个编号，称为“内存地址”。
- 给定一个“内存地址”，就能找到特定的内存单元。
- 计算机可以从内存单元中写入或读出数据。

使用比较底层的编程语言，比如“**汇编语言（Assembly Language）**”，我们可以直接指定地址去存取特定的内存单元，但这么干“太低效了”，为什么？

不同的计算机硬件，可以访问的物理内存数量是变化的

不同的操作系统，甚至是同一操作系统的不同版本，对内存单元的存取可能都有着不同的要求

你必须针对特定的计算机去写“特定”的程序，硬件略有变化，如果不作修改，你的程序可能就崩溃了……

计算机科学家给出的解决方案是……

1. 应用程序需要存取数据时，不指定具体的内存地址值，而只是给出一个“名字”，这个“名字”引用某块内存区域。

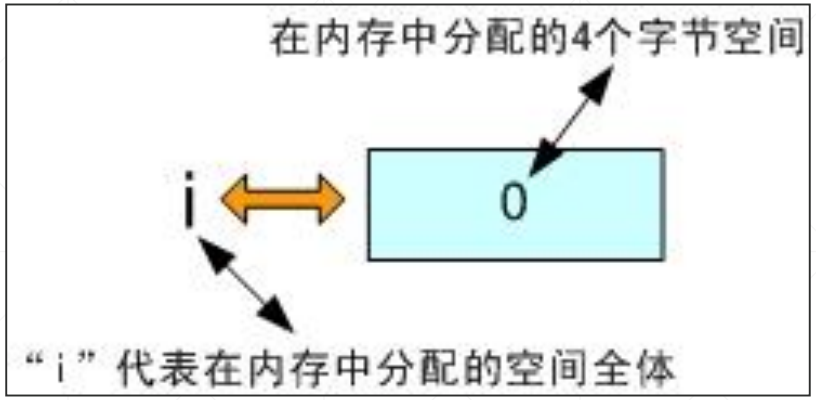
2. 这个“名字”具体到底引用的是哪块内存区域，由操作系统（Windows）负责安排，应用程序就不要费这个心了！

3. 应用程序根本就不管它要用的数据具体保存在哪个地方，它只知道，我可以“按名”存取数据就够了！

这个内存区域的“名字”，就是我们所讲的“变量”！

通过变量名“间接”地存取内存

```
int i = 0 ;
```



* 变量的名字（上面代码中的“i”），可以看成是特定内存区域的“别名”，通过它计算机就能找到特定的内存单元存取数据。
* 有了变量名，就不再需要显式指定一长串的地址数值来访问内存单元了。

变量名  某块内存区域的“别名”

