

C++初学者必看的50个建议

- 7.学class并不难，template、STL、generic programming也不过如此——难的是长期坚持实践和不遗余力的博览群书；
- 8.如果不是天才的话，想学编程就不要想玩游戏——你以为你做到了，其实你的C++水平并没有和你通关的能力一起变高——其实可以时刻记住：学C++是为了编游戏的；
- 9.看Visual C++的书，是学不了C++语言的；

C++初学者必看的50个建议

- 10.浮躁的人容易说：XX语言不行了，应该学YY；——是你自己不行了吧！？
- 11.浮躁的人容易问：我到底该学什么；——别问，学就对了；
- 12.浮躁的人容易问：XX有钱途吗；——建议你去抢银行；
- 13.浮躁的人容易说：我要中文版！我英文不行！——不行？学呀！

第2篇

面向过程的程序设计

第3章 程序设计初步

第4章 函数

第5章 数组

第6章 指针

第7章 自定义数据类型

第3章 程序设计初步

- 3.1 面向过程的程序设计和算法
- 3.2 C++程序结构和语句
- 3.3 赋值语句
- 3.4 C++的输入与输出
- 3.5 编写顺序结构的程序
- 3.6 关系运算和逻辑运算
- 3.7 选择结构和条件语句
- 3.8 循环结构和循环语句

数据对象

- **数据**（**data**）是计算机处理的对象，是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并由计算机程序处理的信息的总称。
- **数据元素**（**Data Element**）是数据的基本单位，在计算机程序中通常作为一个整体考虑。
- **数据对象**（**Data Object**）是性质相同的数据元素的集合，是数据的一个子集。数据对象可以是有限的，也可以是无限的。

数据结构

- 数据结构在计算机科学界至今没有标准的定义。个人根据各自的理解而有不同的表述方法：
- 数据结构是指相互之间存在着一种或多种关系的数据元素的集合和该集合中数据元素之间的关系组成。
- Sartaj Sahni 在他的《数据结构、算法与应用》一书中称：“数据结构是数据对象，以及存在于该对象的实例和组成实例的数据元素之间的各种联系。这些联系可以通过定义和相关的函数来给出。”他将数据的对象（data object）定义为“一个数据对象是实例或值的集合”。
- Clifford A. Shaffer 在《数据结构与算法分析》一书中的定义是：“数据结构是 ADT（抽象数据类型 Abstract Data Type）的物理实现。”
- Lobert L. Kruse 在《数据结构与程序设计》一书中，将一个数据结构的设置过程抽象成数据层和象层，讨论一个数据结构的抽象层是指数据的抽象结构，讨论一个数据结构的象层是指数据的物理实现。将逻辑层的数据结构抽象成数据层和象层，讨论一个数据结构的抽象层是指数据的抽象结构，讨论一个数据结构的象层是指数据的物理实现。

数据结构

- **数据结构**（data structure）指同一数据元素类中各数据元素之间存在的关系，即数据的组织形式，是一系列性质相同的数据组织成一定的**逻辑结构**（包括在计算机中的**存储结构**）并带有自身的一系列**操作**（实现）。
- 一般认为，数据结构是由数据元素依据某种逻辑联系组织起来的。对数据元素间逻辑关系的描述称为数据的逻辑结构；数据必须在计算机内存储，数据的存储结构（物理结构）是数据结构的实现形式，是其在计算机内的表示；此外讨论一个数据结构必须同时讨论在该类数据上执行的运算才有意义。
- 处理同一类问题，如果数据结构不同，算法也会不同。

数据类型

- **数据类型**（**data type**）是一组性质相同的具有一定范围的值集以及定义于这个值集上的一组操作。
- 数据类型的本质是**数据组织**和其（**允许的**）**操作的捆绑性**。
- 数据类型是数据结构在一定编程语言中的描述形式。

抽象数据类型

- 抽象数据类型(Abstract Data Type 简称ADT)是指一个数学模型以及定义在此数学模型上的一组操作。实际上就是对该数据结构的定义。因为它定义了一个数据的逻辑结构以及在此结构上的一组算法。数据结构是抽象数据类型的物理实现。
- 在某种意义上讲,抽象数据类型和数据类型实质上是一个概念。抽象数据类型是基于已有数据类型而组合生成的复合数据类型。自定义的数据类型是比基本数据类型更高级(抽象)的数据类型。
- 抽象数据类型是实体的抽象,是描述问题的模型,也可理解为数据类型的进一步抽象。它是独立于具体实现的。抽象数据类型将数据和操作封装在一起,使得用户程序只能通过ADT里定义的某些操作来访问其中的数据,从而实现了信息隐藏。其在C++中的实现即是类。

- 数据：是能输入到计算机中并能被计算机程序处理的符号的总称。
- 数据元素：是数据的基本单位，它在计算机处理和程序设计中通常作为一个整体进行考虑和处理。一个数据元素可由若干数据项组成。
- 数据对象：是具有相同特征的数据元素的集合，是数据的一个子集。
- 数据结构：是数据元素的组织形式，或数据元素相互之间存在一种或多种特定关系的集合。
- 数据的逻辑结构：是带有结构的数据元素的集合，它是指数据元素之间的相互关系，即数据的组织形式。有时被简称为数据结构。
- 数据的存储结构：是数据的逻辑结构在计算机内存中的存储方式，又称物理结构。
- 数据的运算：即对数据施加的操作。数据的运算定义在数据的逻辑结构上，每种逻辑结构都有一个运算的集合。
- 数据类型：是一组具有相同性质的操作对象以及该组操作对象上的运算方法的集合。
- 抽象数据类型：是指一个数学模型以及在该模型上定义的一套运算规则的集合。

3.1 面向过程的程序设计和算法

语言必须具有自建数据结构的能力。

数据结构（data structure）指数据的组织形式，是一系列性质相同的**数据组织**成一定的**逻辑结构**（包括在计算机中的**存储结构**）并带有自身的一系列操作。

数据类型（data type）是一组性质相同的具有一定范围的值集以及定义于这个值集上的一组操作。数据类型的本质是**数据组织和其操作的捆绑性**。数据类型是数据结构在一定编程语言中的描述形式。

抽象数据类型（abstract data type）是基于已有数据类型而组合生成的复合数据结构。描述数据的组织和相关的操作，反映问题的抽象模型，是数据结构的真正体现。

- 选择了数据结构，算法也随之确定，是数据而不是算法是系统构造的关键因素。这种洞见导致了许多种软件设计方法和程序设计语言的出现，面向对象的程序设计语言就是其中之一。

计算机解决一个具体问题，大致需要经过下列几个步骤：首先要从具体问题中抽象出一个适当的数学模型，然后设计一个解此数学模型的算法（Algorithm），最后编出程序、进行测试、调整直至得到最终解答。寻求数学模型的实质是分析问题，从中提取操作的对象，并找出这些操作对象之间含有的关系，然后用数学的语言加以描述。计算机算法与数据的结构密切相关，算法无不依附于具体的数据结构，数据结构直接关系到算法的选择和效率。运算是由计算机来完成，这就要设计相应的插入、删除和修改的算法。也就是说，数据结构还需要给出每种结构类型所定义的各种运算的算法。

结构化程序设计

- 结构化程序设计由迪杰斯特拉(E.W.dijkstra)在1969年提出，是以**模块化设计**为中心。
- 按照结构化程序设计的观点，任何算法功能都可以通过由程序模块组成的三种基本程序结构的组合：**顺序结构、选择结构和循环结构**来实现。
- 结构化程序设计的基本思想是采用“**自顶向下，逐步求精**”的程序设计方法和“**单入口单出口**”的控制结构。

抽象 (Abstract)

- 抽象分行为抽象和数据抽象两种。
- **行为抽象**：通俗地说便是将一个行为序列归并(抽象)为一个行为的过程。例如：将取碗筷、盛饭、盛菜，扒一口饭、夹一筷菜、再扒一口饭、再夹一筷菜的若干重复，然后放下碗筷的过程归并为吃饭。
- **数据抽象**：通俗地说，就是将事物归类，或者说，将事物看成是一定型号、规格的数据，然后将性质接近的数据归纳(抽象)为一类。例如：将圆、三角形、长方形归为形状类。

面向过程的程序设计

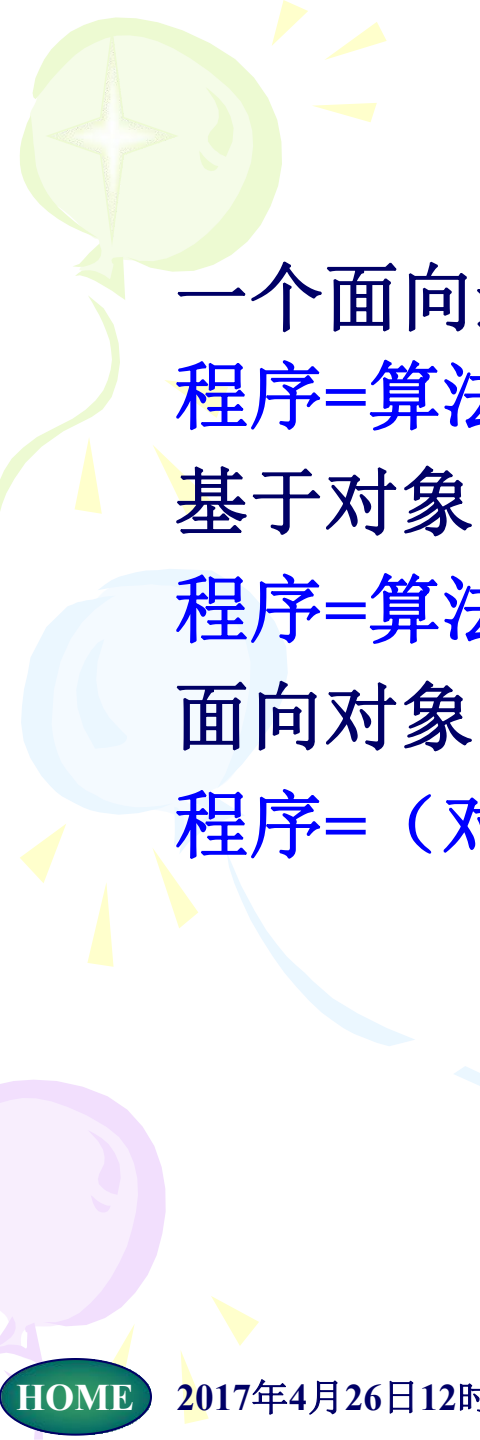
- 面向过程编程（过程化编程）将问题通过功能分解，各个击破的编程方法是一种以行为抽象为主的编程思想。
- 面向对象编程将问题通过实体分析，分层分类地实现抽象数据类型，从而进行简单应用编程是一种以数据抽象为主的编程思想。
- 所谓面向过程的程序设计是：用结构化编程语句来编写程序。它把一个复杂的程序分解成若干个较小的过程，每个过程都可以单独地设计、修改、调试。其程序流程完全由程序员控制，用户只能按照程序员设计好的程序处理问题。

算法

- 算法(algorithm)是对特定问题求解步骤的一种描述,是指令的有限序列。描述算法需要一种语言,可以是自然语言、数学语言或者是某种计算机语言。
- 算法是求解特定问题的一组有限的操作序列。
- 算法离不开数据结构。
- 用计算机语言描述算法,就是计算机程序。
- 计算机算法可分为两大类别:
- 数值算法和非数值算法

算法

- 算法(algorithm)是对特定问题求解步骤的一种描述,是指令的有限序列。描述算法需要一种语言,可以是自然语言、数学语言或者是某种计算机语言。
- 一个算法一般具有下列 5 个重要特性: 葛
- (1) 输入: 一个算法应该有一个或多个输入。 葛
- (2) 有穷性: 一个算法必须在执行有穷步骤之后正常结束,而不能形成无穷循环。
- (3) 确定性: 算法中的每一条指令必须有确切的含义,不能产生多义性。 葛
- (4) 可行性: 算法中的每一条指令必须是切实可执行的,即原则上可以通过已经实现的基本运算执行有限次来实现。 葛
- (5) 输出: 一个算法应该有零个或多个输出,这些输出是同输入有某个特定关系的量。



一个面向过程的程序应包括：

程序=算法+数据结构

基于对象的程序，可表示成：

程序=算法+抽象数据类型

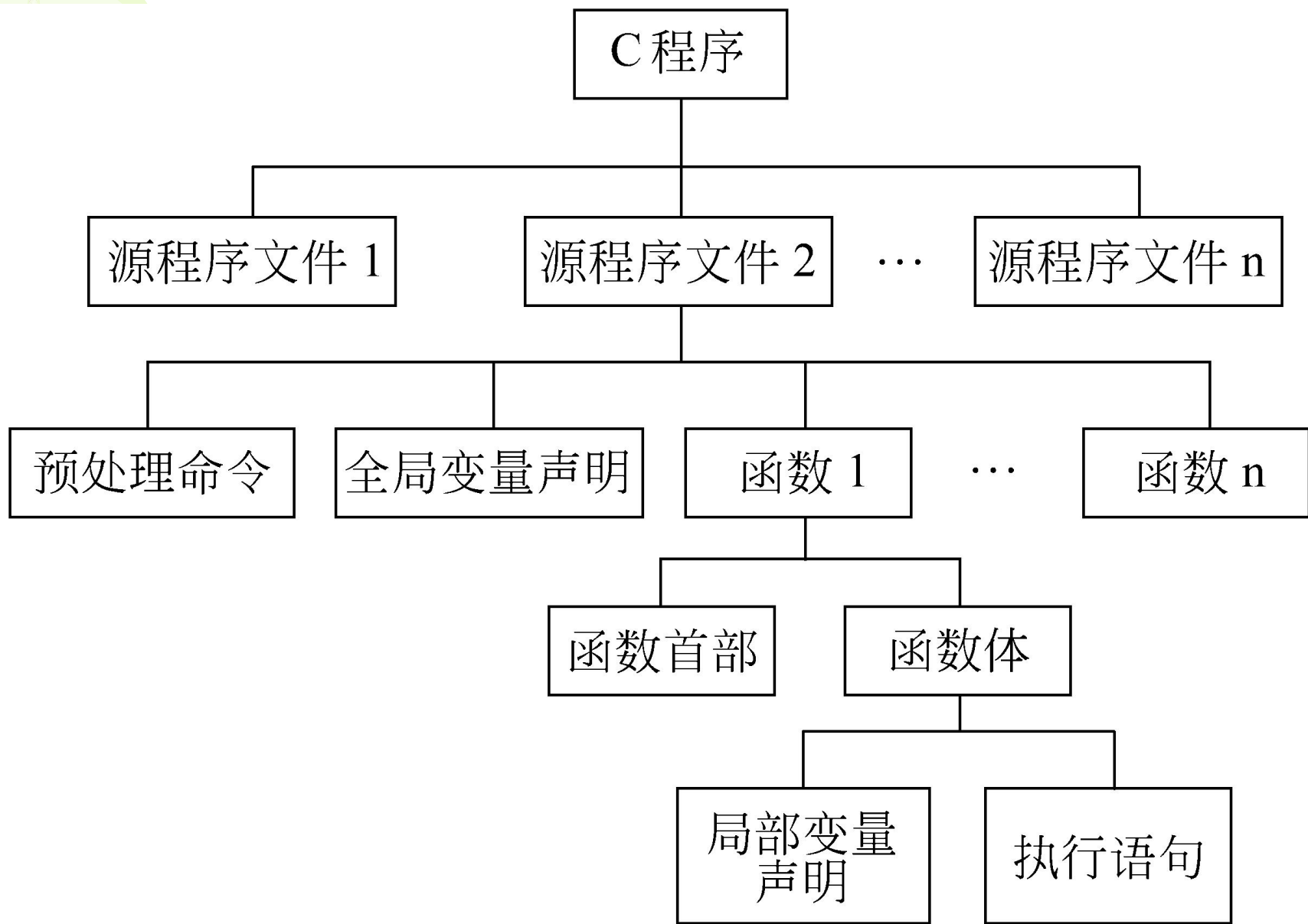
面向对象的程序，可表示成：

程序=（对象，...对象）+消息

3.2 C++程序和语句

一个完整的C++程序：

```
#include <iostream> //预处理命令
using namespace std; //函数外的声明部分
int a=3; //全局变量
int main( ) //函数首部
{
    float b; //局部变量
    b=4.5; //执行语句
    cout<<a<<b; //执行语句
    return 0; //执行语句
}
```



