第2学时　Perl的基本构件：数字和字符串

每种编程语言，以及人类的每种语言，都有一个相似的出发点，那就是必须要有谈话的要素。在Perl中，数字和字符串就是谈话的基本单位，这些基本单位称为标量。

标量是Perl的基本谈话单位。本书中的每个学时都要涉及到标量，对标量可以进行增加、减少、查询、测试、集中、清除、分隔、折叠、排序、保存、加载、输出和删除等操作。标量是Perl的单个名词，它们可以代表一个单词、一个记录、一个文档、一行文本或者一个字符。

Perl中的标量能够代表直接量数据，它在程序的生命期内是不变的。有些编程语言将这些值称为常量或直接量。直接量数据可以用于表示没有变化的值，比如 π的值，物体落地的加速度和美国第15届总统的名字等。如果一个Perl程序需要这些值，那么在程序的某个位置上可以用一个标量直接量来代表它们。

Perl中还有另一些类型的标量是变化的，它们称为标量变量。变量可以在你对它进行操作时用来存放数据。可以改变变量的内容，因为它们只是作为它们代表的数据的句柄而存在的。变量要被赋予相应的名字，这些名字比较简单，而且很容易记住，它们可以帮助你引用你要操作的数据。

本学时还要介绍Perl的运算符。运算符是Perl语言中的一种动词，运算符取出Perl的名词，负责从事你在编写执行特定任务的程序时需要进行的实际操作。

在本学时中，将学习下列内容：

- 直接量数字和字符串。
- 标量变量。
- 运算符。

2.1　直接量

Perl拥有两种不同类型的标量常量，它们都称为直接量。一种是数字直接量，一种是字符串直接量。

2.1.1　数字

数字直接量都是一些数字，Perl可以接受若干种不同的数字写法。表 2-1显示的所有例子在Perl中都是有效的数字直接量。

表2-1　数字直接量示例

数　字	直接量的类型
6	整型数
12.5	浮点数
15.0	另一个浮点数
0.7320508	也是一个浮点数
1e10	科学记数法
6.67E-33	科学记数法（e或E均可以）
4_294_296	带有下划线而不是逗号的大数字

数字可以根据你设想的样子来加以表示。整数是一些连续的数字。至于浮点（十进制）数，可以按照通常的形式使用小数点。科学记数法用一个指数字母 e和一个尾数（对数的十进制部分）来表示。至于大整数，可以在通常使用逗号的地方换上下划线，以便于阅读。当使用数字值的时候，Perl会删除这些下划线。

在数字前面不要使用前导 0，比如010。对于Perl来说，这个数字代表一个八进制数字，它的基数是 8。Perl还允许使用十六进制直接量数字（基数是16）和二进制数字（基数是2）。关于这些数字的详细信息请参见 perldata部分的在线文档。

2.1.2 字符串

Perl中的字符串直接量是指原义字符构成的串。它们能够包含你所想要的那么多数据。字符串的长度实际上是没有限制的，不过不能超出计算机中的虚拟内存的容量。字符串也可以包含任何种类的数据，比如简单的 ASCII文本，最高位为1的ASCII码，甚至二进制数据。字符串也可以是空的。

应该用引号将字符串直接量括起来，不过在 Perl中也有很少的一些例外。这个过程称为给字符串“ 加引号 ”。给字符串加引号有两种主要方法，一种是使用单引号（''），一种是使用双引号（""）。下面是字符串直接量的一些示例：

```
"foo!"
'Fourscore and seven years ago'
"One fish,\nTwo fish,\nRed fish,\nBlue fish\n"
""
"Frankly, my dear, I don't give a hoot.\n"
```

在双引号中，如果需要插入另一个引号，则必须使用反斜杠转义符。字符串直接量中的反斜杠（\）用于告诉Perl，它后面的字符不应该按通常的情况来处理，在这种情况下，它应该被忽略。例如，下面这个字符串直接量对 Perl来说就没有任何意义：

```
"Then I said to him, "Go ahead, make my day""
```

