一起++ 第4弹

运算

作者：@孤言　　审校：@Alex Cui

对数据的处理离不开运算。在变量一弹中，我们已经至少学会了部分赋值运算符，今天我们将更加系统地来学习运算符的知识，并了解一些常用函数。

知识点 1　认识运算符

运算符是一种告诉编译器执行特定的数学或逻辑操作的符号。C++支持多种运算符。用于运算符的值称为**操作数**。

1.算数运算符

算术运算符你已经差不多“学过”了，有 `+`，`-`，`*`，`/` 和 `%`. 前四种运算符的用法与数学上写算式的用法类似。操作数既可以是常量，也可以是变量。

需要特别注意的是，对于`/`运算符，当两个操作数均为整数时，结果是一个整数（小数部分被丢弃，正数时结果看似是向下取整的）。比如表达式 5/4 的值为 1 而不是 1.25，因为小数部分被丢弃。而操作数中至少一者为小数时，运算结果才可能为小数。

在算数运算符里，你不太熟悉的可能是`%`了。它是**求模**运算符。类似于小学低年级曾经做过的取余数的运算。比如 11%4 的值为 3，因为 11 除以 4 等于 2 余 3，余数为 3，因而值为 3. 写成 Scratch 如图 1（当然负数时取余数不一定等价，本弹不作探讨）。

图 1

注意 **%的两个操作数均应为整数。**

下面的程序展现了如何将多少秒转化为多少分钟多少秒。

示例 4-1　算数运算符的应用

```
1   #include <iostream>
2
3   using namespace std;
4
5   int seconds, mins; //声明整型变量秒、分钟
6
7   int main()
8   {
9       cin >> seconds;
10      mins = seconds / 60;
11      //效果上是计算秒数除以 60 向下取整的结果，赋给分钟
12      seconds = seconds % 60; //取模得到剩余秒数
13      cout << mins << "mins "
14           << seconds << "seconds" << endl;
15      return 0;
16  }
```

2.赋值运算符

在变量一弹中，我们已经介绍了一部分赋值运算符的用法，这里不再赘述。~~(其实是作者比较懒)~~

3.关系运算符

关系运算符用于比较操作数间的关系。比较易懂的有`>`，`<`，`>=`，`<=`。直接望文生义基本不成问题。

另两个常用的关系运算符分别是 `==`（等于）和 `!=`（不等于）。

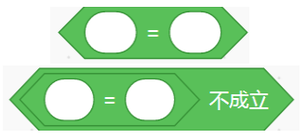

图 2

注意 **=是赋值运算符，==才是关系运算符**

这些运算符与第 4 点的逻辑运算符都将在接下来的分支语句一弹中发挥作用。

4.逻辑运算符

在生活中，我们会有类似于以下的逻辑感受：“如果**不**下雨，就去打篮球”，“买满 300 元**并且**关注了公众号可以获得一张优惠券”……这些逻辑在程序中也有体现。

逻辑运算符臭名昭著的三兄弟——“与、或、非”来了。它们对应的符号是：`&&`（与），`||`(或)，`!`（非，类似于 Scratch 中的<()不成立>）。

5.条件运算符


*图片来自网络

条件运算符，是 C++中唯一一个含有 3 个操作数的运算符，长成这样：`: ?`。其格式为：

```
表达式 1 ? 表达式 2 : 表达式 3
```

当表达式 1 成立时，整个表达式的值为表达式 2，否则为表达式 3.

例如对于如下的表达式：

```
5 == 5 ? "same" : "different"
3 > 5 ? 1 : 0
(1 > 2 || 3 != 4) ? 5 : 6
```

它们的值分别为："same"，0，5.（最后一个注意运算符之间的优先级）

C++中的运算符远不止以上所提的这些，有兴趣的同学可以去查找资料。除了按照作用来分类运算符以外，还可以按照操作数的数量来分类，分为**单目运算符**、**双目运算符**和**三目运算符**。

位运算

直接对整数在内存中的二进制位进行操作的运算称为**位运算**。C++中有多个位运算符。

1.按位与：`&`，两个都为 1 值才为 1，否则为 0.

2.按位或：`|`，两个都为 0 值才为 0，否则为 1.

3.按位异或：`^`，相同为 0，不同为 1.

它们的运算情况如下表所示。（灰色部分表示操作数）

&	1	0
1	1	0
0	0	0

|	1	0
1	1	1
0	1	0

^	1	0
1	0	1
0	1	0

例如：12|5 的值为（0000 1100)|(0000 0101)（二进制），即为(0000 1100)|(0000 0101)，运算后得到 0000 1101，即为 13（十进制）。有关进制转换的知识可自行参阅其它资料。

4.按位取反：`~`，1 变 0，0 变 1.

例如：~6 的值为 ~(0000 0110)，即为 1111 1001，转换为十进制为 249.