语句 (statement)

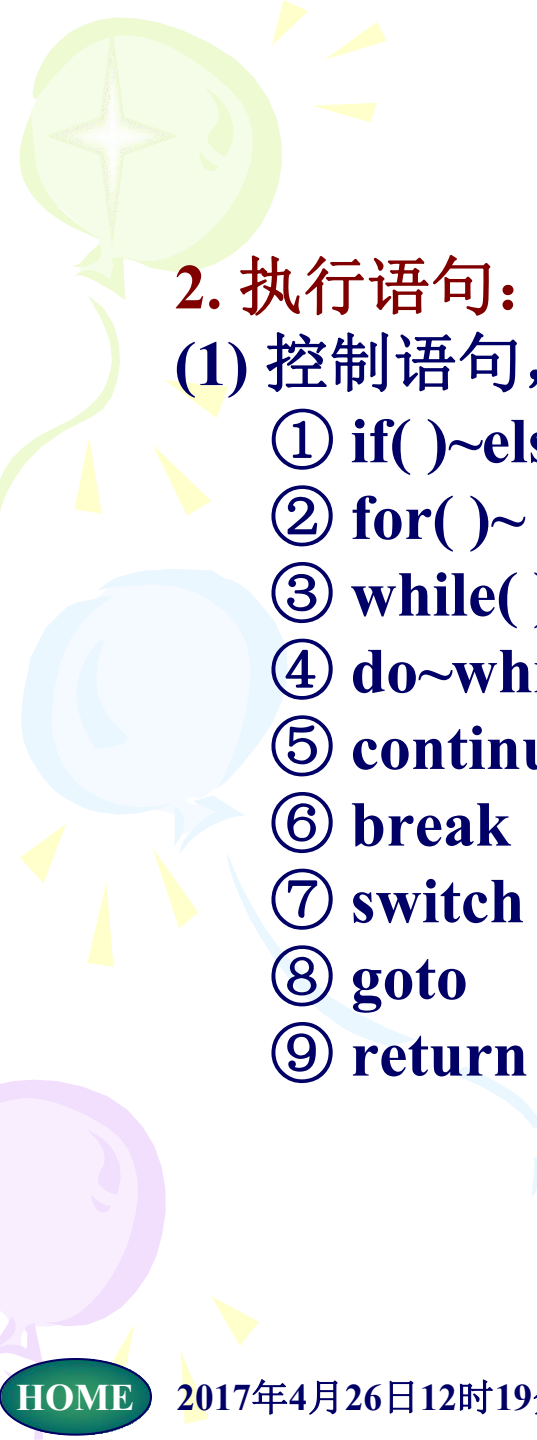
语句是C++程序中最小独立单位。

1. 声明语句

如 `int a,b;`

声明语句可以出现在函数中的任何地方，也可以放在函数之外。

这样可以很方便地实现变量的局部化（变量的作用范围从声明语句开始到本函数或本块结束）。



2. 执行语句：通知计算机完成一定的操作。包括：

(1) 控制语句，完成一定的控制功能。

- ① **if()~else~** (条件语句)
- ② **for()~** (循环语句)
- ③ **while()~** (循环语句)
- ④ **do~while()** (循环语句)
- ⑤ **continue** (结束本次循环语句)
- ⑥ **break** (中止循环语句)
- ⑦ **switch** (多分支选择语句)
- ⑧ **goto** (转向语句)
- ⑨ **return** (从函数返回语句)

(2) 函数和流对象调用语句

函数调用语句由一次函数调用加一个分号构成一个语句，例如

```
sort(x,y,z);           //sort函数有3个参数  
cout<<x<<endl;       //流对象调用语句
```

(3) 表达式语句

由一个表达式加一个分号构成一个语句。

最典型的是：由赋值表达式构成一个赋值语句。

```
i=i+1                 //是一个赋值表达式  
i=i+1;               //是一个赋值语句
```

3. 空语句

;
即只有一个分号的语句。有时语法上需要一个语句，但逻辑上并不需要，此时应该使用空语句。使用空语句时应该加上注释，表明该语句是有意省略的。

```
while ( *string++ = *inBuf++ )
```

```
; // 空语句
```

4. 复合语句

可以用 { } 把一些语句括起来成为复合语句。如：

```
{  
    z=x+y;  
    if(z>100) z=z-100;  
    cout<<z;  
}
```

3.3 赋值语句

赋值表达式与赋值语句

赋值语句是由赋值表达式加上一个分号构成。

在C++中，赋值表达式可以包括在其他表达式之中，例如

```
if((a=b)>0) cout<<"a>0"<<endl;
```

不能写成

```
if((a=b;)>0) cout<<"a>0"<<endl;
```

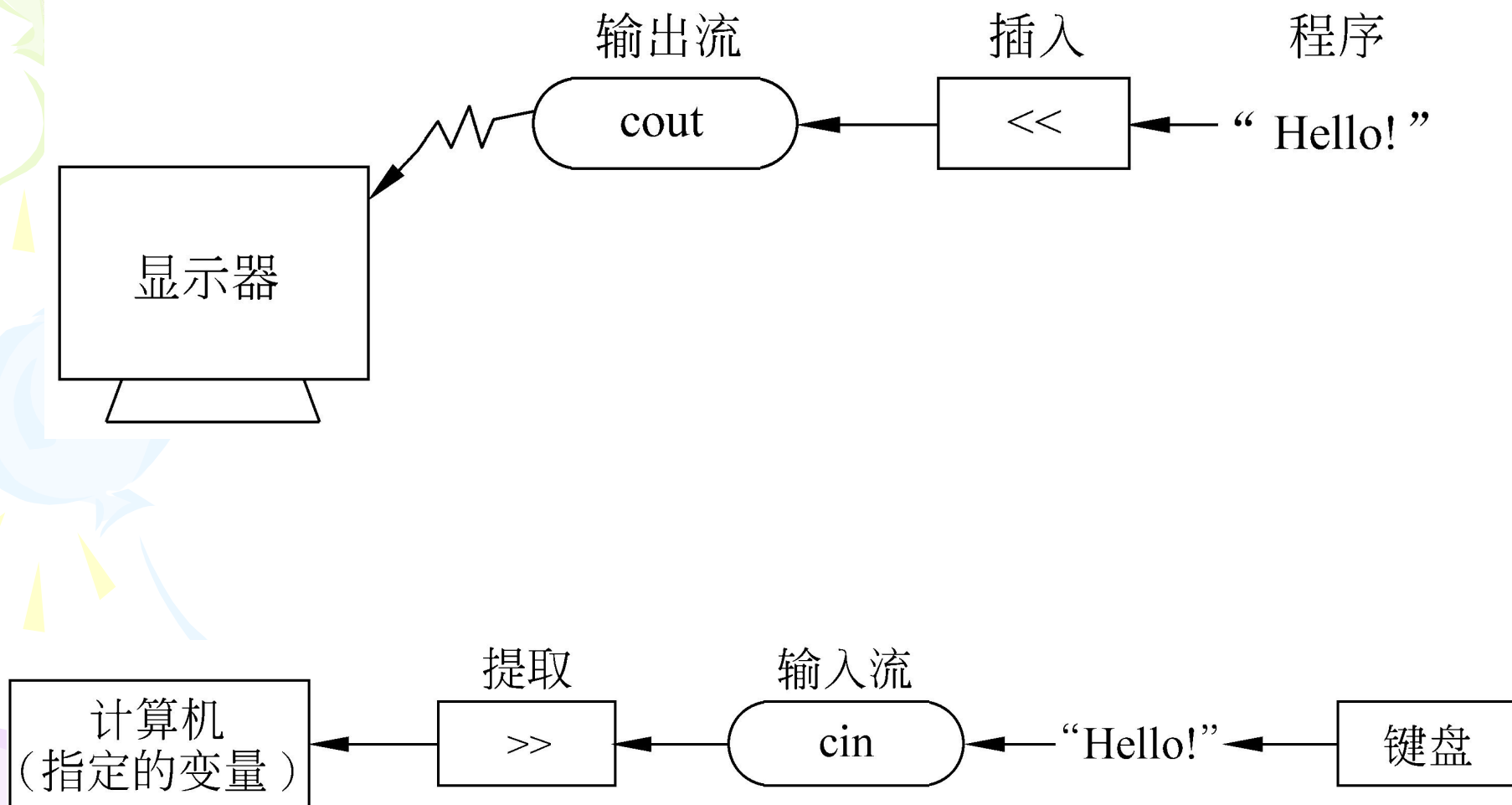
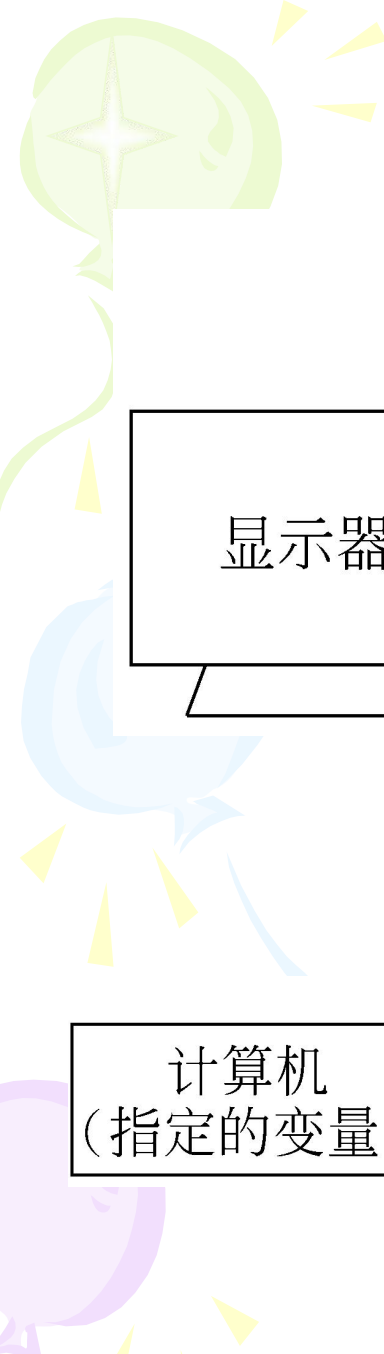
3.4 C++的输入与输出

输入和输出并不是C++语言中的正式组成成分。C和C++本身都没有为输入和输出提供专门的语句结构。输入输出不是由C++本身定义的，而是在编译系统提供的I/O库中定义的。

C++的输出和输入是用“流”(stream)的方式实现的。

在程序中使用cin、cout和流运算符，必须包含头文件iostream:

```
#include <iostream>
```



*3.4.1 输入流与输出流的基本操作

cout语句的一般格式为

cout<<表达式1<<表达式2<<.....<<表达式n;

在用**cout**输出时，用户不必通知计算机按何种类型输出，系统会自动判别输出数据的类型，使输出的数据按相应的类型输出。

不能用一个插入运算符“<<”插入多个输出项：

cout<<a,b,c; //错误，不能一次插入多项

cout<< a<< “ ” <<b << “ ” <<c; //正确



cin语句的一般格式为

cin>>变量1>>变量2>>.....>>变量n;

在用**cin**输入时，系统也会根据变量的类型从输入流中提取相应长度的字节。

不能用**cin**语句把空格字符和回车换行符作为字符输入给字符变量，它们将被跳过。

在组织输入流数据时，要仔细分析**cin**语句中变量的类型，按照相应的格式输入，否则容易出错。

*3.4.2 在输入流与输出流中使用控制符

C++提供了在输入输出流中使用的控制符，以满足在输入输出时的一些特殊的要求，如在输出实数时规定字段宽度，只保留两位小数，数据向左或向右对齐等。

如果使用了控制符，在中还要包含**`iomanip`**头文件。

例3.1 各行小数点对齐

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{
    double a=123.456,b=3.14159,c=-3214.67;
    cout<<setiosflags(ios:: fixed)<<setiosflags(ios:: right)
    <<setprecision(2);
    cout<<setw(10)<<a<<endl;
    cout<<setw(10)<<b<<endl;
    cout<<setw(10)<<c<<endl;
    return 0;
}
```

3.4.3 用getchar和putchar 进行字符输入输出

C++保留了C语言中用于输入和输出单个字符
getchar函数和putchar的函数。

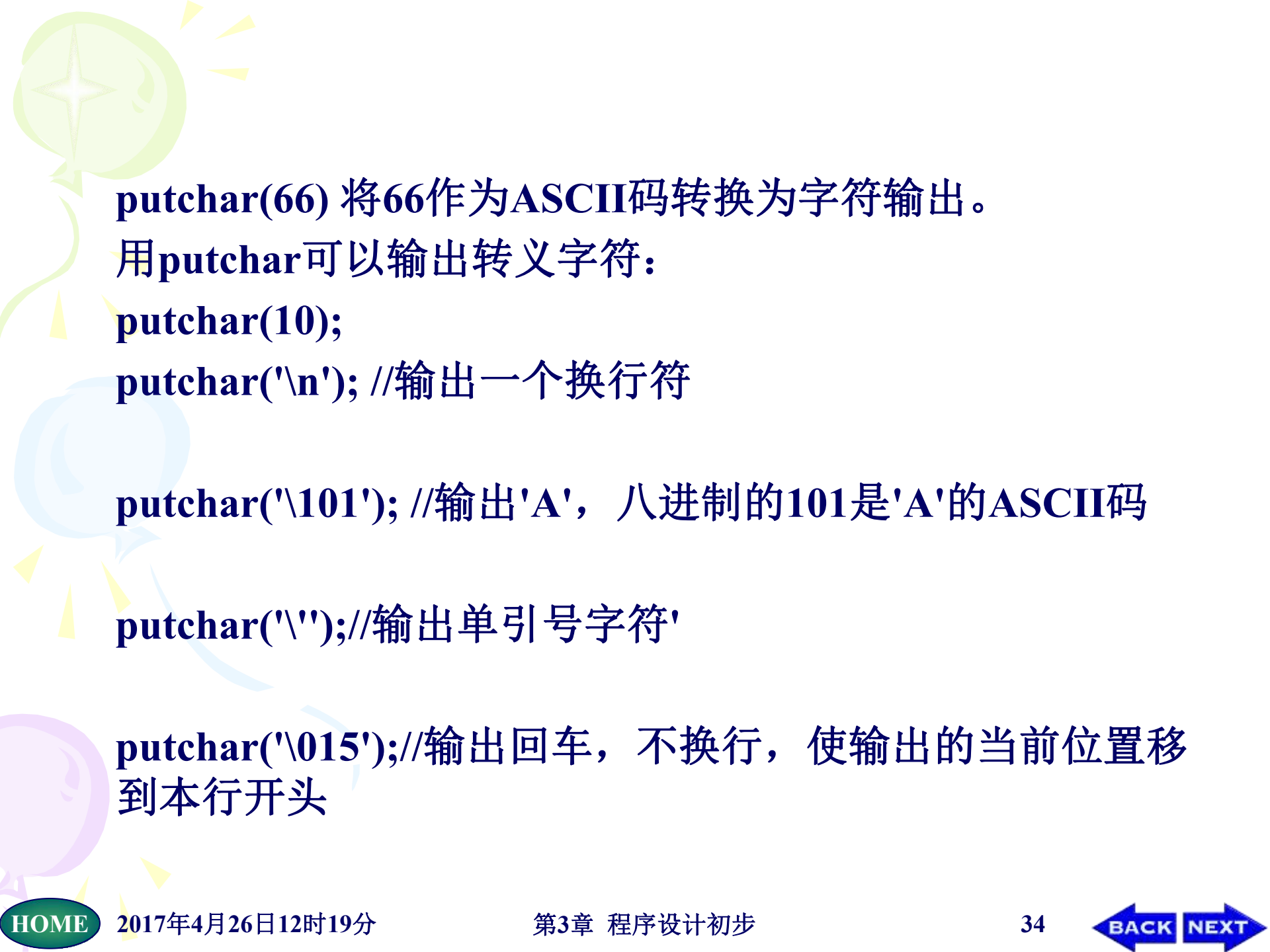
1. putchar函数（字符输出函数）

putchar函数的作用是向终端输出一个字符。例如
putchar(c);

它输出字符变量 c 的值。

例3.2 输出单个字符

