第二节课课后习题

【1047：判断能否被 3，5，7 整除】

给定一个整数，判断它能否被 3，5，7 整除，并输出以下信息：

1、能同时被 3，5，7 整除（直接输出 3 5 7，每个数中间一个空格）；

2、只能被其中两个数整除（输出两个数，小的在前，大的在后。例如：3 5 或者 3 7 或者 5 7，中间用空格分隔）；

3、只能被其中一个数整除（输出这个除数）；

4、不能被任何数整除，输出小写字符‘n’，不包括单引号。

【1055：判断闰年】

1. 公元年分除以 4 可整除但除以 100 不可整除，为闰年。
2. 公元年分除以 400 可整除而不能被 3200 整除的，为闰年。

闰年计算方法：

```
if ((y % 4 == 0 && y % 100 != 0) || (y % 400 == 0 && y % 3200 != 0))
```

NOI(全国青少年信息学奥林匹克竞赛)

第三课

循环结构

有的时候，可能需要多次执行同一块代码。一般情况下，语句是顺序执行的：函数中的第一个语句先执行，接着是第二个语句，依此类推。

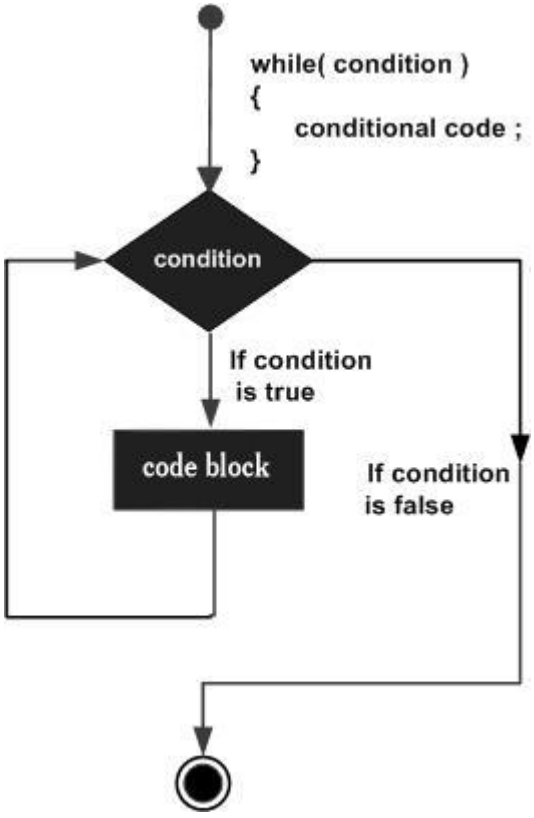
但循环语句允许我们多次执行一个语句或语句组。

while 循环语法

```
while(condition)
{
   conditional code; // 循环主体
}
```

while 循环的执行过程：

1. 首先判断 condition（条件表达式）。如果为真，则执行循环主体。如果为假，则不执行循环主体，结束整个 while 语句。
2. 执行完循环主体后转到第 1 步。



例子

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

```
int main ()
{
   // 局部变量声明
   int a = 10;
   // while 循环执行
   while( a < 20 )
   {
       cout << "a 的值：" << a << endl;
       a++;
   }
   return 0;
}
```

练习

1. 求 s=1+2+3+…+n，当 s 的值超过 1000 时，n 等于多少
2. 设计一个程序，可以不断的输入正整数（数字最大不超过 1000），并且实时统计出所有已经输入的数字中的最大值和最小值。
   提示：cin >> a;的返回值是 istream& operator>>(istream&, T &);的返回值。其返回值类型为 istream&类型，大多数情况下其返回值为 cin 本身（非 0 值），只有当遇到 EOF 输入时，返回值为 0。Windows 下输入 EOF 是 ctrl+z。 cin.eof()也可以用来判断是否输入 EOF