“给变量赋值”是什么意思？

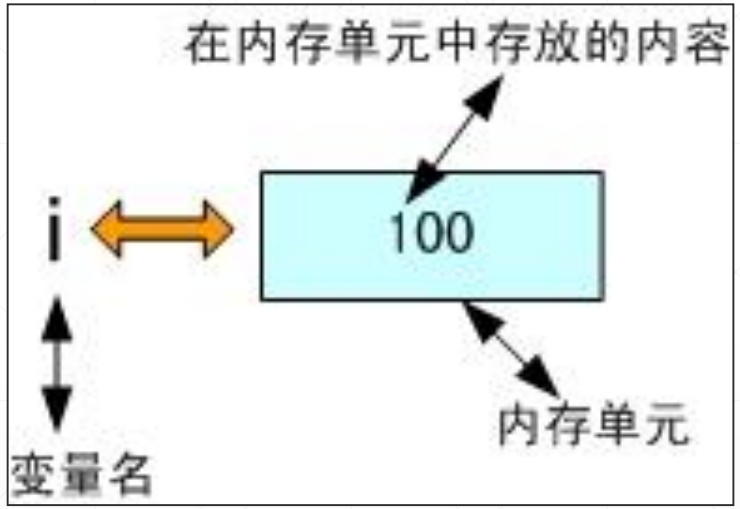
变量定义好之后，可以随时地通过**赋值语句**传给它不同的值……

语法格式：**变量名 = 变量新值;**

实例：

```
i = 100;
```

C#代码中的单个等号，称为“**赋值符**”，与数学中的等号是两回事，它表示要把等号右边的值，传给左边的变量。

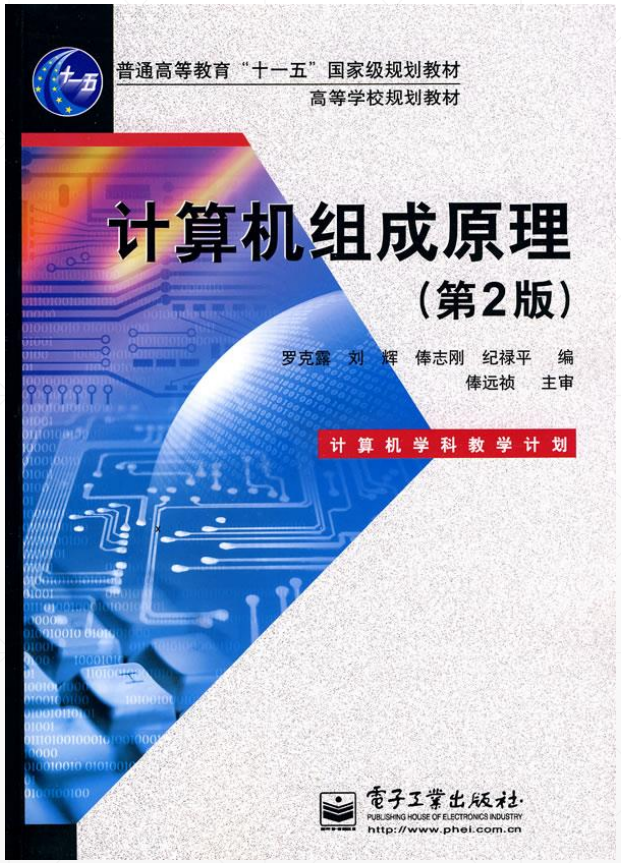


给变量**赋值**，其实就是找到变量所代表的内存区域，把指定的数值写入其中。

注意：写入到内存单元中的数据，都被转换为**二进制**数值，计算机并不能直接处理我们常用的十进制数。

自学指导

《计算机组成原理》这门计算机专业课的教材中，通常都会有专门的章节详细介绍人们常用的十进制数怎样转换为二进制数，并且基于二进制运算法则，如何使用晶体管和集成电路芯片完成“加、减、乘、除”等数值计算的。



《编码：隐匿在计算机软硬件背后的语言》

Charles Petzold 著

通俗易懂地介绍“0”和“1”在计算机设计中所起的巨大作用！

变量之间的相互赋值，又该怎么理解？

```
int i  = 100 ;
int j;
j = i;
```

* 变量间的相互赋值，本质上是内存单元间的值复制。
* 在上述代码中，j＝i，其实就是把“i”中所保存的数值复制一份，然后保存到“j”所代表的内存单元中。
* 注意：变量“i”和“j”之间是完全独立的，修改“i”的值，不会导致“j”的值同步变化。

2 C#中的数据类型

数据类型可用于定义变量

```
int intValue = 100;
long longValue = 100l;
double doubleValue = 100.5d;
float floatValue = 100.5f;
```

C#中的常见数据类型

数据类型名字	说明	数值类型后缀字符
int	整型数，用32位二进制数表达	
long	长整型数，用64位二进制表达	l
float	浮点数，用32位二进制数表达	f
double	双精度数，用64位二进制数表达	d
string	用于引用一个字符串对角	