```
#include <iostream>
using namespace std;
//或者包含头文件stdio.h: #include <stdio.h>
int main( )
{
    char a,b,c;
    a='B';b='O';c='Y';
    putchar(a);putchar(b);putchar(c);putchar('\n');
    putchar(66);putchar(79);putchar(89);putchar(10);
    return 0;
}
```



putchar(66) 将66作为ASCII码转换为字符输出。

用**putchar**可以输出转义字符：

putchar(10);

putchar('\n'); //输出一个换行符

putchar('\101'); //输出'A'，八进制的101是'A'的ASCII码

putchar('\\'); //输出单引号字符'

putchar('\015'); //输出回车，不换行，使输出的当前位置移到本行开头

2. **getchar**函数（字符输入函数）

此函数的作用是从终端输入一个字符。**getchar**函数没有参数，函数的值就是从输入设备得到的字符。

例3.3 输入单个字符。

```
#include <iostream>
using namespace std;
int main( )
{
    char c;
    c=getchar( ); putchar(c+32); putchar('\n');
    return 0;
}
```

注意，`getchar()`只能接收一个字符。

`getchar()`得到的字符可以直接作为表达式的一部分，如：

```
putchar (getchar()+32) ; putchar('\n');
```

```
cout<<getchar();
```

这时输出的是整数97，因为用`getchar()`读入的实际上是字符的ASCII码，现在并未把它赋给一个字符变量，`cout`就按整数形式输出。

```
cout<<(c=getchar());           //设c已定义为字符变量
```

则输出为字母'a'，因为要求输出字符变量c的值。

```
cout<<(c=getchar()+32);       //合法
```

3.4.4 用scanf和printf函数进行输入和输出

C++保留了C语言中的printf函数、scanf函数进行输出输入。

scanf函数一般格式是

scanf(格式控制, 输出表列)

printf函数的一般格式是

printf(格式控制, 输出表列)

例3.4 用scanf和printf函数进行输入和输出。

```
#include <iostream>
using namespace std;
int main( )
{
    int a; float b; char c;
    scanf("%d %c %f",&a,&c,&b);
    //变量名前要加地址运算符&
    printf("a=%d,b=%f,c=%c\n",a,b,c);
    return 0;
}
```

3.5 编写顺序结构的程序

例3.5 求一元二次方程 $ax^2+bx+c=0$ 的根。 a,b,c 的值在运行时由键盘输入，它们的值满足 $b^2-4ac \geq 0$ 。

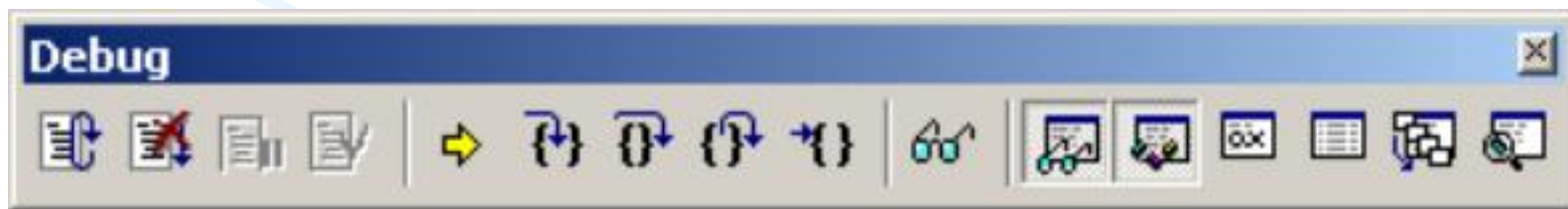
```
#include <iostream>
#include <cmath>           //数学函数库
using namespace std;
int main( )
{
    float a,b,c,x1,x2;
    cin>>a>>b>>c;
    x1=(-b+sqrt(b*b-4*a*c))/(2*a);
    x2=(-b-sqrt(b*b-4*a*c))/(2*a);
    cout<<"x1="<<x1<<endl;
    cout<<"x2="<<x2<<endl;
    return 0;
}
```

Debug快捷键说明

- **F5** 继续运行
- **Shift+F5** 停止
- **CTRL+Shift+F5** 重新开始
- **F10**单步，如果涉及到子函数，不进入子函数内部
- **F11**单步，如果涉及到子函数，进入子函数内部
- **Shift+F11**单步，退出子函数
- **CTRL+F10**运行到当前光标处。

Developer Studio的跟踪调试功能

- (1) Restart (快捷键: Ctrl+Shift+F5)
- (2) Stop Debugging (快捷键: Shift+F5)
- (3) Break Execution:
- (4) Apply Code Changes (快捷键: Alt+F10)
- (5) Show Next Statement (快捷键: Alt+Num *)
- (6) Step Into (快捷键: F11)
- (7) Step Over (快捷键: F10)
- (8) Step Out (快捷键: Shift+F11)
- (9) Run To Cursor (快捷键: Ctrl+F10)



- (1) 观察
- (2) 变化
- (3) 寄托
- (4) 内容
- (5) 调
- (6) 汇

- 



3.6 关系运算和逻辑运算

3.6.1 关系运算和关系表达式

- C++的关系运算符有：
- $<$ 、 $<=$ 、 $>$ 、 $>=$ 优先级相同 (高)
- $==$ 、 $!=$ 优先级相同 (低)
- 关系运算符的优先级低于算术运算符，高于赋值运算符。
- 关系表达式的值是**逻辑**（布尔）型数据（整数子类），其值为：**true**（1）或**false**（0）。
- 关系运算可以作为逻辑运算的子运算，构成比较复杂的逻辑表达式。

3.6.2 逻辑常量和逻辑变量

逻辑型数据只有两个值，1（**true**）和0（**false**），表示“真”和“假”。用**bool**来定义，如

```
bool found=true, flag=false;
```

逻辑变量又称为布尔（George Boole）变量。

编译系统将true处理为1，将false处理为0，因此逻辑可以与数值型数据进行算术运算。

如果将**非零数据**赋给逻辑型变量，则按“真”处理，如

```
flag=123; cout<<flag;
```

3.6.3 逻辑运算和逻辑表达式

C++提供3种逻辑运算符:

&& (AND) **||** (OR) **!** (NOT)

逻辑运算符的优先级:

(1) **!** $\rightarrow$ **&&** $\rightarrow$ **||**, 即 “**!**”为三者中最高。

(2) 逻辑运算符中的 “**&&**”和 “**||**”低于关系运算符,
“**!**”高于算术运算符。

例如:

(a>b) && (x>y) 可写成 **a>b && x>y**

(a==b) || (x==y) 可写成 **a==b || x==y**

(!a) || (a>b) 可写成 **!a || a>b**

逻辑表达式的值是一个逻辑量“真”或“假”。

例如：

(1) 若 $a=4$ ，则 $!a$ 的值为0。

(2) 若 $a=4, b=5$ ，则 $a \&\& b$ 的值为1。

(3) $4 \&\& 0 \parallel 2$ 的值为1。

实际上，逻辑运算符两侧的表达式不但可以是关系表达式或整数(0和非0)，也可以**是任何类型的数据**，如字符型、浮点型或指针型等。系统最终以**0**和**非0**来判定它们属于“真”或“假”。

例如： $a < b < c$;//语义错但不报错，

应改为： $b > a \&\& b < c$;//构成逻辑表达式

3.7 选择结构和条件语句

if语句也称条件语句（**conditional statements**）是用来判定所给定的条件是否满足，根据判定的结果（真或假）决定执行相应操作。

它与switch语句合称为分支语句。

Three kinds of if-sentence

- **if (expression) sentence;**
 if (x>y) cout<<x;
- **if (expression) sentence1;**
 else sentence2;
 if (x>y) cout<<x;
 else cout<<y;
- **if (expression1) sentence1;**
 else if (expression2) sentence2;
 else if (expression3) sentence3;
 ...
 else sentence n;

Example of single-offset

- `#include <iostream>`
- `using namespace std;`
- `int main()`
- `{`
- `char ch;`
- `cout << "请输入字符: \n";`
- `cin >> ch1;`
- `if (ch1>='a' && ch1<='z')`
- `cout << ch-32 << endl;`
- `else`
- `cout << ch << endl;`
- `return 0;`
- `}`

if sentence——Nesting

- General Form

if()

if() sentence 1;

else sentence 2;

else

if() sentence 3;

else sentence 4;

- Notes:

- Sentence 1、2、3、4 may be compound sentence, and each “{” corresponds to its close next “}”.

Example of more-offset

- `#include <iostream>`
- `using namespace std;`
- `int main()`
- `{`
- `int x,y;`
- `cout<<"Enter x and y:";`
- `cin>>x>>y;`
- `if (x!=y)`
- `if (x>y)`
- `cout<<"x>y"<<endl;`
- `else`
- `cout<<"x<y"<<endl;`
- `else`
- `cout<<"x=y"<<endl;`
- `return 0;`
- `}`

3.7.1 if语句的3种形式

1. if（表达式）语句

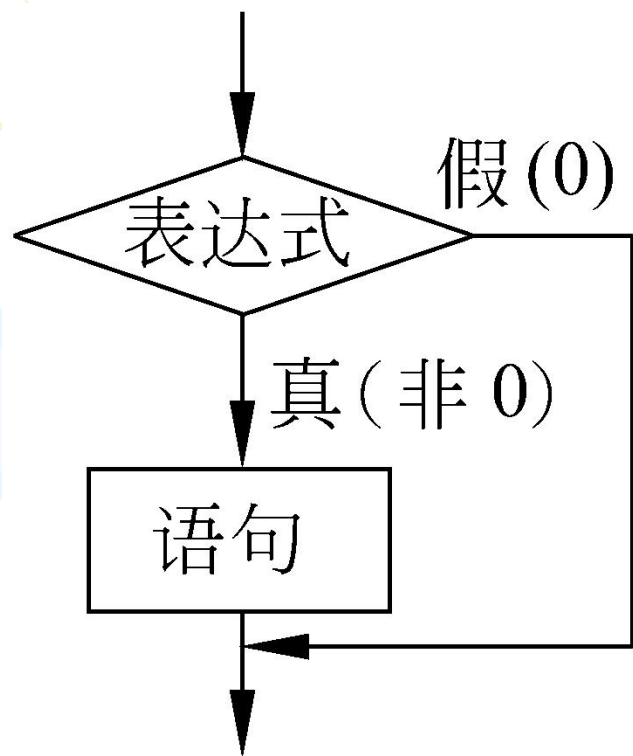
例如：

```
if(x>y) cout<<x<<endl;
```

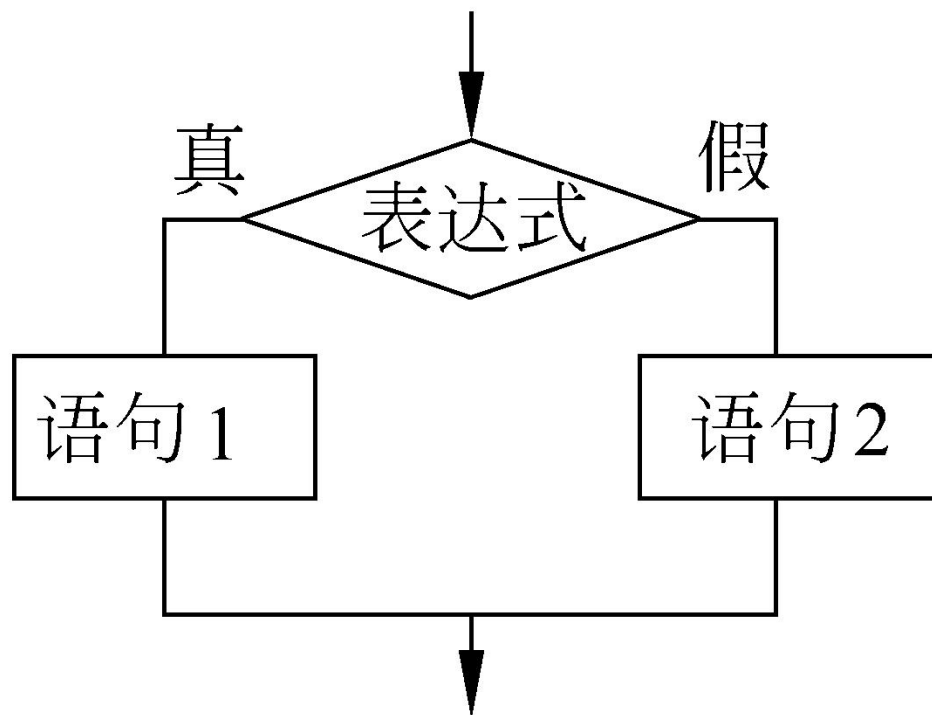
2. if（表达式）语句1 else 语句2

例如：