5.按位左移：把二进制位整体向左移动。

6.按位右移：把二进制位整体向右移动。

例如 7>>2 的值为(0000 0111)>>2，即为 0000 0001，为 1（十进制）.

你可能会想，按位左移和按位右移与我们进行流 I/O 时的运算符长得一样艾？其实，这就是我们之前谈到的运算符重载的情况。

一般认为位运算效率较高。

本弹以 8 位为例，并且不对负数情况作探讨。

知识点 2　运算顺序

在小学数学里，我们经常遇到各种算数题比如：4×(3+2)-1， 我们需要先算括号内的加法，再算乘法，最后算减法。再如 1+2-3，加、减法的优先级是相同的，此时我们从左往右运算。在 C++中，运算符的功能不仅局限于这些纯数学的运算，但具有类似的性质。

1.优先级

在四则运算上，大体顺序遵循括号内先算、乘除法后算，加减法最后算的规则。推而广之，C++中，运算符也有优先级别，多个运算符作用于同一个操作数时，先用哪个运算符由其**优先级**决定。具体的优先级可以自行查表。部分优先级**由优先到次后**归纳如下：

！（逻辑非）　算数运算符　逻辑运算符　&&（逻辑与）　||（逻辑或）　赋值运算符

算数运算符的优先级基本遵循代数运算的优先级。

2.结合性

当**优先级相同**的运算符作用于同一操作数时，先用哪个运算符由其**结合性**决定。结合性分为两种：**从左向右**和**从右向左**（一些特殊的运算符无结合性）。

从左向右的意思是，当优先级相同的运算符作用于同一操作数时，先作用于左侧的运算符。

从右向左可以类比。

运算符的优先级和结合性无需过分记忆，有时候不确定加个括号就行。

知识点 3　表达式

大概地来说，像单一的常量或变量或者运算符及其操作数组成的整体等等，这样能求得**值**的部分称为**表达式**。

我们有时会根据运算符的种类为表达式分类，如条件表达式、算数表达式等。而有时则会称常量表达式等。

知识点 4　常用函数

C++中提供了很多数学函数，用于数学运算，今天我们就来认识其中一些。

函数	表述
floor(x)	对 x 向下取整
ceil(x)	对 x 向上取整
round(x)	对 x 四舍五入为整数
abs(x)	求 x 的绝对值
pow(x,y)	求 x^y
sqrt(x)	求 x 的平方根

使用以上函数时，应包含头文件 cmath：

```
#include <cmath>
```

下面来看一个示例体会以下这类函数的用法。设有一个直角边长为 a, b 的直角三角形，求出斜边的长，并向上取整。

示例 4-2　数学函数

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
    double a, b, c;
    cin >> a >> b;

    c = sqrt(pow(a, 2) + pow(b, 2)); //勾股定理计算斜边长
    c = ceil(c);       //向上取整。注意，不要把前面的 c=遗漏

    cout << c;
    return 0;
}
```

代码练习 2

Q1 利用三目运算符，输出两个整数中较大的那个。

#输入示例
```
3 5
```
#输出示例
```
5
```

Q2 输出两正整数的带余除法结果（余数为 0 不作特殊处理）

#输入示例
```
17 4
```
#输出示例
```
17/4=4…1
```

Q3 给出圆的半径，计算其周长和面积（π 取 3.14），并保留一位小数输出（由于目前还没有学习格式化输出，所以对十分位为 0 的情况可以省略小数部分）。

#输入示例
```
5.5
```
#输出示例
```
34.5
95
```

答案

Q1 利用三目运算符。

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    int a, b, maxn;
    cin >> a >> b;
    maxn = a > b ? a : b;
    //a>b 如果成立，maxn 被赋为 a，否则赋为 b
    cout << maxn << endl;
    system("pause");
```

```
    return 0;
}
```

Q2 /对两个整数相除时会截去小数部分，%为求模运算，二者结合使用。

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    int a, b;
    cin >> a >> b;
    cout << a << "/" << b << "="
         << a / b << "..." << a % b << endl;
  //获得舍去小数部分的商       获得余数
    system("pause");
    return 0;
}
```

Q3 在 C++里实现保留几位小数输出，常用 printf()函数，由于还没有学习，所以这里采用一种颇为简陋的办法。

```
#include <iostream>
#include <cstdlib>
#include <cmath> //使用 pow()函数，需要包含 cmath

using namespace std;

int main()
{
    const double pi = 3.14;
    double r, c, s;
    cin >> r;
    c = 2 * pi * r;
    s = pi * pow(r, 2); //也可以写成 pi*r*r
    cout << round(c * 10) / 10 << endl
         << round(s * 10) / 10 << endl;
    system("pause");
    return 0;
}
```


第一版日期：2019.10.3
第二版日期：2019.11.3
第三版日期：2020.8.27
字数：2805