在这个字符串中，单词 Go前面的引号与第一个引号是一对，从而将 Go ahead,make my day这个短语留在引号的外面，因此这不是一个有效的 Perl。为了防止这种情况的发生，请在你想要让Perl不当成引号对待的引号的前面放一个反斜杠，如下所示：

```
"Then I said to him, \"Go ahead, make my day.\""
```

反斜杠使得Perl能够知道后面的引号与第一个引号不是匹配的一对，因此它应该不被作为引号处理。这条规则既适用于单引号，也适用于双引号，请看下面的例子：

```
'The doctors\'s stethoscope was cold.'
```

给字符串加双引号和单引号的主要差别是：使用单引号的字符串含义是非常直观的，单引号字符串中的每个字符就是表示它自己的含义。在双引号中的字符串中， Perl要查看是否存在变量名或转义序列。转义序列是一些特殊字符串，你可以将难以键入和以后难以识别的字符嵌入字符串。表2-2显示了一个Perl的转义序列的简短列表。

表2-2 示例字符串的转义序列

序 列	含 义
\n	换行
\r	回车
\t	制表符
\b	退格
\u	将下一个字符改为大写
\l	将下一个字符改为小写
\\	直接量反斜杠字符
\'	用单引号（''）括起来的字符串中的直接量'
\"	用引号括起来的字符串中的直接量"

可以从在线手册中找到转义序列的完整列表。正如在第一学时中介绍的那样，可以使用Perl中包含的perldoc实用程序，找到整个Perl语言的文档资料。转义序列位于“ perlop ”手册中的“ Quote and Quote-like Operators（引号与引号式的运算符）”这个标题下。

如果在字符串中包含许多个引号，那么当每个嵌入的引号必须转义时，键入字符串就会非常困难，而且很容易发生错误。请看下面这个例子：

```
'The doctors\'s stethoscope was cold.'
```

Perl还提供了另一个引号机制，即 qq和q运算符。使用 qq运算符，就可以用 qq（）而不是引号将字符串括起来：

```
qq(I said, "Go then," and he said "I'm gone")
```

qq取代了双引号。这个机制的作用几乎在所有方面都与双引号完全一样。

也可以用q运算符来代替单引号将文本括起来：

```
q(Tom's kite wedged in Sue's tree)
```

qq和q运算符可以使用任何非字母、非数字字符来标记字符串的开始和结束。这些标记称为界限符。在前面这个例子中使用了括号，不过也可以使用任何其他的非字母或非数字字符作为界限符：

```
q/Tom's kite wedged in Sue's tree/

q,Tom's kite wedged in Sue's tree,
```

你想用作界限符的字符必须出现在紧靠运算符 qq或q的后面。使用任何一个成对的界限符，如（），<>,{},[]，它们都能够正确地将字符串括起来。这就是说，如果在 qq或q运算符中以偶数对的形式使用这些界限符，就不必使用反斜杠转义符：

```
q(Joe (Tom's dad) fell out of a (rather large) tree.);
```

但是，加上这些界限符后， Perl程序的可读性就会有所降低。通常来说，只选择字符串中不出现的界限符，读起来就比较容易。

```
q[Joe (Tom's dad) fell out of a (rather large) tree.];
```

2.2 标量变量

若要将标量数据存放在 Perl中，必须使用标量变量。在 Perl中，如果要写一个标量变量，

可以写一个美元符号，后跟变量的名字。下面是一些标量变量的例子：

```
$a
$total
$Date
$serial_number
$cat450
```

美元符号称为类型标识符，用于告诉 Perl该变量包含标量数据。其他变量类型（哈希变量和数组）则使用不同的标识符，甚至根本没有标识符（文件句柄）。Perl中的变量名，比如哈希变量、数组、文件句柄和标量，必须符合下列规则：

- 变量名可以包含字母（a至z,A至Z）字符、数字或类型标识符后面的一个下划线字符（_）。不过，变量名的第一个字符不能是数字。
- 变量名是区分大小写字母的。这意味着变量名中的大写和小写字母都是有特定意义的，因此下面这些变量均代表不同的标量变量：

```
$value
$VALUE
$Value
$valuE
```