```
if (x>y) cout<<x;  
else cout<<y;
```



(a)



(b)



3. if(表达式1) 语句1

else if(表达式2) 语句2

else if(表达式3) 语句3

...

else 语句n

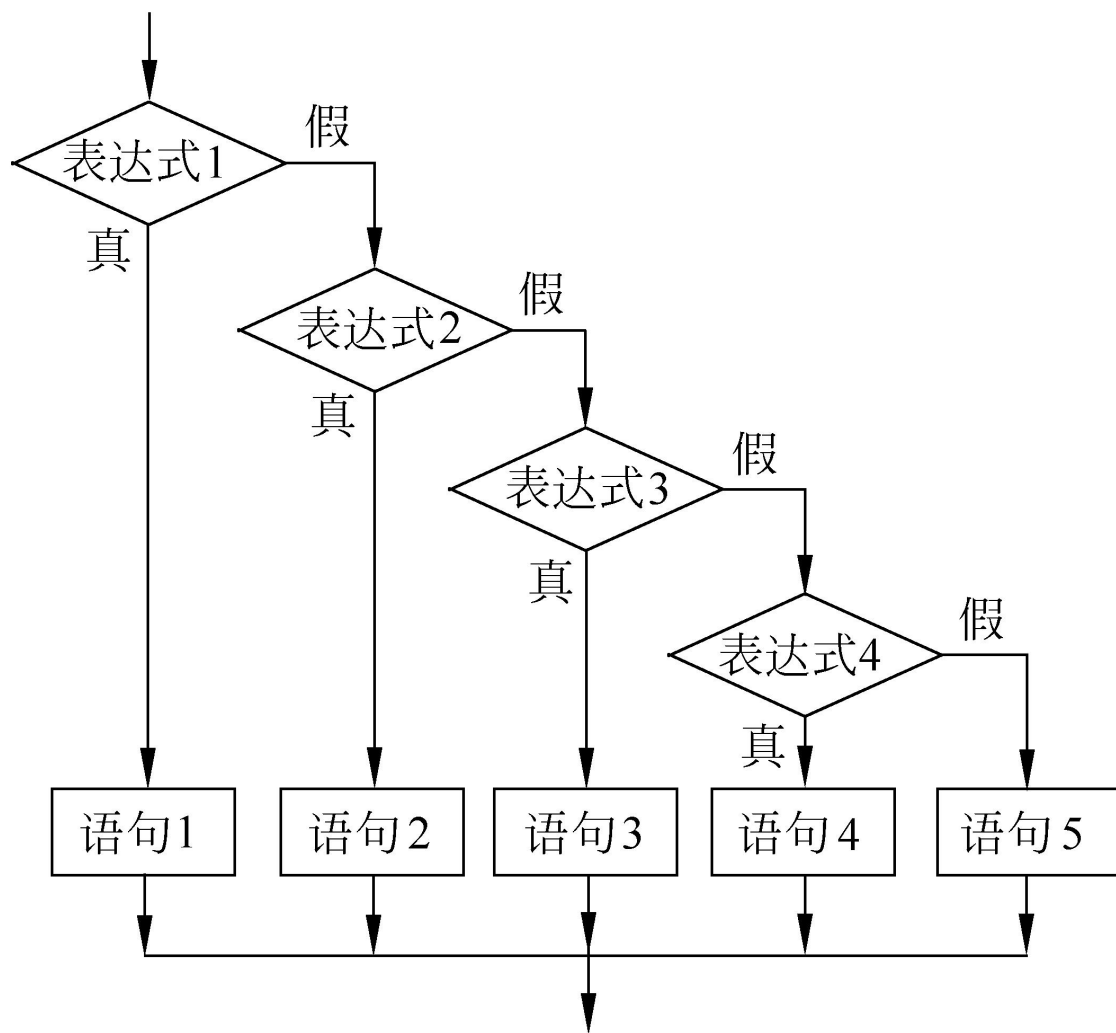
例如:

if (number>500) cost=0.15;

else if(number>300) cost=0.10;

else if(number>100) cost=0.05;

else cost=0;



说明

- (1) if语句有一个入口，一个出口。
- (2) if后面都有一个括起来的逻辑表达式或关系表达式，是要求程序判断的“条件”。
- (3) 在else前应分号，整个语句结束处有一分号。
- (4) 在if和else后面的多个操作语句应用花括号“{ }”括起来成为一个复合语句。

例3.6 求三角形的面积

```
#include <iostream>
#include <cmath>           //使用数学函数时要包含头文件cmath
#include <iomanip>          //使用I/O流控制符要包含头文件iomanip
using namespace std;
int main( )
{
    double a,b,c;
    cout<<"please enter a,b,c: ";
    cin>>a>>b>>c;
    if (a+b>c && b+c>a && c+a>b)
    {                          //复合语句开始
        double s,area;        //在复合语句内定义变量
        s=(a+b+c)/2;
        area=sqrt(s*(s-a)*(s-b)*(s-c));
        cout<<setiosflags(ios:: fixed)<<setprecision(4); //指定输出包含4位小数
        cout<<"area="<<area<<endl;           //在复合语句内输出局部变量的值
    }                          //复合语句结束
    else cout<<"it is not a trilateral!"<<endl;
    return 0;
}
```

3.7.2 if语句的嵌套

在if语句中又包含一个或多个if语句称为if语句的嵌套。一般形式如下：

if()

if()语句1

else 语句2

} 内嵌if

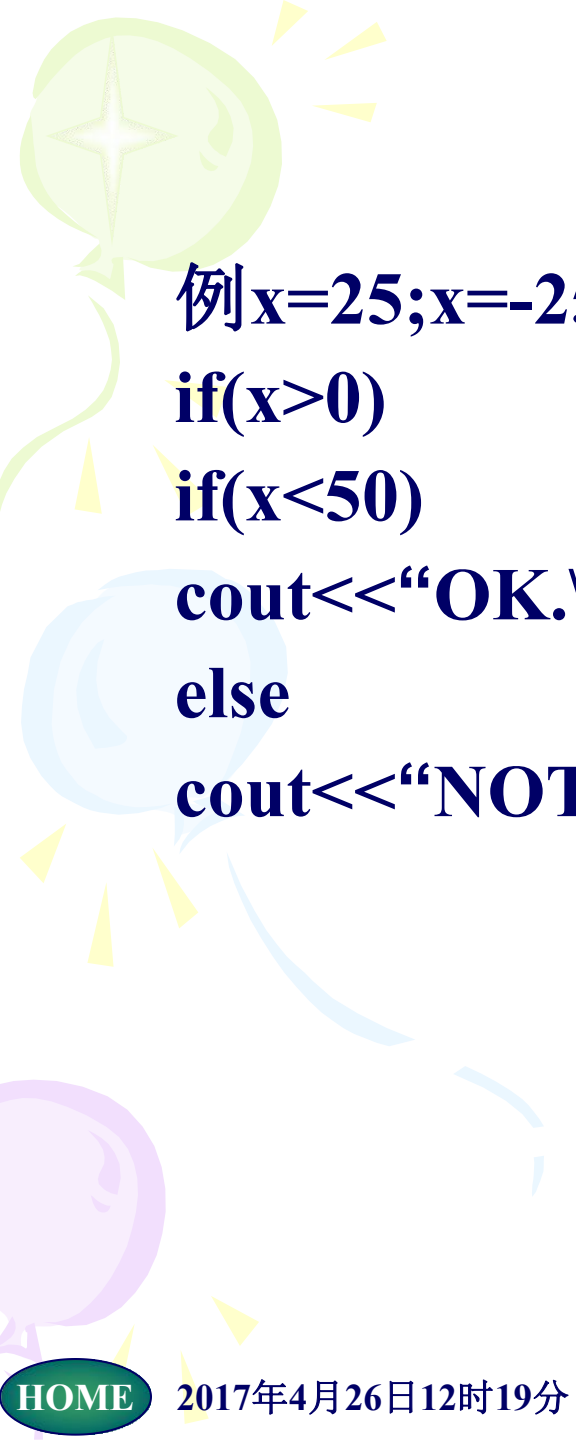
else

if()语句3

else 语句4

} 内嵌if

应当注意if与else的配对关系。**else总是与它上面最近的、且未配对的if配对。**



例`x=25;x=-25;`

if`(x>0)`

if`(x<50)`

cout`<<"OK.\n";`

else

cout`<<"NOT OK.\n";`



```
int x=-25;
```

```
if(x>0)
```

```
    if(x<50)
```

```
        cout<<"OK.\n";
```

```
    else
```

```
        cout<<"NOT OK.\n";
```

```
int x=-25;
```

```
if(x>0)
```

```
    if(x<50)
```

```
        cout<<"OK.\n";
```

```
    else
```

```
        cout<<"NOT OK.\n";
```

else总是与它上面最近的、且未配对的if配对。

尽量使每一层内嵌的if语句都包含else子句；
若if与else的数目不一样，可加花括号来界定：

if()

{ if () 语句1 } //本句是上if语句的内嵌if

else 语句2 //本行与第一个if配对

{ }限定了内嵌if语句的范围，{ }外的else不会与{ }
内的if配对。

例3.8 编写程序，判断某一年是否为闰年。

闰年的条件：

- ①能被4整除，但不能被100整除。
- ②能被100整除，又能被400整除。

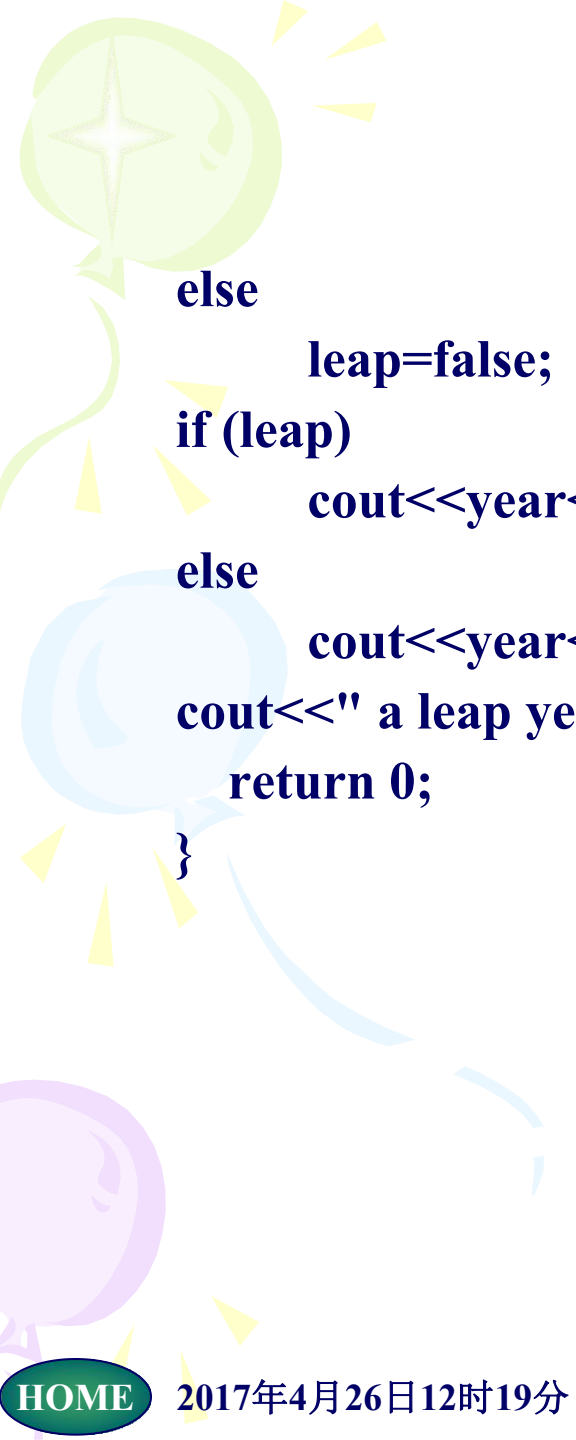
```
if (year%4==0)
```

```
    if(year%100==0)
```

```
        if (year%400==0) leap=true;
```

例3.8 判断某一年是否为闰年。

```
#include <iostream>
using namespace std;
int main( )
{
    int year;
    bool leap;
    cout<<"please enter year: ";           //输出提示
    cin>>year;                             //输入年份
    if (year%4==0)                          //年份能被4整除
    {
        if(year%100==0)                   //年份能被4整除又能被100整除
        {
            if (year%400==0)              //年份能被4整除又能被400整除
                leap=true;                //是闰年,令leap=true(真)
            else leap=false;              //非闰年, 令leap=false(假)
        }
        else                             //年份能被4整除但不能被100整除肯定是闰年
            leap=true;                    //是闰年, 令leap=true
    }
}
```



```
else
```

```
    leap=false;
```

```
if (leap)
```

```
    cout<<year<<" is ";    //若leap为真，就输出年份和“是”
```

```
else
```

```
    cout<<year<<" is not "; //若leap为真，就输出年份和“不是”
```

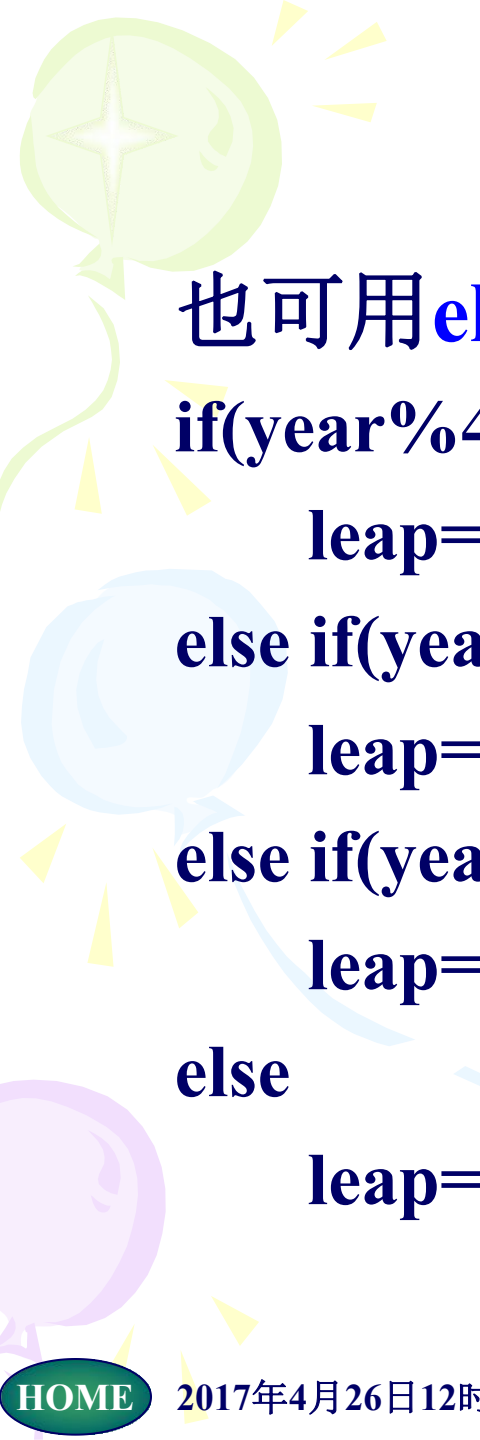
```
    cout<<" a leap year."<<endl; //输出“闰年”
```

```
    return 0;
```

```
}
```

//年份不能被4整除肯定不是闰年

//若为非闰年，令leap=false



也可用**else if**语句实现:

