表达式

我们在认识程序结构的时候接触过表达式，但是我们没有展开说明，还记得在哪里吗?

赋值语句的格式：
变量=表达式；

在这里，我们要更充分地展开对 C++表达式的理解。

表达式，简单地讲，就是由通过运算符连结运算对象的式子。特别的，单个常量或者单个变量也是表达式。请看以下例子：

```
1. a=3; //单个常量 3 是表达式
2. a=b; //单个变量 b 是表达式
3. a=3+4; //运算符+号将运算对象 3 和 4 连结起来
4. a=b+3; //运算符+号将运算对象变量 b 和常量 3 连结起来
5. a=b+c; //运算符+号将运算对象变量 b 和变量 c 连结起来
```

表达式的结果是一个确定的值，如 3+4 的值就是 7。

认识表达式，关键在于认识运算符。在 C++中运算符的类型非常多，在这里我们主要讲三类运算符：**算式运算符、关系运算符和逻辑运算符。**

算式运算符

算式运算符我们比较熟悉，常见的有‘+’、‘-’、‘*’、‘/’号，还有我们不太熟悉的‘%’。

算式运算符	功能	用法
+	加法	Expr + expr
-	减法	Expr - expr
*	乘法	Expr * expr
/	除法	Expr / expr
%	求余	Expr % expr

特别的‘/’号：C++中，‘/’号有两种功能，一个是整除，当运算对象都是整型时，‘/’号表示整除，比如 5/2 的结果是 2，即求 5/2 的商，自动忽略小数部分。另一个是普通意义的除，当运算对象有一个是浮点型时，运算的结果即为浮点型，比如 5/2.0 的结果是 2.5，或者 5.0/2 的结果也是 2.5。‘/’的这两种功能，需要特别弄清楚，否则写程序就会出现神奇的 bug。

特别的‘%’号：取余符号在数学中用得比较少，但在计算机编程中却经常用到。‘%’表示两个整数相除后的余数，如 5%2 的结果为 1。需要注意的是‘%’两边的运算对象类型必须

是整型，否则编译会出错，比如 5.0%2 就是错误的写法，编译不通过。

练习：计算以下表达式的值：
（1）-30*3+21/5
（2）-30+3*21/5
（3）30/3*21%5

结果分别是-86，-18，0，跟你的答案一样吗？

有时候，我们需要将数学的表达式转换成 C++表达式，在这个过程需要注意区分，C++表达式中只能用小括号（）进行嵌套，改变表达式的运算优先级，多个小括号可以层层嵌套，但是不能使用中括号和大括号。数学表达式中常见在变量之间省略运算符，而 C++则不允许这样做。

练习，将下列数学表达式转换成 C++表达式

$$\frac{5+b}{\frac{a+6}{b+5}-c\cdot d} \qquad \frac{p\cdot q\cdot (r+1)^2}{(r+1)^2-1}$$

以下是参考答案，跟你写得一样吗？
(5+b)/((a+6)/(b+5)-c*d)
P*q*(r+1)*(r+1)/((r+1)*(r+1)-1)

例题：
输入一个三位自然数，分离出三位数的个位、十位、百位，将它们倒序输出
输入样例：583
输出样例：3 8 5

分析：
假如我们输入的一个三位数存储在变量 a 中，如何在 a 中分离出个位、十位、百位呢？对于任意位数的 a，将其整除 10 后，它的余数就是个位，比如 123%10 的结果就是 3，所以个位数可以写成 a%10。而 123/10 的结果则是 12，即整除 10 以后的结果相当于将该数的个位删除。那么如果求 10 位数呢？123/10=12，12 中的个位即是原数的十位数，所以，十位数可以写成 a/10%10。同理，百位数可以写成 a/10/10，简写成 a/100。我们将求出的各个位数存到变量 ge、shi、bai 中，即：
ge=a%10;
shi=a/10%10;
bai=a/100;

则以上题目的程序很容易就可以实现：

```
#include<iostream>
using namespace std;
int main(){
    int a,ge,shi,bai;
    cin>>a;  //输入一个三位数
    ge=a%10;    //分离个位
    shi=a/10%10;    //分离十位
    bai=a/100;      //分离百位
    cout<<ge<<" "<<shi<<" "<<bai<<endl;
}
```

思考：如何分离一个四位数的个位、十位、百位、千位呢？

关系运算符

关系运算符用来比较两种运算对象的关系，包括相等、不相等、大于、小于、大于等于、小于等于。关系表达式的结果是一种布尔类型，即运算结果只能是 true 或者 false。关系成立结果则为 true，关系不成立结果则为 false。

运算符	运算	运算对象	结果类型
==	等于	简单类型	布尔型
!=	不等于	简单类型	布尔型
<	小于	简单类型	布尔型
>	大于	简单类型	布尔型
<=	小于等于	简单类型	布尔型
>=	大于等于	简单类型	布尔型

观察以下表达式的值：

```
2<3   //结果是 true
345.5<=100   //结果是 false
12 != 10   //结果是 true
```

逻辑运算符

逻辑运算符用来关联两种条件的关系，常见的运算符有逻辑与（&&）、逻辑或（||）和逻辑非（!），逻辑运算符关联的运算对象必须是布尔类型，其运算结果也是一个布尔类型。