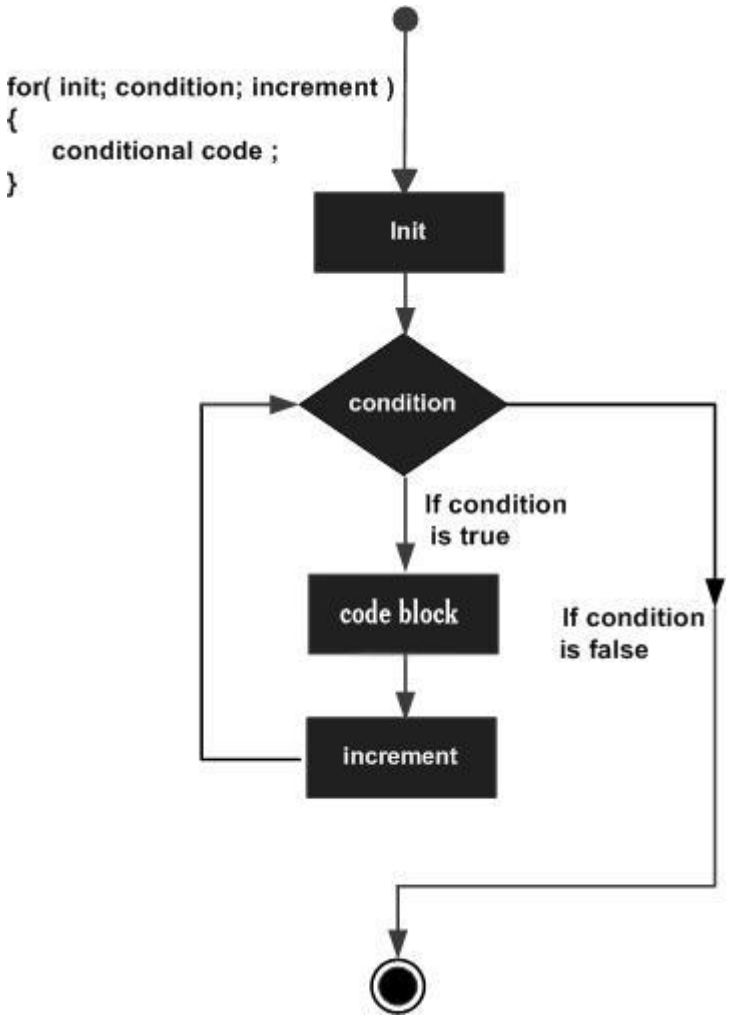
for 循环的语法：

```
for ( init; condition; increment )
{
   conditional code; //循环主体;
}
```

下面是 for 循环的执行过程：

1. **init** （变量初始化表达式）。会首先被执行，且只会执行一次。一般会让控制变量获得一个初始值。备注：也可以不在这里写任何语句，只要有一个分号出现即可。
2. 接下来，会判断 **condition （条件表达式）**。如果为真，则执行循环主体。如果为假，则不执行循环主体，结束整个 for 语句。

3. 在执行完 for 循环主体后，控制流会跳回上面的 increment （增量表达式）。该语句允许您更新控制变量。备注：该语句也可以留空，只要在条件后有一个分号出现即可。
4. 转到第 2 步。



例子

```
#include <iostream>
using namespace std;
int main () {
   // for 循环执行
   for( int a = 10; a < 20; a = a + 1 )
   {
      cout << "a 的值：" << a << endl;
   }
   return 0;
}
```

练习

1. 输出 1-100 之间所有的偶数。
2. 计算 1-100 之间所有的偶数的和。

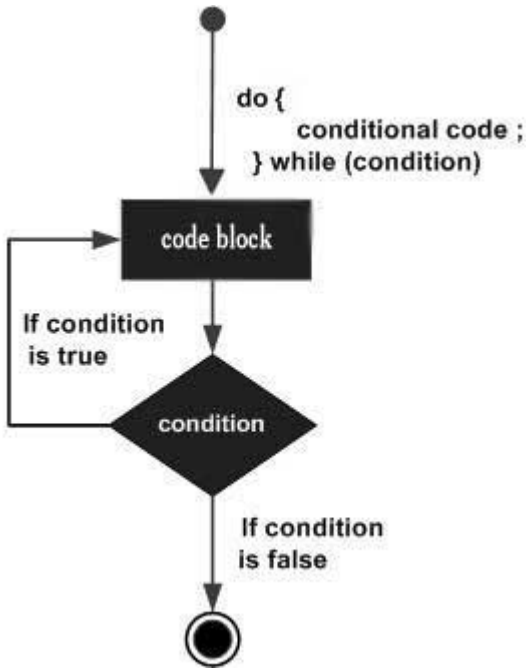
do...while 语法

```
do
{
   conditional code; //循环主体
} while( condition );
```

对比 while 循环，do..while 中的循环主体**至少会执行一遍**。

下面是 do..while 循环的执行过程：

1. 执行一遍循环体
2. 会判断 condition （**条件表达式**）。如果为真，则跳到第 1 步去执行循环主体。如果为假，结束整个 for 语句。



例子

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

int main ()
{
   // 局部变量声明
   int a = 10;
   // do 循环执行
   do
   {
       cout << "a 的值：" << a << endl;
       a = a + 1;
   }while( a < 20 );

   return 0;
}
```

练习

1. 求 s=1+2+3+…+n，当 s 的值超过 1000 时，n 等于多少 （do..while 实现）

嵌套循环

下面的输出语句执行了多少遍？