```
if(year%4!=0)
```

```
    leap=false;
```

```
else if(year%100!=0)
```

```
    leap=true;
```

```
else if(year%400!=0)
```

```
    leap=false;
```

```
else
```

```
    leap=true;
```



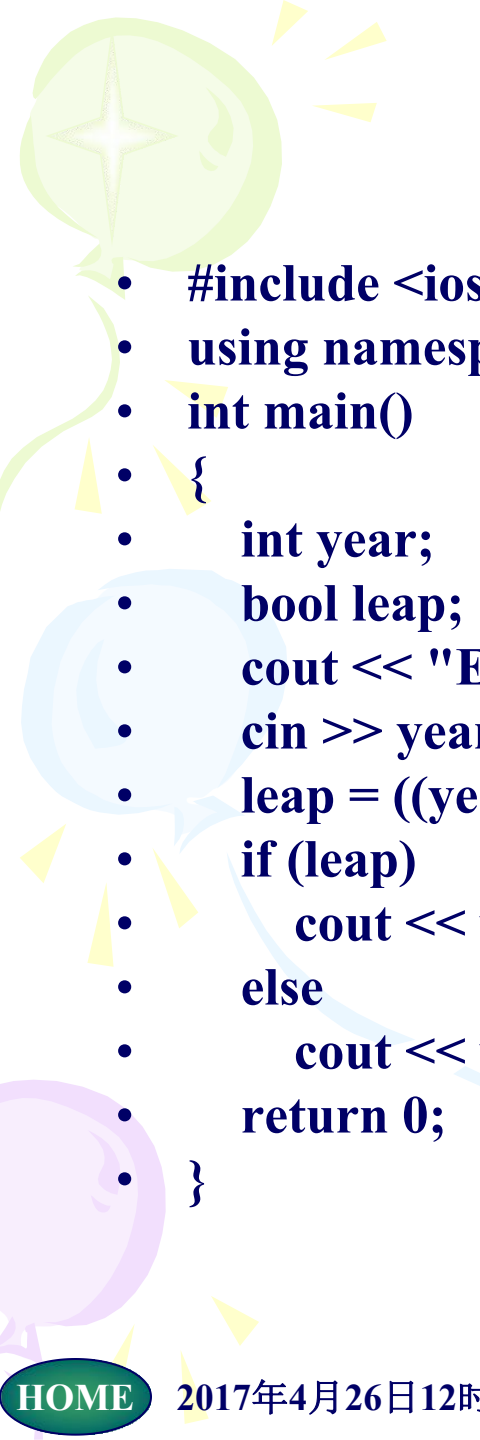
还可用一个逻辑表达式来表示:

```
if((year%4 == 0 && year%100 !=0) || (year%400 == 0))  
    leap=true;  
else leap=false;
```

如果判别非闰年:

```
!((year % 4 == 0 && year % 100 != 0) || year % 400 == 0)
```

```
(year % 4 != 0) || (year % 100 == 0 && year % 400 !=0)
```



- **#include <iostream>**
- **using namespace std;**
- **int main()**
- **{**
- **int year;**
- **bool leap;**
- **cout << "Enter the year: ";**
- **cin >> year;**
- **leap = ((year %4==0&&year%100!=0)||(year%400==0));**
- **if (leap)**
- **cout << year << " is a leap year"<< endl;**
- **else**
- **cout << year << " is not a leap year"<< endl;**
- **return 0;**
- **}**

C++初学者必看的50个建议

- 14.浮躁的人容易问：XX和YY哪个好；——告诉你吧，都好——只要你学就行；
- 15.浮躁的人分两种：a)只观望而不学的人；b)只学而不坚持的人；
- 16.把时髦的技术挂在嘴边，还不如把过时的技术记在心里；

例：按成绩等级输出分数段

利用if语句的嵌套来完成此题：学习成绩 ≥ 90 分的同学用A表示，60-89分之间的用B表示，60分以下的用C表示。

1.程序分析：

2.程序源代码：



```
• #include <iostream>
• using namespace std;
• int main( )
• {
•     int score;
•     char grade;
•     cout<<"please input a score\n";
•     cin>>score;
•     if(score>=60)
•     {
•         if(score>=90)
•             grade= 'A';
•         else
•             grade= 'B';
•     }
•     else
•         grade= 'C';
•     cout<<score<<" belongs to " <<grade<<'\n';
•     return 0;
• }
```

3.7.3 条件运算符和条件表达式

以下if语句：

```
if (a>b) max=a;
```

```
else max=b;
```

可以用条件运算符(? :)来处理：

```
max=(a>b)?a:b;
```

如果(a>b)条件为真，

则条件表达式的值就取 “?”后面的值，即a，

否则条件表达式的值为 “:”后面的值，即b。

- 
- 条件运算符是一个三目运算符。条件表达式的一般形式为

表达式1 ? 表达式2 : 表达式3

- 条件运算符的执行顺序是：
 - 若表达式1的值为1（真）则求解表达式2，表达式2的值就是整个条件表达式的值。
 - 若表达式1的值为0（假）则求解表达式3，表达式3的值就是整个条件表达式的值。

• 条件表达式的一般形式为

表达式1 ? 表达式2 : 表达式3

表达式1的类型可以与表达式2和3的类型不同。如
`x ? 'a' : 'b'`

如果x为整型变量，若x=0，则条件表达式的值为字符'b'的ASCII码。

表达式2和表达式3的类型也可以不同，结果的类型为二者中较高的类型。设有

`x > y ? 1 : 1.5;`

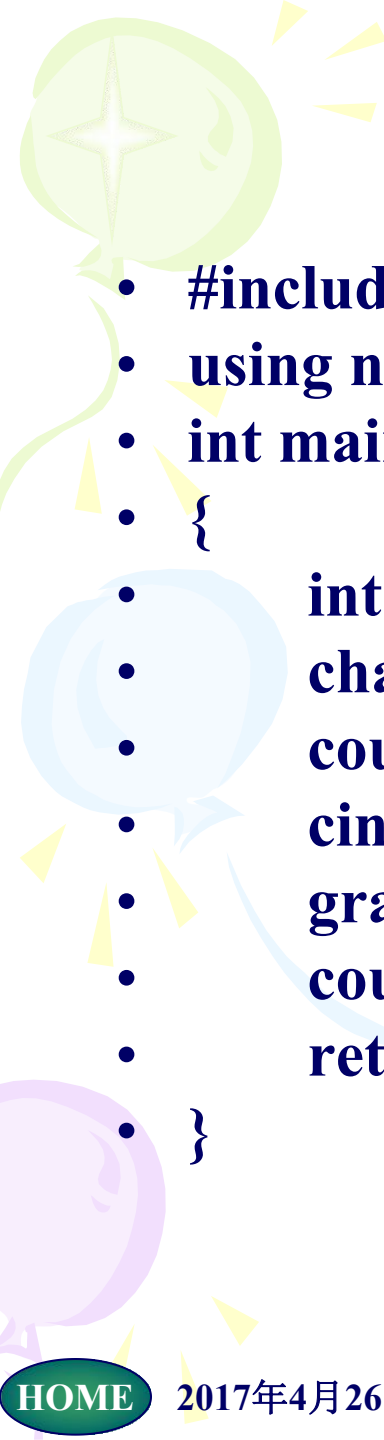
`cout << sizeof(x > y ? 1 : 1.5);`

例3.7 输入一个字符，判别它是否为大写字母，如果是，将它转换成小写字母；如果不是，不转换。然后输出最后得到的字符。

```
#include <iostream>
using namespace std;
int main( )
{
    char ch;
    cin>>ch;
    ch=(ch>='A' && ch<='Z')?(ch+32):  ch;    //判别ch是
    否大写字母，是则转换
    cout<<ch<<endl;
    return 0;
}
```

例：按成绩等级输出分数段

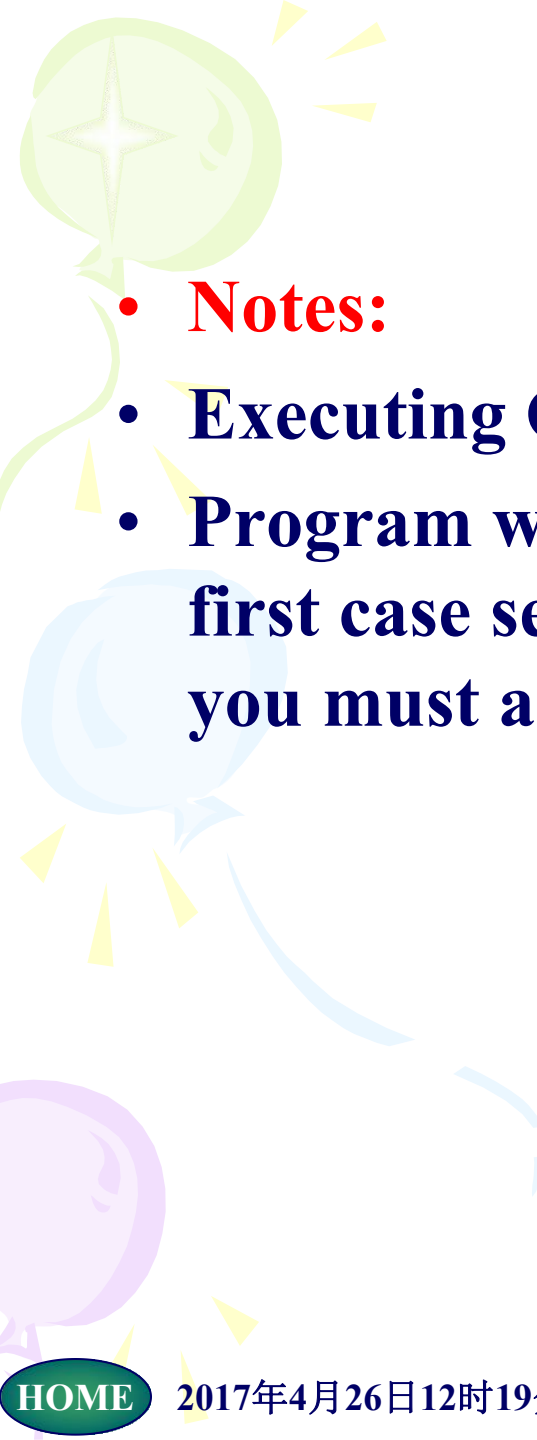
- 利用条件运算符的嵌套来完成此题：学习成绩 ≥ 90 分的同学用A表示，60-89分之间的用B表示，60分以下的用C表示。
 - 1.程序分析： $(a > b) ? a : b$ 这是条件运算符的基本例子。
 - 2.程序源代码：



- **#include <iostream>**
- **using namespace std;**
- **int main()**
- **{**
- **int score;**
- **char grade;**
- **cout<<"please input a score\n";**
- **cin>>score;**
- **grade=score>=90?'A':(score>=60?'B':'C');**
- **cout<<score<<" belongs to "<<grade<<"\n";**
- **return 0;**
- **}**

3.7.4 switch sentence

- **General Form**
- **switch (expression)**
- **{**
- **case constant expression 1: sentence1**
- **case constant expression 2: sentence2**
- **⋮**
- **case constant expression n: sentence (n)**
- **default : sentence(n+1)**
- **}**

- 
- **Notes:**
 - **Executing Order**
 - **Program will be executed sequentially from the first case sentence. So, behind each case sentence, you must add a sentence of “break”.**

3.7.4 多分支选择结构和switch 语句

switch语句是多分支选择语句，用来实现多分支选择结构。

它的一般形式如下：

switch（表达式）

{

case 常量表达式1: 语句1

case 常量表达式2: 语句2

... ..

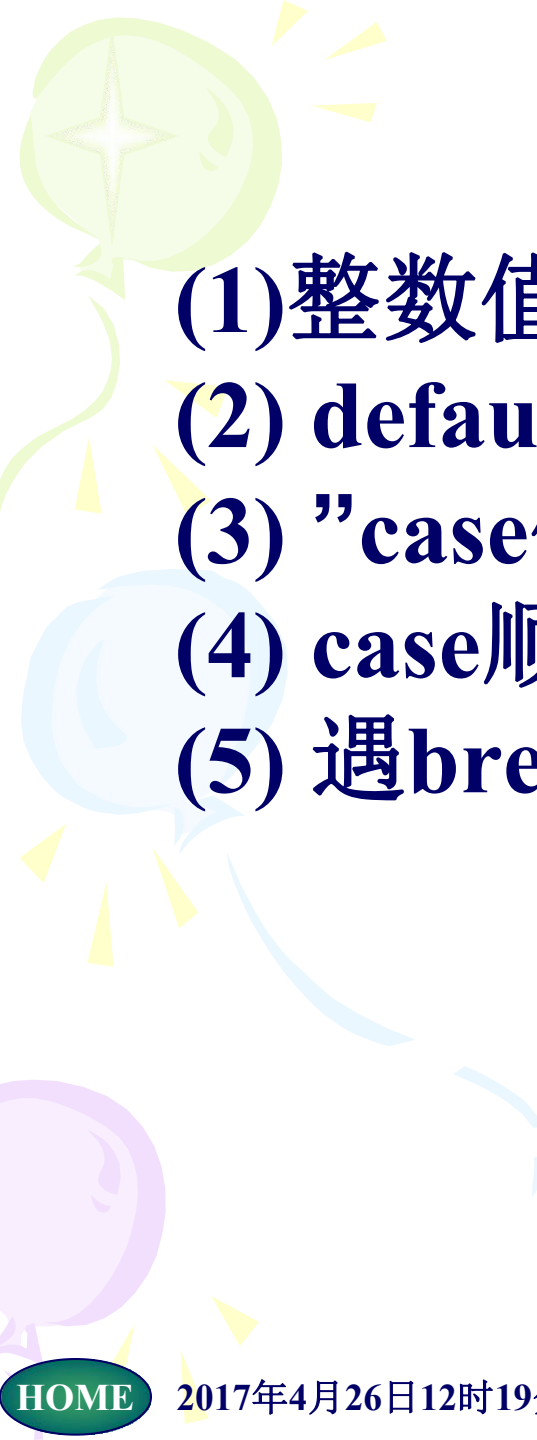
case 常量表达式n: 语句n

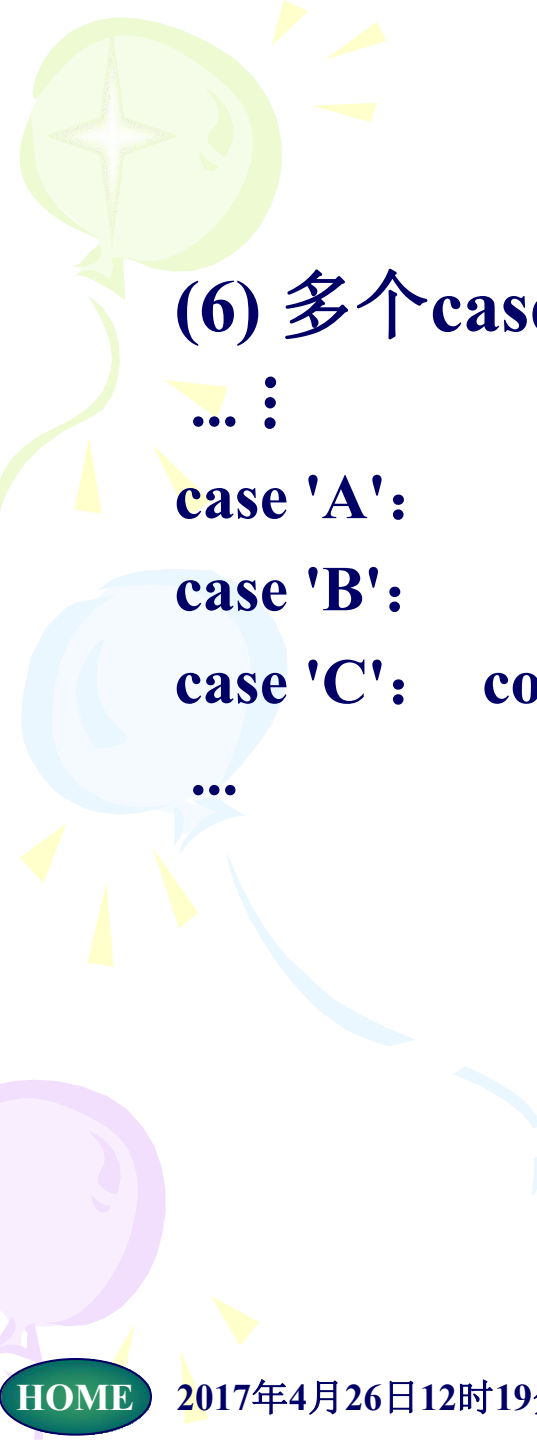
default: 语句n+1

}

执行流程

- switch表达式的值进行**一次**判断；
- 找到与之匹配的case子句，“case常量表达式”只是起语句标号作用，不执行判断；
- 从该case子句开始执行下去；
- 在case子句中包含多个执行语句可以不必用花括号括起来。
- 执行完一个case子句后，流程控制转移到下一个case子句继续执行。
- 在case子句后，通常用break语句使流程跳出switch结构。

- 
- (1) 整数值分支判断
 - (2) default分支
 - (3) "case值: " 即标号
 - (4) case顺序随意
 - (5) 遇break跳出



(6) 多个case可以共用一组执行语句，如

... :

case 'A':

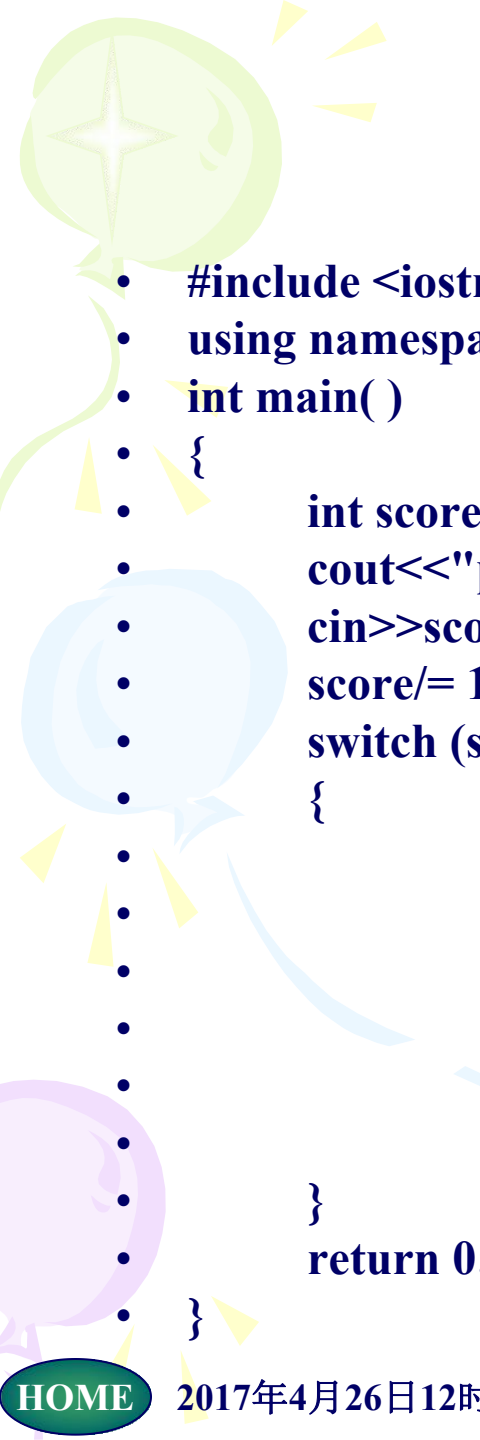
case 'B':

case 'C': cout<<">60\n"; break;

...

例：按成绩等级输出分数段

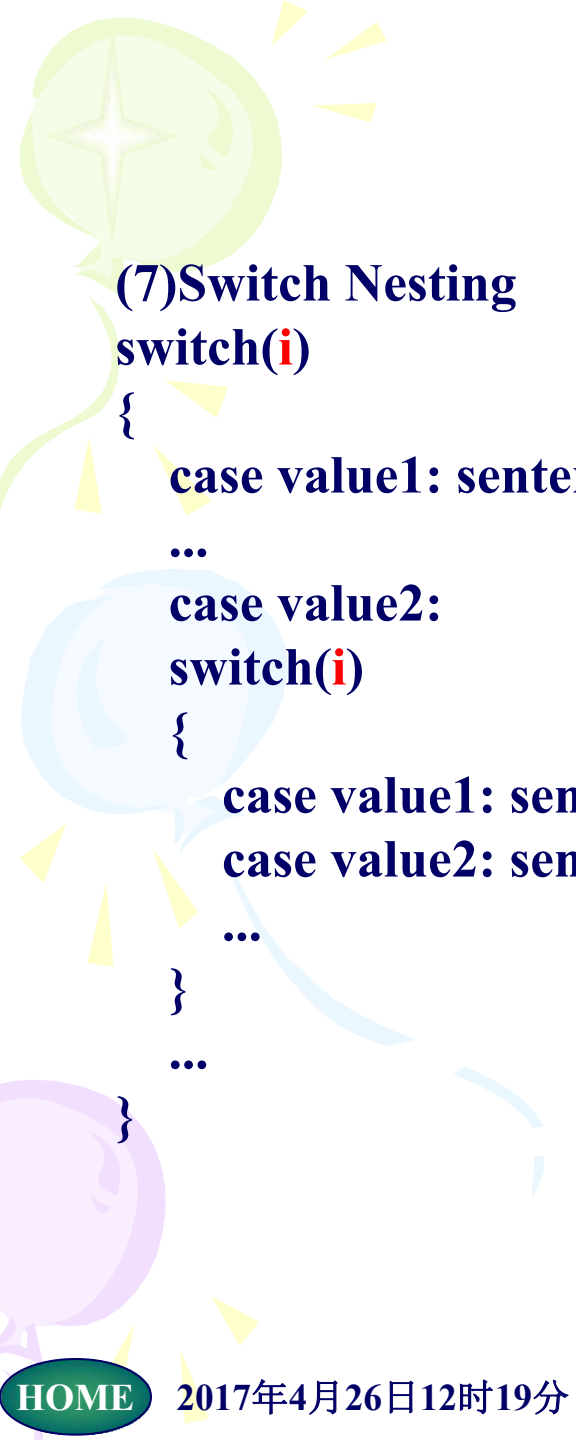
- 利用switch来完成此题：学习成绩 ≥ 90 分的同学用A表示，60-89分之间的用B表示，60分以下的用C表示。
 - 1.程序分析：
 - 2.程序源代码：