Perl只使用单字符变量名，它们不是以字母字符或下划线开始的。以 $_、$ ”、$/、$2和$$开始的变量均属于特殊变量，在 Perl程序中不能用作普通变量。这些特殊变量的作用将在下面介绍。

Perl与某些其他编程语言不同，Perl中的标量变量在你使用它们之前，不必预先进行声明或初始化。若要创建一个标量变量，只要使用它就行了。当使用一个未经初始化的变量时，Perl将使用它的默认值。如果它被用作数字时（如数学运算中的数字），Perl将使用0（零）这个值；如果它被用作字符串（几乎其他任何地方都使用），那么Perl将使用“ ”这个值，即空字符串。

在用一个值对变量进行初始化之前就使用该变量，这被认为是一种不好的编程做法。当你在命令行上使用 -w开关，或者在程序开头的 #!行上使用-w来调用Perl程序时，Perl就会向你发出警告。如果你试图使用的变量值预先没有进行设定，那么当你的程序运行时以及试图使用该值时， Perl就会报一条出错消息Use of uninitialized value（使用未经初始化的值）。

特殊变量$_

Perl拥有一个特殊变量，它的值可以用作“ 默认值 ”。对于许多运算符和函数来说，该变量称为$变量。例如，如果你只是使用输出函数本身，而不设定要输出什么，那么它将输出 $ 的当前值：

```
$_="Dark Side of the Moon";
print;    # Prints the value of $_, "Dark side..."
```

像这样来使用 $_，肯定会造成某些混乱。因为它确实没有指明究竟输出的是什么，尤其是给$_的赋值出现在程序中较高的位置上时更是如此。

有些运算符和函数实际上可以比较容易地用于 $_变量，尤其是第6学时中介绍的模式匹配运算符。不过在本书中将尽可能少用 $_，这样学习起来就比较容易一些。

2.3 表达式和运算符

既然你已经学习了什么是标量数据并且知道如何使用标量变量，那么现在就可以开始用Perl来执行一些有用的操作了。 Perl程序实际上是一些表达式和语句的集合，它们从 Perl程序的顶部向底部顺序执行，不过你也可以设定流控制语句的执行顺序，这将在第 3学时“ 控制程序流 ”中介绍。程序清单2-1显示了一个有效的Perl程序。

程序清单2-1 一个简单的Perl程序

```
1:   #!/usr/bin/perl -w
2:
3:   $radius=50;
4:
5:   $area=3.14159*($radius ** 2);
6:   print $area;
```

第1行：这一行是到达 Perl解释程序的路径，这在第1学时中已经做了介绍。开关 -w告诉Perl，只要遇到警告就通知你。

第3行：这一行是个赋值行。数字标量数据 50存放在标量变量$radius中。

第5行：这一行是另一个赋值行。在赋值运算符的右边是个表达式，该表达式包含了标量变量$radius、运算符（*和**，下面将介绍这两个运算符）和数字标量（ 2）。该表达式的值经过计算后被赋予$area。

第6行：这一行负责输出存放在 $area中的计算结果。

表达式是具有值的简单元素。例如， 2是个有效的表达式。 54*$r、“ Java ”、sin（$pi*8）和$t=6等，也都是表达式。表达式的值是在程序运行时进行计算的。程序对表达式中的函数、运算符和标量常量进行计算，然后得出一个值。可以将这些表达式用于赋值，或者作为其他表达式的一个组成部分，也可以作为其他 Perl语句的组成部分。

2.3.1 基本运算符

正如你在程序清单 2-1中看到的那样，若要将标量数据赋予一个标量变量，可以使用赋值运算符=。这个赋值运算符取出位于右边的值，再将它放入左边的变量中：

```
$title="Gone With the Wind";
$pi=3.14159;
```

赋值运算符左边的操作数必须是能够被赋予变量的一个值。右边的操作数可以是任何种类的表达式。赋值的本身就是一个表达式，它的值是右边表达式的值。这就是说，在下面这个代码段中， $a、$b和$c都被设置为42。

```
$a=$b=$c=42;
```