逻辑与（&&）：表示两个条件都成立结果才成立。比如：妈妈给小明制定的奖励规则是，小明的语文成绩（yw）大于 90 分并且数学成绩（sx）大于 95（yw>90&&sx>95）才给予奖励。

那么，如果小明的数学考了 100 分而语文考了 89 也是不奖励的。

逻辑或（||）：表示两个条件只要有一个成立结果就成立。比如：公园收费规则为 10 岁以下或者 60 岁以上免费，只要符合两个条件之一就可以享受公园免费规则。

逻辑非（!）：单目运算符，取反操作，就像电灯开关一样，如果开则变成关，关则变成开。

以下是三个逻辑运算符的真值表：

a	b	! a	a && b	a || b
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

练习：根据以下信息写出表达式

1、整数 x 是偶数

2、写出一个数 x 既能被 3 整除又能被 5 整除的表达式

3、判断 y 是闰年的表达式（闰年的判断方法：能够被 4 整除且不能被 100 整除，或者能被 400 整除）

参考答案，注意，以下不是唯一写法：
1、x % 2==0
2、(x % 3==0) && (x % 5==0)
3、(y % 4==0) && (y % 100！=0) || (y% 400==0)

好了，在这里我们介绍了三类运算符，需要注意的是三类运算符不是单独分离的，而是可以互相结合组成复杂的表达式。算式表达式的结果可以成为关系运算符的运算对象，而关系表达式的运算结果可以成为逻辑运算符的运算对象。以上面第 2 题为例，x%3 是算式表达式，在 x%3==0 中做关系运算符==的运算对象，而 x%3==0 是关系表达式，在(x % 3==0) && (x % 5==0)中做逻辑运算符&&的运算对象。根据这一规则，可以组成更复杂的表达式，如上面第 3 题。

表达式的注意事项：

1、整型、实型、字符型数据间可以混合运算。在这种情况下，需要将不一致的数据类型转换成一致的数据类型，然后进行运算。为了保证运算精度，系统在运算时的转换规则是将存储长度较短的运算对象转成存储长度较长的类型，然后再进行处理。这种转换是系统自动进行的。如：3+4.5*2-7 返回的结果为浮点型

2、在求复合表达式的值需要考虑运算符的优先级。如果不明确运算符的优先级，可以使用括号将表达式的某个局部括起来使其得到优先运算。以下是各个运算符的优先级：

优先级	运算符	说明	结合性
1	()	括号	自左向右
2	!	逻辑非/按位取反	自右向左
3	* / %	乘法/除法/取余	自左向右
4	+ -	加号/减号	
5	< <=	小于/小于等于	
	> >=	大于/大于等于	
6	== !=	等于/不等于	
7	&&	与运算	
8	||	或运算	
9	=	赋值	自右向左

在上表中可以观察到算是运算符的优先级高于关系运算符，关系运算符的优先级高于逻辑运算符。

思考，求闰年的表达式，如果将式子的括号去掉，写成以下样子，结果会一样吗？

y % 4==0 && y % 100！=0 || y% 400==0

==========网站练习==========

追及问题

【问题描述】

甲、乙两人环绕周长是 300 米的跑道跑步，甲每秒跑 6 米，乙每秒跑 4 米，如果两人是从同一地点同向出发，假设两人永远保持体力可以一直均速跑下去，我们知道，甲可以第一次、第二次、……、第 n 次追上乙。

求第 n 次相遇的时间(n<=10000)，单位为秒。

【输入格式】

一个整数 n(n<=10000)。

【输出格式】

求第 n 次相遇的时间，单位为秒。

【输入样例】

1

【输出样例】

150

【分析】

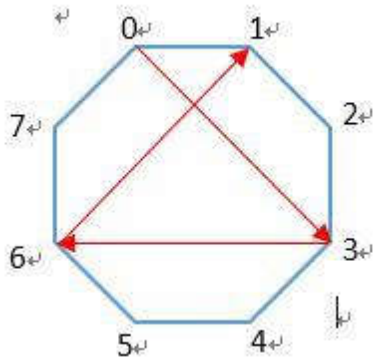
小学奥数题，题目告诉我们第一次相遇时间是 150 秒，后面每 150 秒一个周期相遇，所以第 n 次相遇的时间为 n*150。

【代码】

```
#include<iostream>
using namespace std;
int main(){
    int n;
    cin>>n;
    cout<<n*150;
}
```

青蛙跳荷叶

【问题描述】
在一个荷塘里，有 8 个荷叶按照正八边形的形状生长着，我们给荷叶依次编号为 0——7。在荷叶 0 的位置上，有一只青蛙，青蛙可以每次沿顺时针方向跳过 a 个荷叶而停在下一个荷叶上，请问青蛙经过 k 次跳跃后，它会停在哪朵荷叶上。
【输入格式】
两个整数，a 和 k，其中 0<a<=5，0<K<=100。
【输出格式】
一个整数，表示青蛙最终所在荷叶的编号。
【输入样例】
2 3
【输出样例】
1
【注释】