```

• #include <iostream>
• using namespace std;
• int main( )
• {
•     int score;
•     cout<<"please input a score\n";
•     cin>>score;
•     score/= 10;
•     switch (score)
•     {
•         case 10:
•         case 9:cout<<"A"<<\n';break;
•         case 8:
•         case 7:
•         case 6:cout<<"B"<<\n';break;
•         default:cout<<"C"<<\n';
•     }
•     return 0;
• }

```



(7)Switch Nesting

```
switch(i)
```

```
{
```

```
    case value1: sentence 1; break;
```

```
    ...
```

```
    case value2:
```

```
        switch(i)
```

```
        {
```

```
            case value1: sentence 1; break;
```

```
            case value2: sentence 2; break;
```

```
            ...
```

```
        }
```

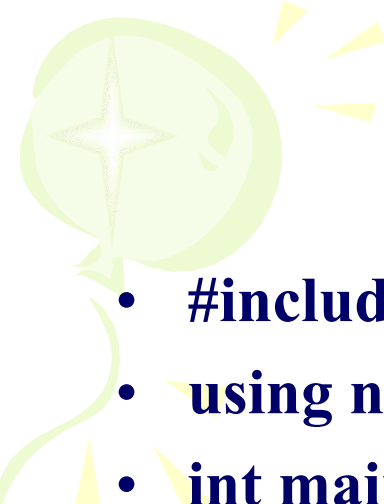
```
    ...
```

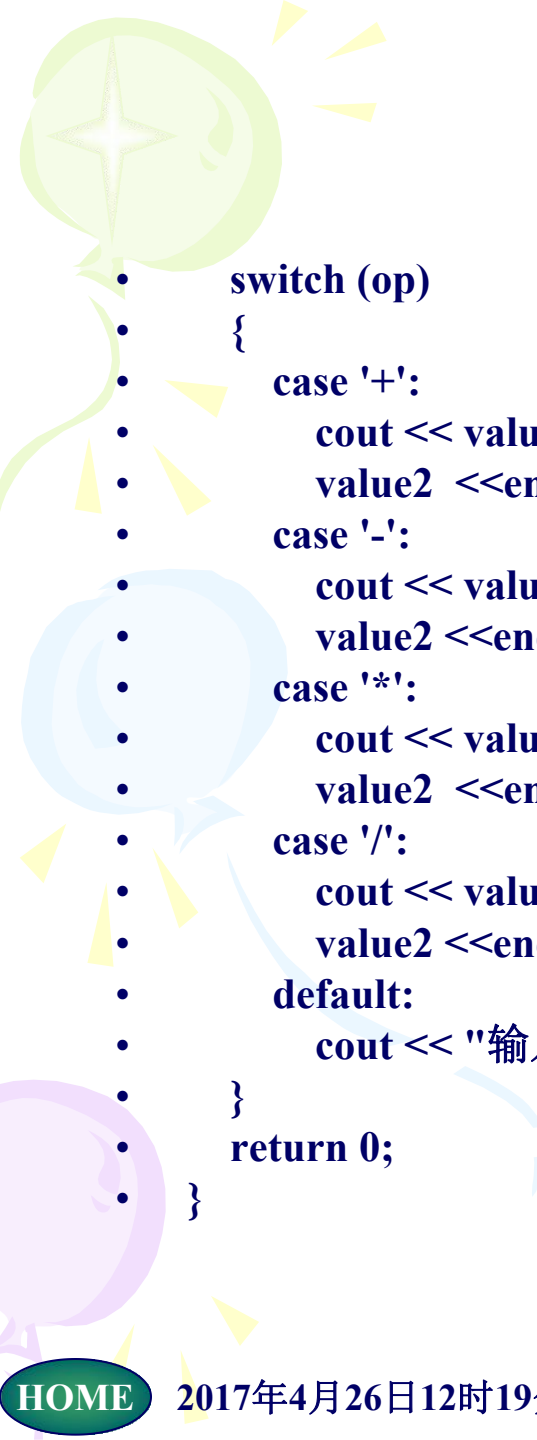
```
}
```

- **switch**多分支语句:
- **switch(整数表达式)**
- **{**
- **case value1: 语句1; break;**
- **case value2: 语句2; break;**
- **...**
- **default: 语句n;**
- **}**
- 等价于:
- **if(整数表达式==value1) 语句1;**
- **else if(整数表达式==value2) 语句2;**
- **...**
- **else 语句n;**

Switch语句的应用

- 例 输入两个整型数和一个算术运算符，输出对应的表达式及计算结果。
- 问题分析
- (1) 所输入的两个数按其输入顺序分别作为第一操作数 (Value1) 和第二操作数 (Value2) ;
- (2) 算术运算符op限制为+、-、*、/，四种；
- (3) 表达式表示为：Value1 <算术运算符> Value2。

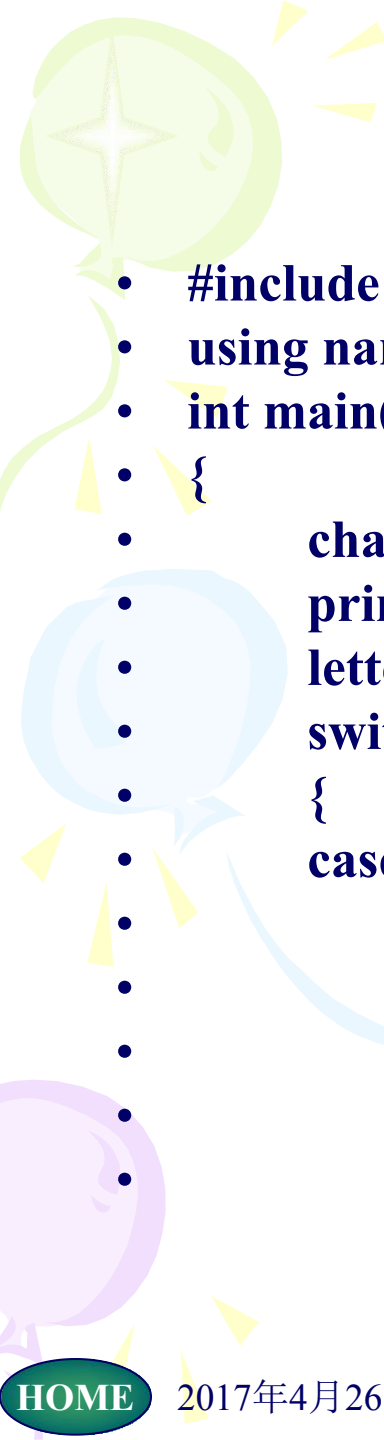
- 
- `#include <iostream>`
 - `using namespace std;`
 - `int main( )`
 - `{`
 - `int value1,value2;`
 - `char op;`
 - `cout << "请输入两个操作数: \n";`
 - `cin >> value1 >> value2; //输入操作数1和操作数2`
 - `cout << "请输入操作符: \n";`
 - `cin >> op; //输入操作符到op变量中`



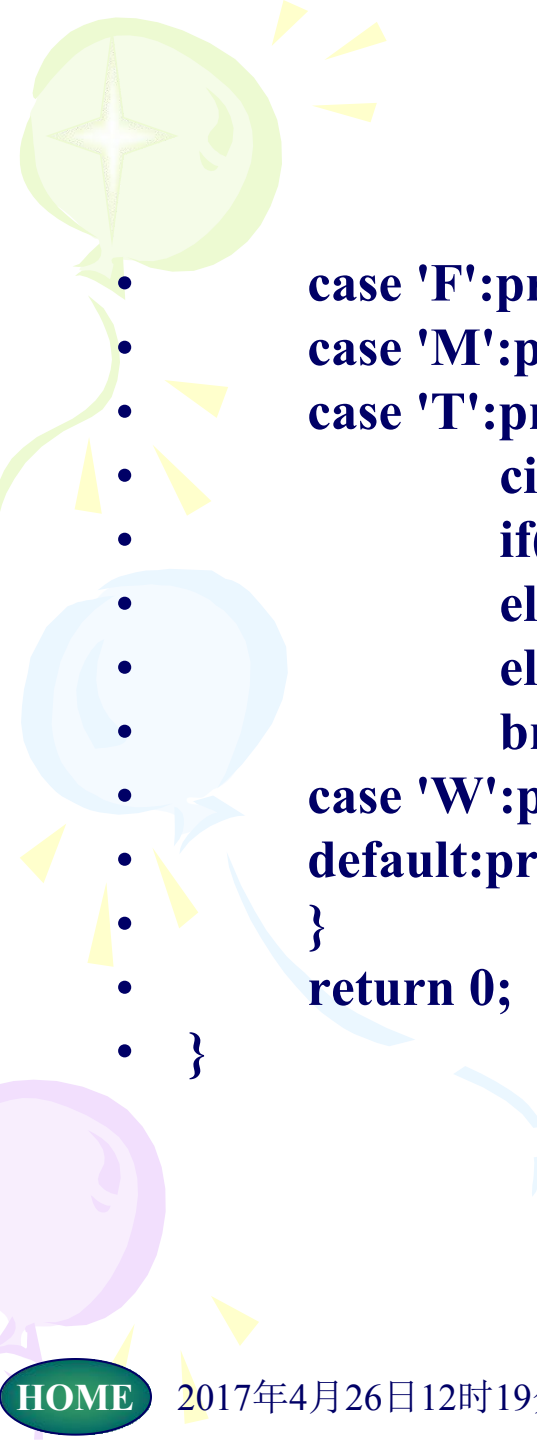
```
• switch (op)
• {
•     case '+':
•         cout << value1 << op << value2 << "=" <<value1+
•         value2 <<endl; break;
•     case '-':
•         cout << value1 << op << value2 << "=" <<value1-
•         value2 <<endl; break;
•     case '*':
•         cout << value1 << op << value2 << "=" <<value1*
•         value2 <<endl;break;
•     case '/':
•         cout << value1 << op << value2 << "=" <<value1/
•         value2 <<endl; break;
•     default:
•         cout << "输入的操作符有误！ " << endl;
• }
• return 0;
• }
```

课堂练习

- 题目：请输入星期几的第一个字母来判断一下是星期几，如果第一个字母一样，则继续判断第二个字母。
 - 1.程序分析：用switch比较好，如果第一个字母一样，则判断用switch或if语句判断第二个字母。
 - 2.程序源代码：