这个代码段中， $c首先被设置为42。$b被设置为表达式$c=42（它就是42）的值，然后$a被设置为表达式$b = 42的值。被赋值的变量甚至可以出现在赋值运算符的右边，如下所示：

```
$a=89*$a;
$count=$count+1;
```

赋值运算符的右边使用 $a或$count的老的值进行计算，然后，它的结果作为新值被赋予左边。第二个例子在 Perl中有一个特殊的名字，它称为递增。在后面我们还要更加详细地介绍递增。

2.3.2 数字运算符

Perl中有许多运算符可以用来对数字表达式进行操作。这些运算符中，有些你已经非常熟悉，有些是第一次遇到。已经知道的第一种运算符是数学运算符。表 2-3显示了这些运算符的一个列表。

表2-3 数学运算符

举 例	运 算 符 名	表达式的值
5 + $t	加	5与$t相加的和
$y - $x	减	$y与$x的差
$e * $pi	乘	$e与$pi的积
$f / 6	除	$f除以6得出的商
24 % 5	求余数	24除以5得出的余数（4）
4**2	取幂	取4的二次方

数学运算符是按照我们通常的规则进行计算的，先取幂，再乘除，然后求余数，最后进行加减。如果你不清楚表达式中各个元素的计算顺序，可以使用括号将元素括起来，以确保计算顺序的正确。嵌套的括号总是从里向外进行计算：

```
5*6+9;          # Evaluates to 39
5*(6+9);        # Evaluates to 75
5+(6*(4-3));    # Evaluates to 11
```

2.3.3 字符串运算符

在Perl中，数字值并不是可能受运算符影响的惟一值，字符串也可以进行相应的操作。第一个字符串运算符是并置运算符，用圆点（ .）来代表。并置运算符取出左边的字符串和右边的字符串，返回一个将两个字符串合并在一起的字符串：

```
$a="Hello, World!";
$b=" Nice to meet you";
$c=$a . $b;
```

在这个例子中， $a和$b被赋予一些简单的字符串值。在最后一行上， $a和$b并置在一起，变成Hello,World! Nice to meet you，然后存放在$c中。$a与$b没有被并置运算符修改。

并置式的操作特性可以用另一种方式来执行。在前面，你了解到 Perl在双引号字符串中查找变量。如果Perl在双引号字符串中找到了一个变量，那么它将被内插替换。这就是说，双引号字符串中的变量名将被它的实际值代替：

```
$name="John";
print "I went to the store with $name.";
```

在这个例子中， Perl查看双引号字符串，发现 $name，并对字符串John进行替换。这个过程称为变量内插替换。为了防止变量查找的字符串被内插替换，可以使用单引号（它不进行任何形式的内插替换），也可以在变量标识符的前面加上一个反斜杠：

```
$name="Ringo";
print 'I used the variable $name';      # Will not print "Ringo"
print "I used the variable \$name";     # Neither will this.
```

上面这个例子中的两个print语句均用于输出I used the variable $name，字符串$name没有被内插替换。因此，若要在不使用并置运算符的情况下执行并置操作，只要使用双引号字符串即可，如下所示：

```
$fruit1="apples";
$fruit2="and oranges";
$bowl="$fruit1 $fruit2";
```

如果Perl不能清楚地指明变量名在何处结束和字符串的其余部分从何处开始，那么可以使用花括号将变量名括起来。使用这个句法， Perl就能够找到可能模糊的变量名：

```
$date="Thurs";
print "I went to the fair on ${date}day";
```

如果没有花括号，Perl将不知道是要对双引号中的$date还是对变量$dateday进行内插替换。加上花括号后，内插替换的对象就清楚了。

这里出现的另一个字符串运算符是重复运算符 x。运算符x配有两个参数，一个是要重复的字符串，另一个是该字符串重复的次数，请看下面这个例子：

```
$line="-" x 70;
```