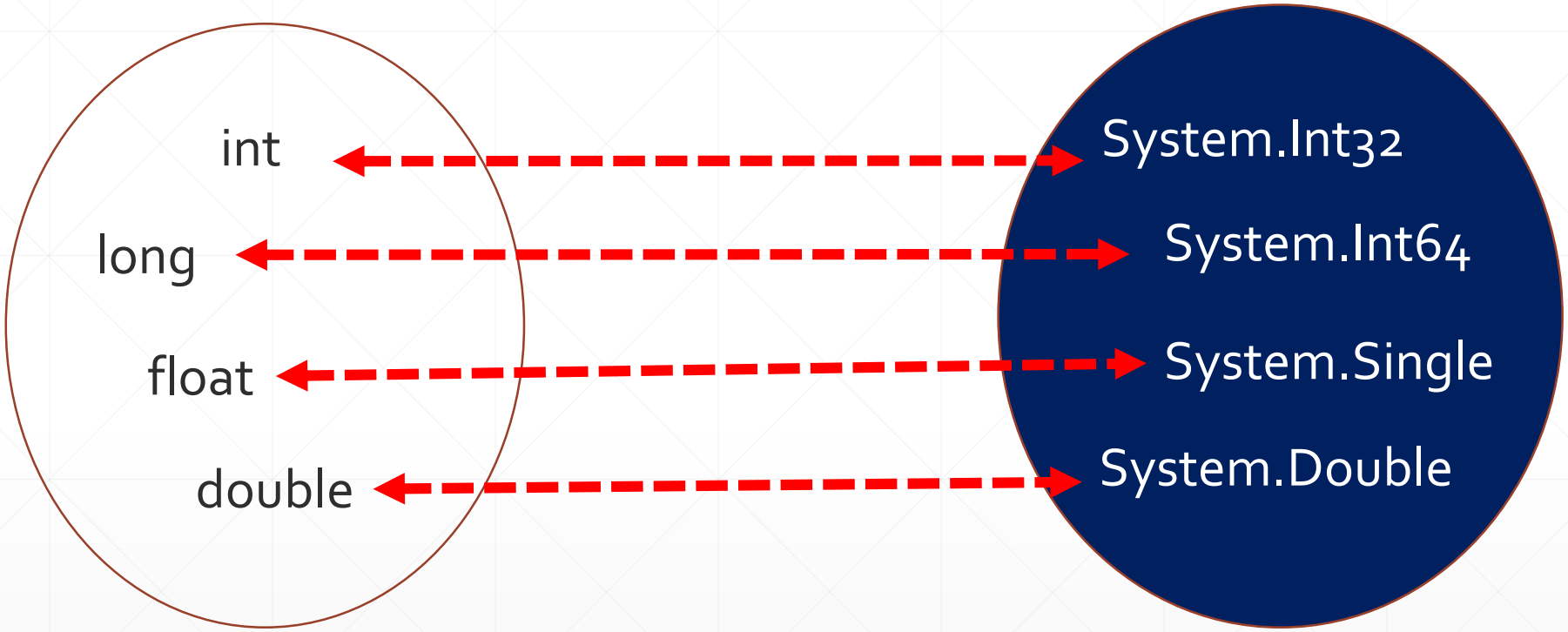
上述数据类型在编程中经常用到，要熟记其基本含义。

了解特定变量的类型

使用“**变量.GetType()**”方法可以获取变量的类型

使用“**typeof**”运算符关键字可以检测特定的变量是否是特定的类型

```
int intValue = 100;
Console.WriteLine(intValue.GetType());  //System.Int32
Console.WriteLine(intValue.GetType()==typeof(int));  //true
```



C#语言内置数据类型　　CLR支持的基础数据类型

String 还是 string

```
static void StringOrstring()
{
    String str1 = "Hello";
    string str2 = "World";
    Console.WriteLine(str1.GetType());
    Console.WriteLine(str2.GetType());
    Console.WriteLine(typeof(String)==typeof(string));
}
```

```
System.String
System.String
True
```

在C#中，定义字符串可以混用“String”或“string”，因为在编译时，C#会将它们都对应到.NET基类库所定义的标准“System.String”类型。

“没有类型”的变量？

C#定义了一个“**var**”关键字可以用于定义变量：

```
var value = 100;
```

```
int value= 100;
```

等价于

当使用var关键字定义变量时，C#编译器会依据右边给它赋的具体值，**推断**出变量的类型，所以，在编程时我们就可以“偷懒”不写变量类型，少打几个字符。

```
var dic = new Dictionary<string, List<int>>();
```

等价于

```
Dictionary<string, List<int>> dic = new Dictionary<string, List<int>>();
```

变量所占的内存空间

可以使用sizeof运算符关键字，了解特定类型变量所占内存空间的大小

```
static void PrintDataTypeLength()
{
    Console.WriteLine("int类型占用：{0}字节", sizeof(int));
    Console.WriteLine("long类型占用：{0}字节", sizeof(long));
    Console.WriteLine("float类型占用：{0}字节", sizeof(float));
    Console.WriteLine("double类型占用：{0}字节", sizeof(double));
}
```

```
int类型占用：4字节
long类型占用：8字节
float类型占用：4字节
double类型占用：8字节
```

为什么我们要了解数据类型所占用的空间大小？

```
int类型占用：4字节
long类型占用：8字节
float类型占用：4字节
double类型占用：8字节
```

当两个不同类型的数值变量相互赋值时，占用空间小的可以直接赋给占用空间大的，但反过来就不行。

占用空间大的要赋给占用空间小的，必须使用类型转换。

变量类型转换的语法：

变量名1 = (数据类型名) 变量名2;

```
//“小的”可以直接赋给“大的”
longValue = intValue;
doubleValue = intValue;
doubleValue = floatValue;

//“大的”要赋值给“小的”，必须进行“类型转换”
intValue = (int)longValue;
intValue = (int)doubleValue;
//即使占用相同大小的内存，因为类型不同，也必须进行“类型转换”
intValue =(int)floatValue;
```

参看测试方法代码：NumberTypeConverter()