```
• #include <iostream>
• using namespace std;
• int main( )
• {
•     char letter;
•     printf("please input the first letter of someday\n");
•     letter=getchar( );
•     switch (letter)
•     {
•     case 'S':printf("please input second letter\n");
•             cin>>letter;
•             if(letter=='a') printf("saturday\n");
•             else if (letter=='u') printf("sunday\n");
•             else printf("data error\n");
•             break;
```



```
• case 'F':printf("friday\n");break;
• case 'M':printf("monday\n");break;
• case 'T':printf("please input second letter\n");
•     cin>>letter;
•     if(letter=='u') printf("tuesday\n");
•     else if (letter=='h') printf("thursday\n");
•     else printf("data error\n");
•     break;
• case 'W':printf("wednesday\n");break;
• default:printf("data error\n");
• }
• return 0;
• }
```

例3.9 运输公司对用户计算运费

例3.9 运输公司对用户计算运费。路程(s)越远，每公里运费越低。标准如下：

$s < 250\text{km}$ 没有折扣

$250 \leq s < 500$ 2%折扣

$500 \leq s < 1000$ 5%折扣

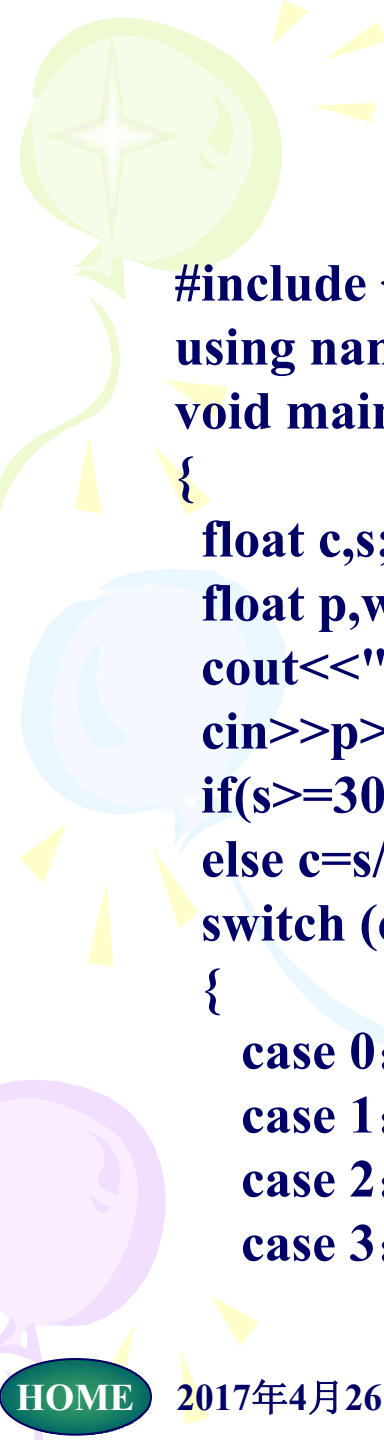
$1000 \leq s < 2000$ 8%折扣

$2000 \leq s < 3000$ 10%折扣

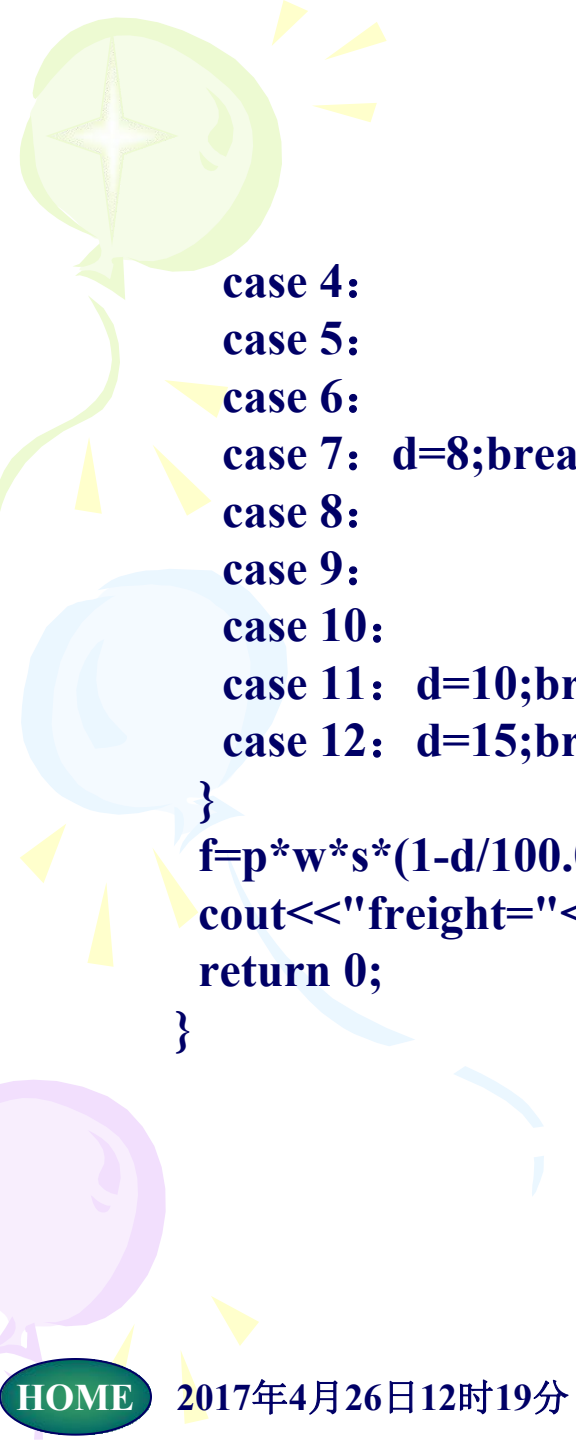
$3000 \leq s$ 15%折扣

设每公里每吨货物的基本运费为p(price)，货物重为w(wright)，距离为s，折扣为d(discount)，则总运费f(freight)的计算公式为

$$f = p * w * s * (1 - d)$$



```
#include <iostream.h>  
using namespace std;  
void main( )  
{  
    float c,s;  
    float p,w,d,f;  
    cout<<"please enter p,w,s:  ";  
    cin>>p>>w>>s;  
    if(s>=3000) c=12;  
    else c=s/250;  
    switch (c)  
    {  
        case 0: d=0;break;  
        case 1: d=2;break;  
        case 2:  
        case 3: d=5;break;  
    }
```



```
case 4:  
case 5:  
case 6:  
case 7: d=8;break;  
case 8:  
case 9:  
case 10:  
case 11: d=10;break;  
case 12: d=15;break;  
}  
f=p*w*s*(1-d/100.0);  
cout<<"freight="<<f<<endl;  
return 0;  
}
```

goto语句示例

- `#include<iostream>`
- `using namespace std;`
- `int main(){`
- `int a;`
- `goto Init;`
- **Forward:**
- `a = a + 1;`
- **Print:**
- `cout<<a<<endl;`
- `goto Down;`
- **Init:**
- `a = 1;`
- `goto Print;`
- **Down:**
- `if(a<100) goto Forward;`
- `}`



Control Structure of Algorithm

- Sequential structure
- Optional structure
- Cycle structure

3.8 循环结构和循环语句

顺序结构、选择结构和循环结构是结构化程序设计的3种基本结构，是各种复杂程序的基本构造单元。

- ① **for()~** (循环语句)
- ② **while()~** (循环语句)
- ③ **do~while()** (循环语句)

3.8.1 用while语句构成循环

while语句的一般形式:

while (condition)

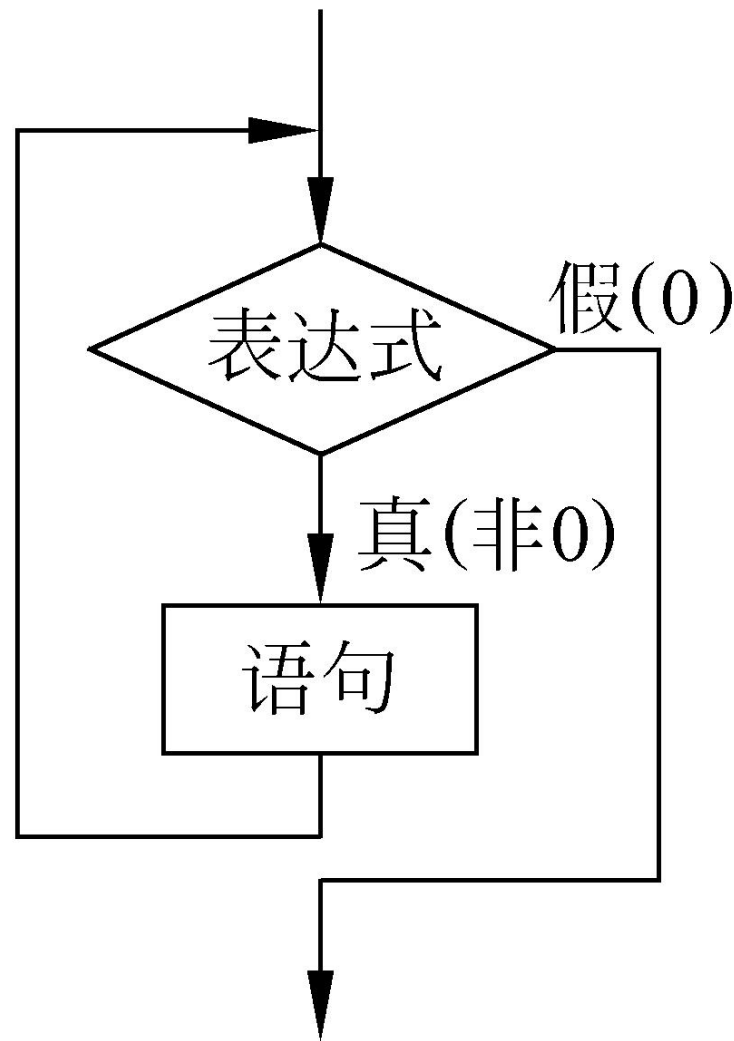
statement

执行序列如下

1 计算condition;

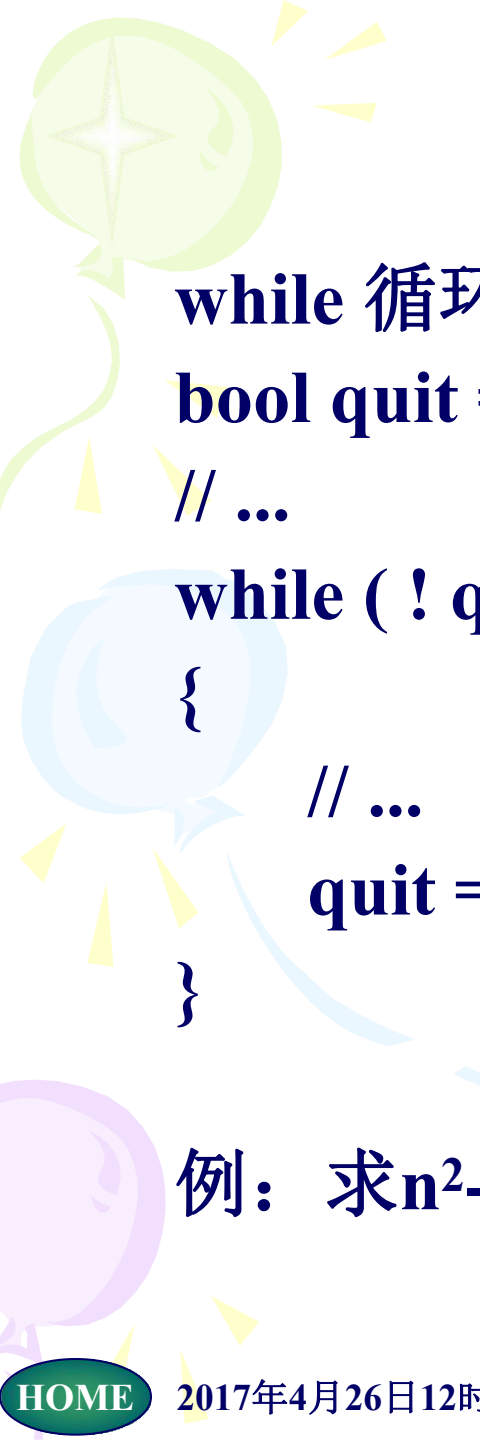
2 如果condition 为true 则
执行statement;

先判断表达式, 后执行语
句。



例3.10 求 $1+2+\dots+100$

```
#include <iostream>
using namespace std;
int main( )
{
    int i=1,sum=0;
    while (i<=100)
    {
        sum=sum+i;
        i++;
    }
    cout<<"sum="<<sum<<endl;
    return 0;
}
```



while 循环的另一种形式