【分析】
此题有点思维难度。不妨假设青蛙在一条线上跳跃，则第 k 次跳跃所在的距离为（a+1）*k，现在把线变成一个环，则所在位置在 0——7 的周期上，即（a+1）*k%8。考点是对取余符号的认识。
【代码】

```
#include<iostream>
using namespace std;
int main(){
    int a,k;
    cin>>a>>k;
```

```
6.      cout<<(a+1)*k%8;
7.  }
```

小矮人

【问题描述】
最初出现“七个小矮人”的是德国著名童话集 《格林 童话》之中的《白雪公主》 。原文讲述了一个可爱 美丽的公主因为被恶毒后母嫉妒其美貌而被迫逃 到森林，在缘分安排下偶遇善良的七个小矮人。最后在他们帮助下，破解了后母的诅咒，找到了王子 的真爱的故事。七个小矮人的姓名分别是： 万事通、害羞鬼、瞌睡虫、喷嚏精、开心果、迷糊鬼、爱生气。白雪公主经常为这七个小矮人讲故事。白雪公 主还为这七个小矮人安排了学号，学号分别是 1 至 7 号。有一次，白雪公主又邀请七个小矮人来听她讲故事，但是只来了六个小矮人，那么缺席的那个小矮人是谁呢？
【输入格式】
一行，6 个学号，空格分开，表示来听故事的六个小矮人的学号。
【输出格式】
没来听故事的小矮人的学号。
【输入样例 1】
3 5 2 1 7 4
【输入样例 2】
7 3 2 4 1 6
【输出样例 1】
6
【输出样例 2】
5
【分析】
可以根据数学方法得，1 至 7 的总和为 28，将 28 减去已出现的 6 个数，则剩下的是最后一个没出现的数。涉及多个变量的读入。
【代码】

```
1.  #include<iostream>
2.  using namespace std;
3.  int main(){
4.      int a1,a2,a3,a4,a5,a6,sum;
5.      cin>>a1>>a2>>a3>>a4>>a5>>a6;
6.      cout<<28-a1-a2-a3-a4-a5-a6;
7.  }
```

温度转换

【问题描述】
编一程序，将摄氏温度换为华氏温度。公式为：f=9/5*c+32。其中 f 为华氏温度，c 是摄氏温度。

【输入格式】
输入一行，只有一个整数 c。
【输出格式】
输出只有一行，包括 1 个实数。（保留两位小数）
【输入样例】
50
【输出样例】
122.00
【注释】
如何在 C++输出中保留两位小数。以下为演示代码：

```
#include<iostream>
#include<iomanip>   //setprecision 和 fixed 函数所需的头文件
using namespace std;
int main(){
    double a;
    cin>>a;
    cout<<setprecision(2)<<fixed<<a;  //输出实数变量 a 且保留 2 位小数
}
```

【分析】
本题主要考验对浮点型数据的处理。
【代码】

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
    double f,c;
    cin>>c;
    f=9.0/5*c+32;   //思考，这里能否写成 9/5*c+32
    cout<<setprecision(2)<<fixed<<f;   //保留两位小数
}
```

小明买箱子

【问题描述】
放暑假了，小明准备去旅游，当然啦，旅行之前要整理行李。于是，小明紧锣密鼓地准备买行李箱装行李（网购形式），但是，他看到卖家都是以英寸来标示行李箱的大小，这让小明很头痛。在他知道 1 英寸=2.54 厘米后，他决定编写一个程序，将他需要的尺寸（厘米）转化成英寸，以便在网上选到合适的箱子。
【输入格式】
一个整数，表示多少厘米，数据小于等于 1000000000。
【输出格式】

一个实数，保留小数点后两位，表示转化后的英寸。
【输入样例】
100
【输出样例】
39.37
【分析】
本题与上题相似，考验对公式的逆推。
【代码】

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
    double cm,inch;
    cin>>cm;
    inch=cm/2.54;
    cout<<setprecision(2)<<fixed<<inch;
}
```

大象喝水

【问题描述】
一只大象口渴了，要喝 20 升水才能解渴，但现在只有一个深 h 厘米，底面半径为 r 厘米的小圆桶(h 和 r 都是整数)。问大象至少要喝多少桶水才会解渴。
【输入格式】
输入有一行：包行两个整数，以一个空格分开，分别表示小圆桶的深 h 和底面半径 r，单位都是厘米。
【输出格式】
输出一行，包含一个整数，表示大象至少要喝水的桶数。
【输入样例】
23 11
【输出样例】
3
【注释】
如果一个圆桶的深为 h 厘米，底面半径为 r 厘米，那么它最多能装 Pi * r * r * h 立方厘米的水。(设 Pi=3.14159)
1 升 = 1000 毫升
1 毫升 = 1 立方厘米
【分析】
本题主要涉及到实型数据与整型的转换。
【代码】

```
#include<iostream>
using namespace std;
```

```
3.  int main(){
4.      double h,r;
5.      int ans;
6.      cin>>h>>r;
7.      ans=20*1000/(r*r*3.14159*h);    //将实型数据赋值给整型，去除小数
8.      cout<<ans+1;
9.  }
```