```
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 10; j++) {
        cout << “s” << endl;
    }
}
```

课后练习

1059：求平均年龄

http://ybt.ssoier.cn:8088/problem_show.php?pid=1059

1071：菲波那契数

http://ybt.ssoier.cn:8088/problem_show.php?pid=1071

1088：分离整数的各个数

http://ybt.ssoier.cn:8088/problem_show.php?pid=1088

1095：数 1 的个数

http://ybt.ssoier.cn:8088/problem_show.php?pid=1095

数组

它可以存储一个固定大小的相同类型元素的顺序集合。数组往往被认为是一系列相同类型的变量。

声明一维数组

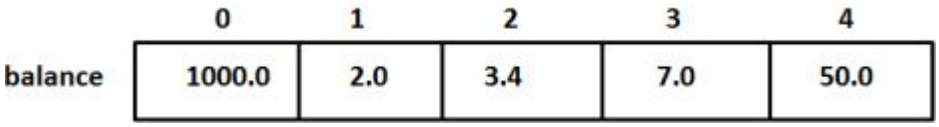
需要指定元素的类型和元素的数量，如下所示：

type arrayName [arraySize]; //注意 arraySize 不可以是变量（dev c++的编译器比较新，用变量不会报错，但依然是非法的写法）

初始化数组

可以逐个初始化数组，也可以使用一个初始化语句，如下所示：

double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};



数组元素的引用

数组元素可以通过数组名加下标进行引用。下标是放在方括号内的，可以是任意值为整形的表达式。例如：

balance[9]
balance[i] //i 是小于等于 4 的整形变量
balance[i+j] // i 和 j 都是整形变量，i+j 小于等于 4
对于数组元素，可以进行赋值和运算，就像普通的变量一样。

数组中某个元素进行赋值

balance[4] = 50.0;

数组越界

在引用数组元素时，要注意下标不要超过数组元素的最大个数。
int a = balance[5]; //语法正确，能通过编译，但是运行时出错。

例子

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

int main ()
{
   int n[ 10 ]; // n 是一个包含 10 个整数的数组

   // 初始化数组元素
   for ( int i = 0; i < 10; i++ )
   {
      n[ i ] = i + 100; // 设置元素 i 为 i + 100
   }
```

```
   // 输出数组中每个元素的值
   for ( int j = 0; j < 10; j++ )
   {
      cout << n[ j ] << endl;
   }

   return 0;
}
```

练习

1. 输入 n 个数字，按照输入时的逆序把 n 个数字打印出来。提示：while 循环
2. 已知 int 数组 a 包含 10 个元素，请将数组中的第一个元素移到末尾，其余元素依次向前移动一个位置。提示：利用一个临时变量，考察数组的赋值和引用

二维数组

多维数组最简单的形式是二维数组。一个二维数组，在本质上，是一个一维数组的数组。声明一个 x 行 y 列的二维整型数组，形式如下：

```
type arrayName [ x ][ y ];
```

一个二维数组可以被认为是一个带有 x 行和 y 列的表格。下面是一个二维数组，包含 3 行和 4 列：

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

初始化二维数组

多维数组可以通过在括号内为每行指定值来进行初始化。下面是一个带有 3 行 4 列的数组。

```
int a[3][4] = {
 {0, 1, 2, 3} ,   /*  初始化索引号为 0 的行 */
 {4, 5, 6, 7} ,   /*  初始化索引号为 1 的行 */
 {8, 9, 10, 11}   /*  初始化索引号为 2 的行 */
};
```

内部嵌套的括号是可选的，下面的初始化与上面是等同的：

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11}; //不推荐的写法
```

访问二维数组元素

二维数组中的元素是通过使用下标（即数组的行索引和列索引）来访问的。例如：

```
int val = a[2][3];
```

上面的语句将获取数组中第 3 行第 4 个元素。

练习

1. 已知一个二维数组 a[5][2]，使用嵌套循环来计算每一行和每一列的和。

课后练习

1102：与指定数字相同的数的个数

http://ybt.ssoier.cn:8088/problem_show.php?pid=1102

1124：矩阵加法

http://ybt.ssoier.cn:8088/problem_show.php?pid=1124