在上面这个例子中，字符-被运算符x重复70次。其结果存放在$line中。

2.4 其他运算符

Perl的运算符非常多，由于本书的篇幅有限，因此无法一一列举。在本学时剩下的时间里，将要介绍Perl中最常用的一些运算符和函数。

2.4.1 单参数运算符

到现在为止，我们介绍的所有运算符都带有两个参数。除法（ 6/3）需要一个被除数（6）和一个除数（3），乘法（5*2）需要一个被乘数（5）和一个乘数（2），如此等等。另一种运算符只有一个参数。你可能已经熟悉这种运算符的一个例子，即一目减运算符（ -)。一目减运算返回变成负数的参数值：

```
6;      # Six
-6;     # Negative six.
-(-5);  # Positive five, not negative five.
```

Perl的许多运算符实际上是带名字的运算符，也就是说，它们不是一个符号，比如一目减操作中的-符号，它们是一个单词。带名字的一目减运算符的操作数前后的括号是可有可无的，但是为了清楚起见，表2-4中都将括号显示了出来。由于Perl中带名字的运算符和函数看上去非常相似，因此带名字的运算符的操作数有时也称为变元，这是Perl函数用于它们的参数的一个术语。

表2-4显示了某些带名字的运算符的一个简单列表。

表2-4 一些带名字的运算符

运 算 符	用 法 举 例	结 果
int	int（5.6234）	返回它的参数的整数部分（5）。
length	length("nose")	返回它的字符串参数的长度（4）。
lc	lc（"ME TOO"）	返回它的转换成小写字母的参数（" me too "）
uc	uc（"hal 9000"）	返回与lc相反的参数值（"HAL 9000"）
cos	cos（50）	返回弧度50的余弦值（.964966）
rand	rand（5）	返回从0到小于该参数值之间的一个随机数字。如果该参数被省略，则返回 0至1之间的一个数字

可以从在线手册中找到带名字的运算符的完整列表。我们在第 1学时中讲过，可以使用Perl产品中包含的perldoc实用程序找到Perl语言的完整文档。运算符的完整列表位于“ perlop ”和“ perlfunc ”手册中。其他运算符将根据需要在后面的一些学时中介绍。

2.4.2 递增和递减

在“ 数字运算符 ”这一节中，我们讲到了一种特殊的赋值类型，称为递增，它类似下面的形式：

```
$counter=$counter+1;
```

递增通常用于计数，比如读取的记录数，或者生成序列号，比如给列表中的项目编号。在Perl中，它是个非常常用的术语，因此可以使用一个特殊运算符，称为自动递增运算符(++)。自动递增运算符能够给它的操作数递增 1：

```
$counter++;
```

在执行这个代码后， $counter递增1。

Perl还提供了一种快捷运算符，用于递减一个变量的值，这个运算符称为自动递减运算符(--)。自动递减运算符的使用方法与自动递增运算符的使用方法完全相同：

```
$countdown=10;
$countdown--;      # decrease to 9
```

关于自动递增运算符还有最后一个问题需要加以说明，那就是当你将它用于一个文本字符串，而该文本字符串是以字母字符开始，后随字母字符或数字，那么这个运算符就具有一种非常特殊的作用。字符串的最后一个（最右边的）字符被递增。如果它是个字母字符，它将成为序列中的下一个字母；如果它是个数字，那么该数字将递增 1。你可以像下面这样通过各个字母列和数字列：

```
$a="999";
$a++;
print $a;           # prints 1000, as you'd expect
$a="c9";
$a++;
print $a;           # prints d0.  9+1=10, carry 1 to the c.
$a="zzz";
$a++;
print $a;         # prints "aaaa".
```

自动递减运算符并不像上面那样对字符串进行递减。

2.4.3 尖括号运算符

尖括号运算符（<>），有时也称为菱形运算符，主要用于读写文件。我们将在第 5 学时详细介绍这个运算符。不过现在要对它做一个简单的介绍，这会使我们的练习更加有趣。当你开始学习第5学时的内容时，就会对该运算符有所了解。

到那时，你将能够用它的最简单的形式，即 <STDIN>来使用尖括号运算符。这种形式告诉Perl，应该从标准输入设备（通常是键盘）读取一行输入信息。 <STDIN>表达式返回从键盘读取的这行信息：

```
print "What size is your shoe? ";
$size=<STDIN>;
print "Your shoe size is $size, Thank you";
```

