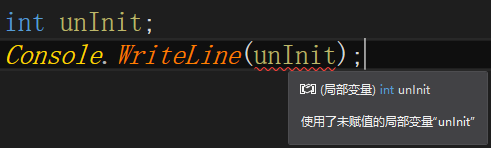
**第二章 变量**

**2.1.2 int-整数**

标识符：int

注意：不可以使用未经初始化的变量



int表示的发范围：int为4字节数据，共32位，最高位为符号位，因此表示数据范围为：-2^31 ~2^31 – 1

负数在内存中的表示：按位取反+1

1. 将该数的绝对值表示成二进制形式；
2. 按位取反（即 1 变为 0，0 变为 1）；
3. 再加 1。

**2.1.3 short-短整型**

short 型与int的唯一区别就是该变量占用 2 个字节，最高位表示正负号，后面的 15 位表示数值大小，取值范围是–2^15 ~2^15 –1，即–32768~32767。

**2.1.4 long-长整型**

long 型变量占用 8 个字节，最高位表示正负号，后面的 63 位表示数值大小，故范围可达–2^63 ~2^63 –1

**2.1.5 无符号型**

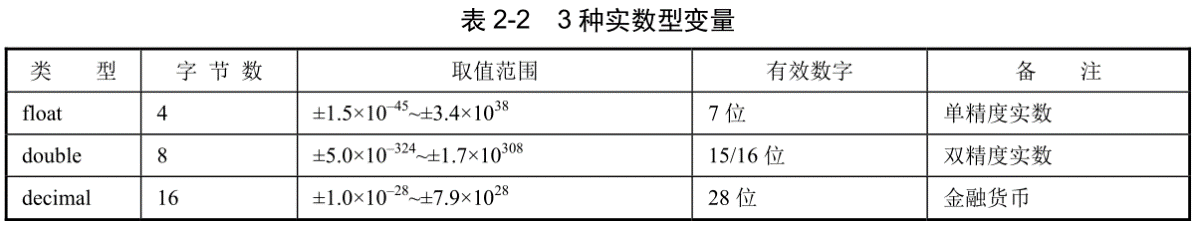
无符号型用来表示非0整数，对应于int、short、long分别是uint、ushort、ulong，范围分别是2^32 –1、2^16 –1、2^64 –1

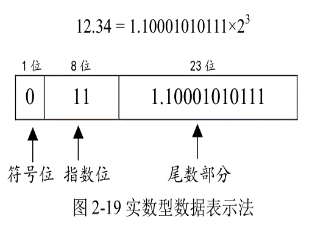
**2.1.6 整数类型小结**



**2.2 实型变量**

实数也称为浮点数，在 C#中有 3种实数型变量类型：float、double 和 decimal



 float 型变量在内存中占 4 个字节（32 位），与整型数据的存储方式不同，实数型数据是按照指数形式存储的。系统把一个实数型数据分成尾数部分和指数部分，尾数部分用二进制表示，指数部分用 2 的幂次表示，分别存放单精度 float 型最低精确到 10^–45 ，最高可达 10^38 ，有效数字为 7 位。

双精度 double 型变量在内存中占 8 个字节（64位），取值范围和有效数字都相应扩大，最低精确到10 ^324 ，最高可达 10^308 ，有效数字为 15 位或 16 位。

实数常量默认情况下是双精度的，为了把实数常量赋给单精度变量，需要添加后缀 f

或 F，将其标志为单精度实数，比如 12.34567f、3.14159265358979F 等。由于单精度浮点有效位数是7位，7位后面的数会被四舍五入。

注意：float和double在进行运算时，最后一位很可能会被舍弃，如有必要可以使用decimal。

**2.3 字符和字符串**

字符、字符串常用函数：

Console.Read()——读取一个字符用函数，需要注意的是该函数的返回值是字符的 ASCII编码，是一个 int 型的值

Console.ReadLine()——读取并返回字符串

Console.WriteLine()——向控制台写一行

Console.Write()——不换行写控制台

Convert.ToDouble（）——将字符转换为double

Convert.ToInt32()函数能把字符串转换为 int 型数值

**2.3.6 转义字符**



2.3.7 @控制符

添加前缀@后，字符串就变成了原义字符串，不再理会里面的转义字符

使用@控制符，可提高代码的可读性。需要注意的是，添加前缀@后，如果字符串里

需要引号本身，可以用连续两个引号表示

2.5 变量的格式化输出

int number=100;

Console.WriteLine("The number is " + number); // +运算符

Console.WriteLine("The number is {0}", number); //{0}占位符

1. 控制数值位数

我们可以向占位符中添加格式字符串来控制数值的位数，其格式为{i, w}，其中 i 是参数索引，w 是宽度值（位数），正值表示右对齐，负值表示左对齐

int j=60;

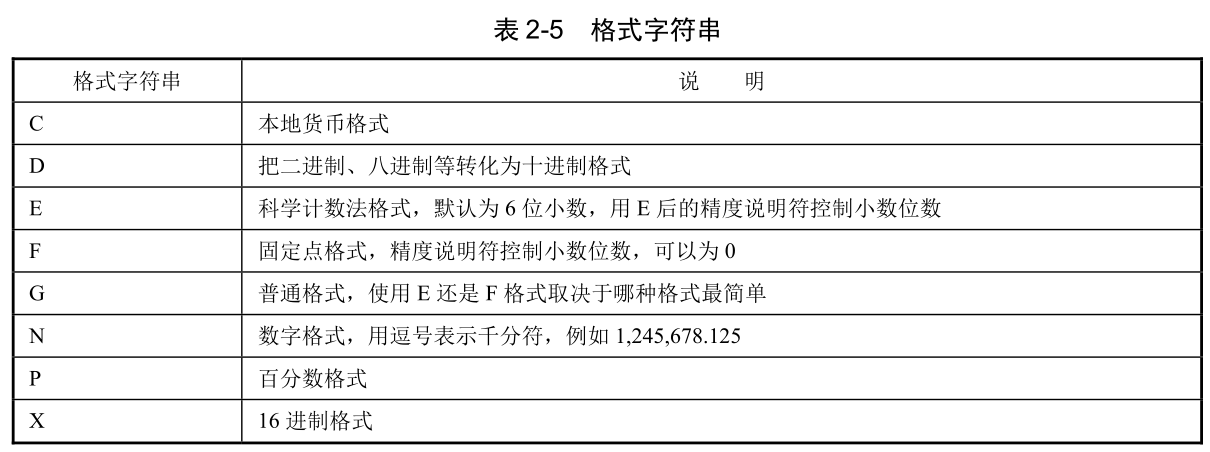
int I = 99;

Console.WriteLine("{0,6}\n +{1,3}\n------\n{2,6}", i, j, i+j);

1. 控制数值格式和小数位数

添加格式字符串和精度值可以控制数值的显示格式和小数位数

"{0,8:C2}\n +{1,7:C2}\n ––– –––––\n {2,8:C2}"



**2.8 关键字（K eyw ord）小结**