字符串与数值类型间的转换

```
string strValue = "100";
```

string -->数值类型

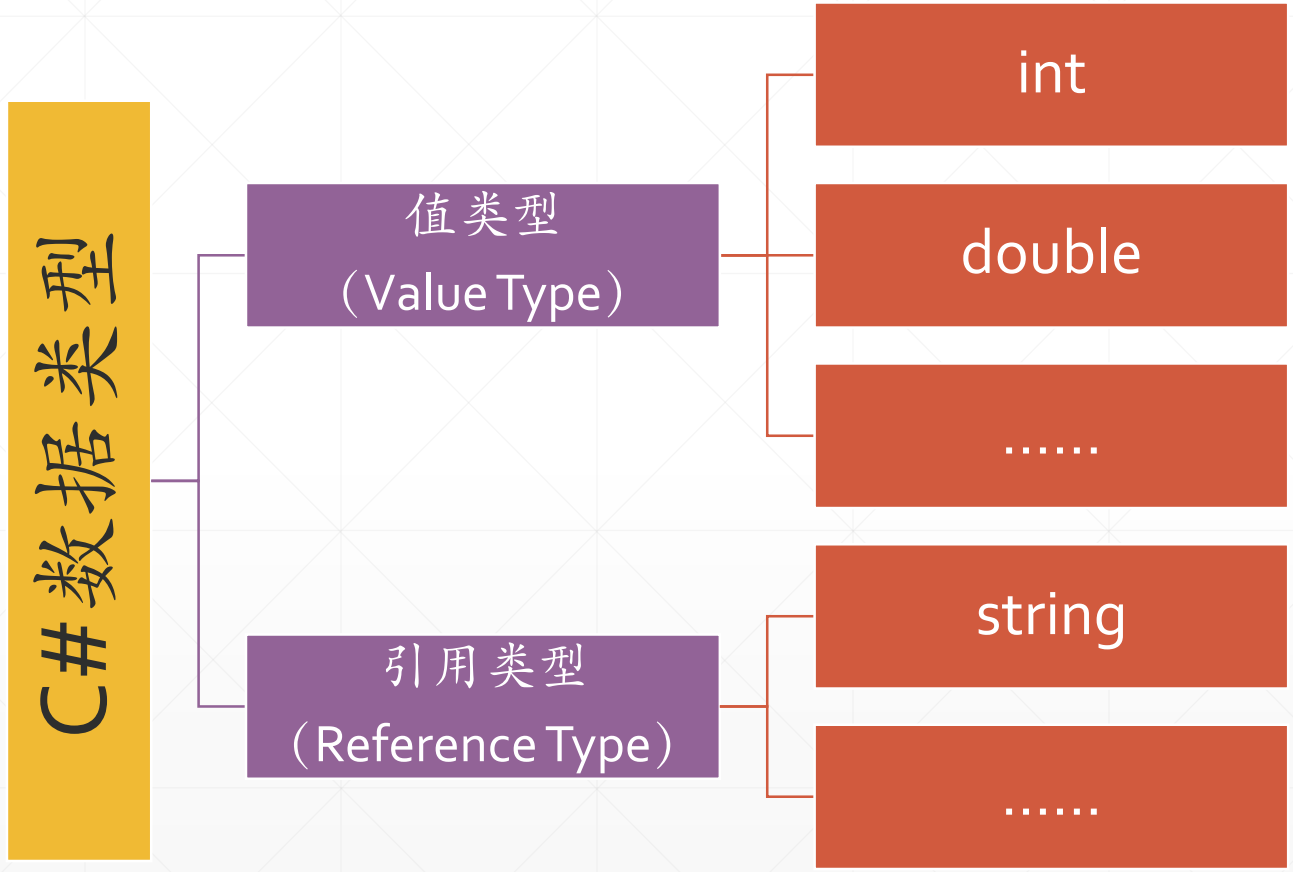
```
//方法一：使用Parse方法
int intValue = int.Parse(strValue);
double doubleValue = double.Parse(strValue);
//方法二：使用Convert类
intValue = Convert.ToInt32(strValue);
doubleValue = Convert.ToDouble(strValue);
```

数值类型 -->string

```
//方法一：调用ToString()方法
strValue = intValue.ToString();
//方法二：将数值类型变量与一个空字符串相加
strValue = doubleValue+"";
```

Convert类包容诸多基本数据类型转换的方法，请课后自己看看。

从内存模型角度，C#变量可分为以下这两种类型



值类型变量存活于“线程堆栈”中

引用类型所引用的对象，生存于“堆”中

3 C#运算符与表达式

C#支持以下运算符（仅列出部分）

- 加（+）、减（-）、乘（*）、除（/）
- 取模（%）
- 自增（++），自减（--）

使用“/”进行除法计算时，注意区分是不是“整除”。

自增自减运算符放在变量前和变量后，其运算顺序是不一样的。

从变量到表达式

使用C#运算符将变量和数字等连接起来，就构成了“**表达式（Expression）**”

```
int value = 100;
Console.WriteLine(value * 2);
```

表达式

不管表达式本身有多长，有多复杂，它最后一定会有一个值，就象变量一样，可以用在各种语句中。

注意：表达式没有“分号”，语句才有分号，有些表达式加上分号，它就变成了“**表达式语句**”。

表达式用在哪？

在实际开发中，表达式几乎无处不在。

对于初学编程的人来说，使用表达式最常见的场景就是在条件选择语句和循环语句中用于判断特定的条件是否满足。