上面这个代码在执行的时候（假设某人键入 9.5作为它的鞋子的尺寸），将类似下面的形式：

```
What is size is your shoe?  9.5
Your shoe size is 9.5
Thank you
```

<STDIN>表达式从键盘读取信息，直到用户按下 Enter键为止。整个输入行返回，并被存放在$size中。由<STDIN>返回的文本行也包含用户键入的换行符（因为按下了 Enter键）。在大多数情况下，你不希望在字符串的结尾处出现换行符。若要删除换行符，可以像下面这样使用chomp运算符：

```
print "What size is your shoe?";
$size=<STDIN>;
chomp $size;
print "Your shoe size is $size, or so you say.\n";
```

chomp运算符能够删除它的参数结尾处的任何换行符。它返回被删除的字符数，这个数字通常是1，但是，有时如果没有字符需要删除，那么返回的是 0。

2.4.4 其他赋值运算符

前面我们曾经讲过，可以使用赋值运算符（ =），将一个值赋予一个标量变量。实际上 Perl拥有一组完整的运算符，可以用于进行赋值操作。 Perl的每个数学运算符和相当多的其他运算符可以组合起来同时进行赋值和运算操作。下面是建立一个带有运算符的赋值语句的一般规则：

变量　运算符=表达式

这个规则与下面这个规则相同：

变量=变量　运算符　表达式

使用赋值语句通常不会使程序更加容易阅读，但是它能够使程序更加简洁。例如，按照这个规则，语句$a=$a+3；可以简化为 $a+=3。

下面是另一些赋值语句的例子：

```
$line.=", at the end";          # ", at the end" is appended to $line
$y*=$x                          # same as $y=$y*$x
$r%=67;                         # Divide by 67, put remainder in $r
```

2.4.5 关于字符串和数字的一些说明

总的来说， Perl允许你对数字和字符串进行互换使用。它使用的表示法取决于在这种情况下Perl查找的是什么。

- 如果某个元素看上去是个数字，那么 Perl在需要数字时可以将它用作数字：

```
$a=42;          # A number
print $a+18;    # displays 60.
$b="50";
print $b-10;    # Displays 40.
```

- 如果某个元素看上去是个数字，那么当 Perl需要一个字符串时，它可以使用数字的字符串表示法：

```
$a=42/3;
$a=$a . "Hello";  # Using a number like a string.
print $a          # displays "14Hello"
```

• 如果某个元素看上去不像一个数字，但是你将它用在需要数字的地方，那么 Perl在它的位置上使用0这个值：

```
$a="Hello, World!";
print $a+6;      # displays the number 6
```

但是，如果你激活了警告特性，那么如果你这样操作的话， Perl就会发出警告消息。

所有这些用法都与Perl的“尽可能随意自然”的原则是一致的。即使毫无意义，就像最后一个例子中的情况那样， Perl也设法用它来做一些有意义的事情。如果在 Perl程序中激活了警告特性，在 #！行上加上了一个-w开关，或者运行带有-w选项的Perl程序，Perl就会发出警告，说明你的操作毫无意义，并给出下面这条消息： Argument x isn ’t numeric（参数x不是个数字）。

2.5 练习：利息计算程序

这个练习是进行复利的计算。这个程序将根据利率、存款和时间等信息来计算储蓄帐户的利息。下面是你要使用的计算公式：

$$\text{accrued} = \text{payment}\left(\frac{(1+\text{monthly interest})^{\text{number of deposits}-1}}{\text{monthly interest}}\right)$$

使用文本编辑器，键入程序清单 2-2中的程序，并将它保存为 Interest。不要键入行号。根据你在第1学时中学习到的方法，使该程序成为可执行程序。

程序清单2-2 利息计算程序的完整源代码

```
1:   #!/usr/bin/perl -w
2:
3:   print "Monthly deposit amount? ";
4:   $pmt=<STDIN>;
5:   chomp $pmt;
6:
7:   print "Annual Interest rate?  (ex. 7% is .07) ";
8:   $interest=<STDIN>;
9:   chomp $interest;
10:
11:  print "Number of months to deposit? ";
12:  $mons=<STDIN>;
13:  chomp $mons;
14:
15:  # Formula requires a monthly interest
16:  $interest/=12;
17:
18:  $total=$pmt * ( ( 1 + $interest) ** $mons -1 )/ $interest;
19:
20:  print "After $mons months, at $interest monthly you\n";
21:  print "will have $total.\n";
```

第1行：这一行包含了到达解释程序的路径（可以对它进行修改，使它适合你的系统的需要）和-w开关。请始终使警告特性处于激活状态。

第3行：这一行提示用户输入一个金额。

第4行：从标准输入设备（键盘）读取 $pmt。

第5行：从$pmt的结尾处删除换行符。

第7至9行：从键盘读取$interest，同时删除换行符。
第11至13行：从键盘读取$mons，同时删除换行符。
第16行：$interest除以12，并且重新存入$interest。
第18行：执行利息计算，将结果存入 $total。
第20至21行：输出计算结果。
当你完成操作时，在一个命令行上键入下面的命令，设法运行该程序：

```
perl Interest
```

程序清单2-3显示了一个利息计算程序输出信息的例子。

程序清单2-3 利息计算程序的输出

```
1:   Monthly deposit amount? 180
2:   Annual Interest rate?  (ex. 7% is .07) .07
3:   Number of months to deposit? 120
4:   After 120 months, at 0.005 annual you
5:   will have 29498.2824251624.
```

2.6 课时小结

在本学时中，我们了解到Perl的最基本的数据类型是标量。标量几乎可以是任何类型的数据，并且可以存放在标量变量中。数字直接量可以用许多不同的格式来表示，比如整数、浮点数等。字符串直接量用加引号的字符串来表示，字符串的前后既可以使用双引号，也可以使用单引号。另外，Perl提供了许多运算符，用于进行字符串的操作和基本的算术运算。

2.7 课外作业

2.7.1 专家答疑

问题：利息计算程序的输出显得有些杂乱,我如何才能控制它显示几位小数点？

解答：控制小数点位数的最简单的方法是使用 printf函数。我们将在第 9学时中介绍这个函数。

问题：Perl是否有一个用于数字舍入的函数呢？

解答：当显示数字时，printf函数通常可以用来进行你想要做的舍入操作。如果你确实需要round函数，请查看POSIX模块，它包含了该函数和其他许多函数。

问题：Perl允许我操作的数字究竟可以有多大（或者多小）？

解答：答案取决于你使用的是什么操作系统。典型的 Intel UNIX系统的双精度浮点数的指数幂可以超过300位。这就是说你操作的数字小数点的右边（或左边）可以有 300个0。不过大数字的精度通常只有14位数。

2.7.2 思考题

1) 变量可以在qq引号中进行内插替换。
   a. 是
   b. 否

2) 运行下面这个代码后，什么值将存放在 $c中？

```
$a=6;
$a++;
$b=$a;
$b--;
$c=$b;
```

a. 6

b. 7

c. 8

3) 并置操作只能使用并置运算符（.）进行。

a. 是

b. 否

2.7.3 解答

1) 答案是a。qq的作用与一对双引号完全相同。这意味着它能够对变量进行内插替换。

2) 答案是a。$a被设置为6，然后$a递增为7，并被赋值给$b。$b递减为6，并被赋值给$c。

3) 答案是b。在第1学时中我们讲过，在 Perl中，一项操作可以使用多种方法来进行（这句英文可以缩写为 TIMTOWTDI）。并置操作可以在双引号字符串中包含两个（或多个）标量，如下面所示：

```
qq（$a$b$c):
```

2.7.4 实习

- 请编写一个短程序，提示用户输入一个华氏温度值，并输出摄氏温度值。若要将华氏变成摄氏，你可以将华氏温度减去 32，然后乘以 5/9。例如，华氏75度等于摄氏21.1度。
- 修改程序清单 2-3中的利息计算程序，输出的金额小数点不要超过两位。你可以不使用printf函数，而巧妙地使用 int运算符、乘法和除法来进行计算。