```
bool quit = false;
```

```
// ...
```

```
while ( ! quit )
```

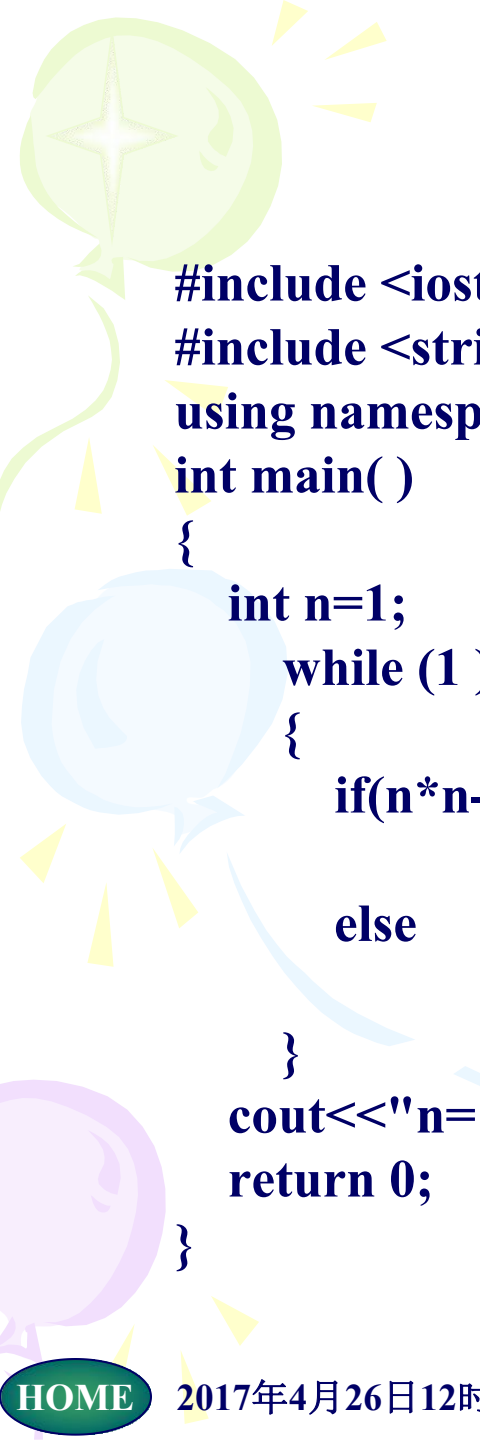
```
{
```

```
    // ...
```

```
    quit = do_something();
```

```
}
```

例：求 $n^2-6n-13$ ($n>0$)在 n 为多大时为正。



```
#include <iostream>
#include <string>
using namespace std;
int main( )
{
    int n=1;
    while (1 )
    {
        if(n*n-6*n-13>0)
            break;
        else
            ++n;
    }
    cout<<"n="<<n<<endl;
    return 0;
}
```

3.8.2 用do-while语句构成循环

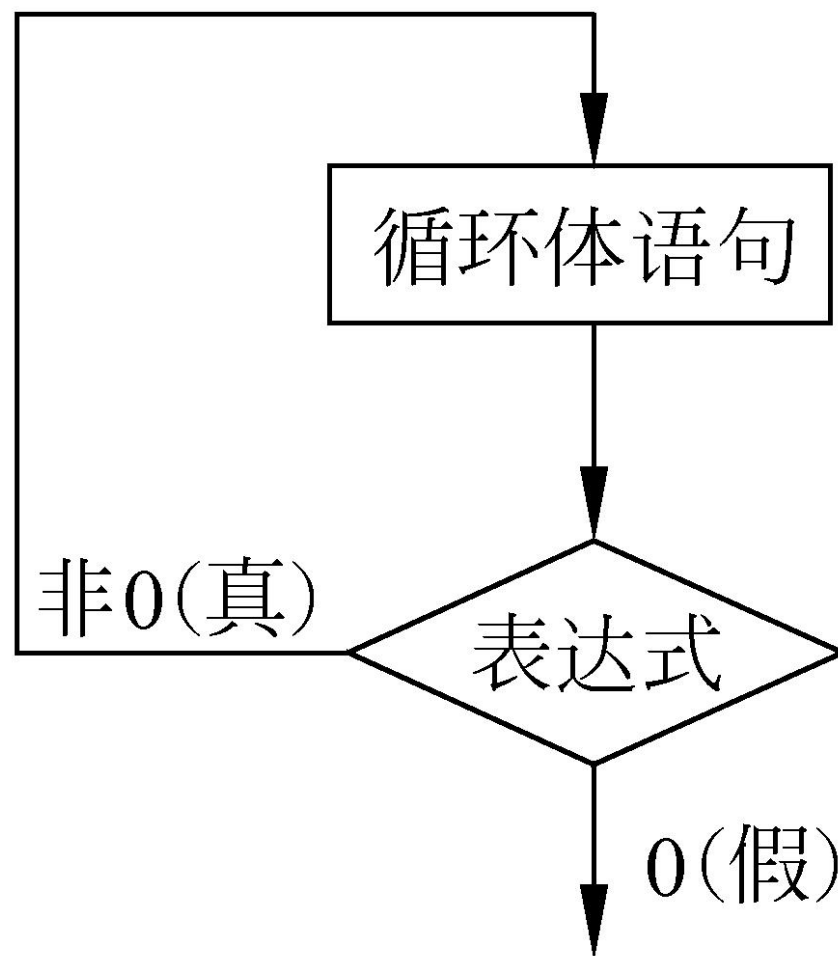
do-while语句的一般形式:

do

语句

while (表达式);

先执行一次指定的语句
(循环体)，然后判别表
达式(循环条件)是否成
立。



例3.11 用do-while语句求 $1+2+3+\dots+100$

```
#include <iostream>
using namespace std;
int main( )
{
    int i=1,sum=0;
    do
    {   sum=sum+i;
        i++;
    }while (i<=100);
    cout<<"sum="<<sum<<endl;
    return 0;
}
```

3.8.3 用for语句构成循环

C++中的for语句使用最为广泛和灵活，完全可以代替while语句。for语句的一般格式为

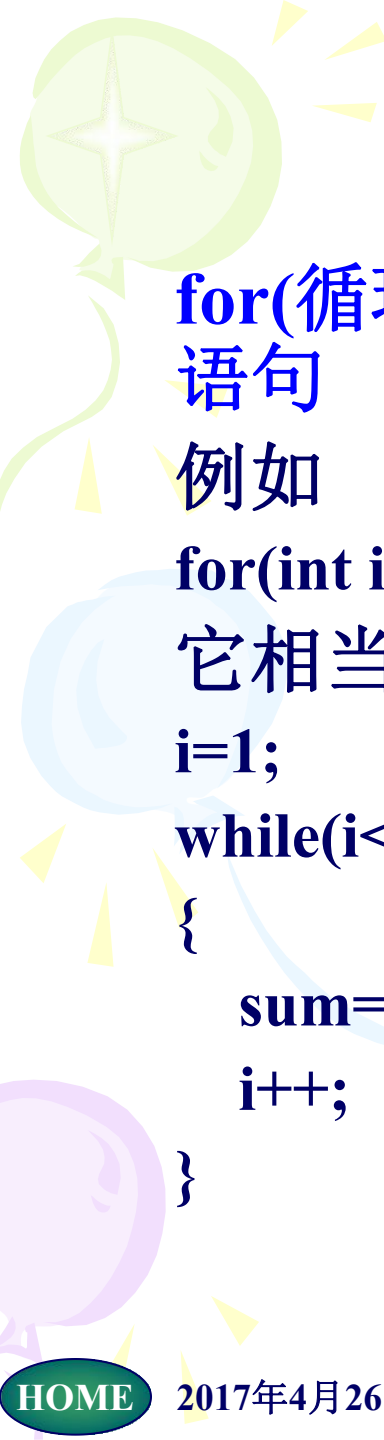
**for (init-statement; condition; expression)
statement**

init-statement 可以是声明语句或表达式，一般用于循环变量初始化

condition 用作循环控制

statement 是单个语句或复合语句

expression 在statement后修改循环变量。



**for(循环变量赋初值; 循环条件; 循环变量增值)
语句**

例如

```
for(int i=1;i<=100;i++) sum=sum+i;
```

它相当于:

```
i=1;
```

```
while(i<=100)
```

```
{
```

```
    sum=sum+i;
```

```
    i++;
```

```
}
```

说明

(1) for语句的init-statement可以省略，此时应在for语句之前给循环变量赋初值。

```
for(i=1;i<=100;i++)    //int i;
```

```
for(;i<=100;i++)      //int i=1;
```

在init-statement 中可以定义多个对象但只能出现一个声明语句因此所有对象都必须是相同的类型
例如

```
for ( int i = 0, *pi = &ia, &ri = val; i < size;  
    ++i, ++pi, ++ri )  
// ...
```

说明

(2) 循环条件可以省略，循环条件判断另行设置。

```
for(i=1;;i++)
```

(3) 循环变量增值也可以省略，循环增量另行设置。

```
for(i=1;i<=100;)
```

(4) 可以只给循环条件。 `for(;i<=100;)`

相当于 `while` 语句。

(5) 3个表达式都可省略。 `for(;;)`

无限循环，相当于 `while (1)`。

说明

(6) 表达式1可以是设置循环变量初值的赋值表达式，也可以是与循环变量无关的其他表达式。

(7) 循环条件一般是关系表达式(如 $i \leq 100$)或逻辑表达式(如 $a < b \ \&\& \ x < y$)，但也可以是数值表达式或字符表达式，只要其值为非零，就执行循环体。

不要把与循环控制无关的内容放到for语句中。

说明

(8)在for 循环condition 中定义的对象的可视性局限在for 循环体内。例如给定下面两个for 循环，它们位于同一个语句块之中

- {
- for (int i = 0; i < size; ++i) cout<<i<<endl;
- for (int i = size -1; i >= 0; --i) cout<<i<<endl;
- }
- 在标准C++之前i 被标记为编译错误“重复定义”；
- 然而在标准C++中i的两个实例都是局部的，仅限于它们各自的for 循环，所以程序逻辑是合法的。

3.8.4 几种循环的比较

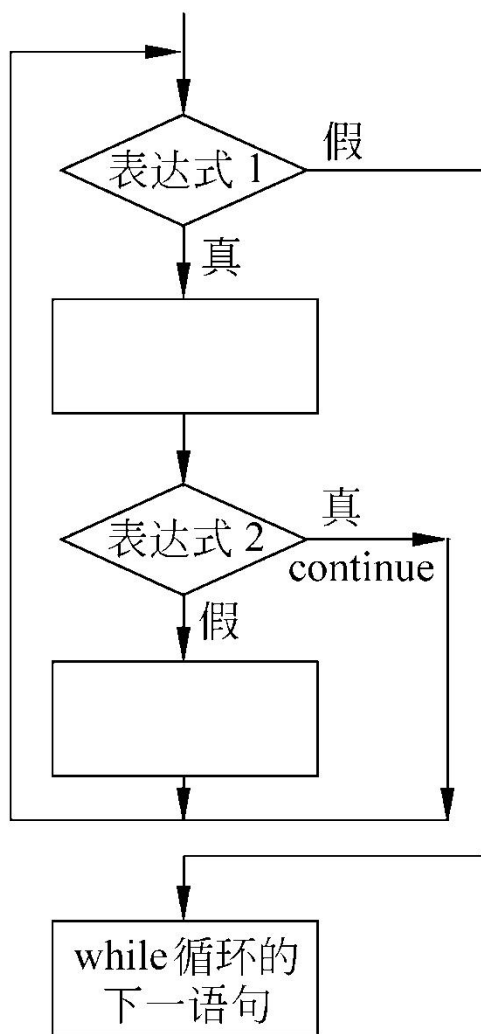
- (1) 3种循环都可以用来处理同一问题，一般情况下它们可以互相代替。
- (2) for语句的功能更强，凡用while循环能完成的，用for循环都能实现。
- (3) 用while和do-while循环时，循环变量初始化的操作应在while和do-while语句之前完成。而for语句可以在表达式1中实现循环变量的初始化。

3.8.5 循环的嵌套

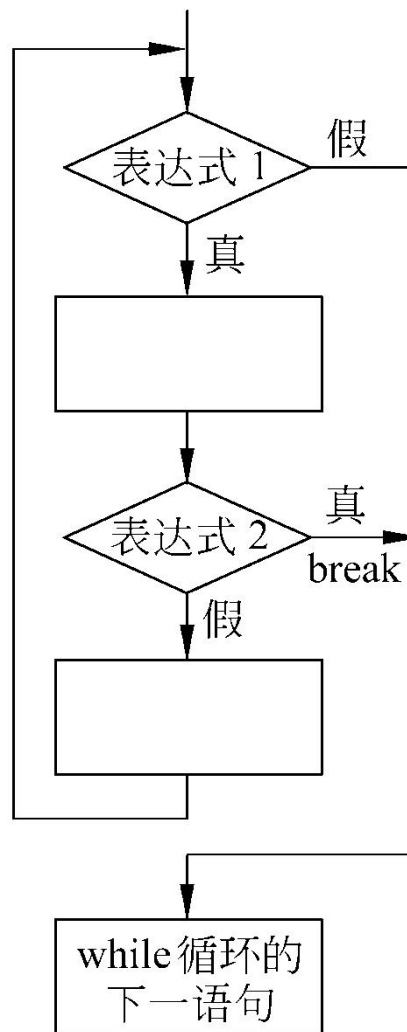
一个循环体内又包含另一个完整的循环结构，称为循环的嵌套。内嵌的循环中还可以嵌套循环，这就是多层循环。

3种循环(**while**循环、**do while**循环和**for**循环)可以互相嵌套。

3.8.6 break语句和continue语句



结束本次循环



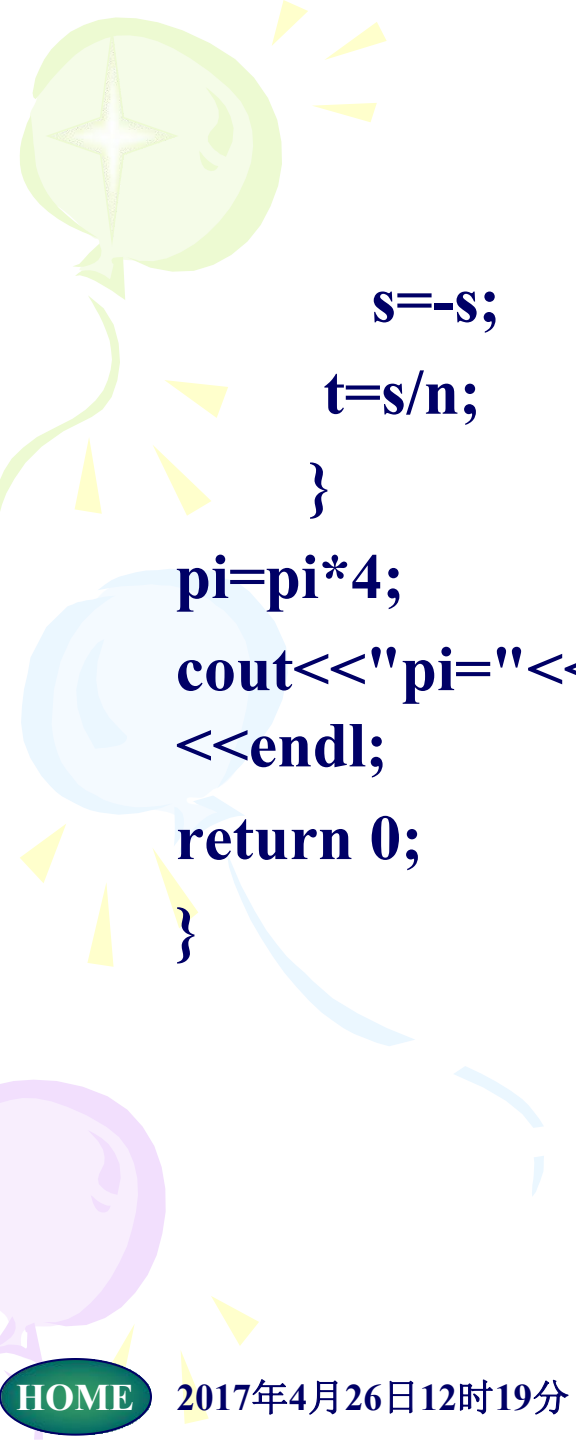
终止整个循环

3.8.7 编写循环结构的程序

例3.12 用下面公式求 π 的近似值。 $\pi/4 \approx 1 - 1/3 + 1/5 - 1/7 + \dots$
直到最后一项的绝对值小于 10^{-7} 为止。

根据给定的算法很容易编写程序如下：

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main( )
{ int s=1;
  double n=1,t=1,pi=0;
  while((fabs(t))>1e-7)
  { pi=pi+t;
    n=n+2;
```



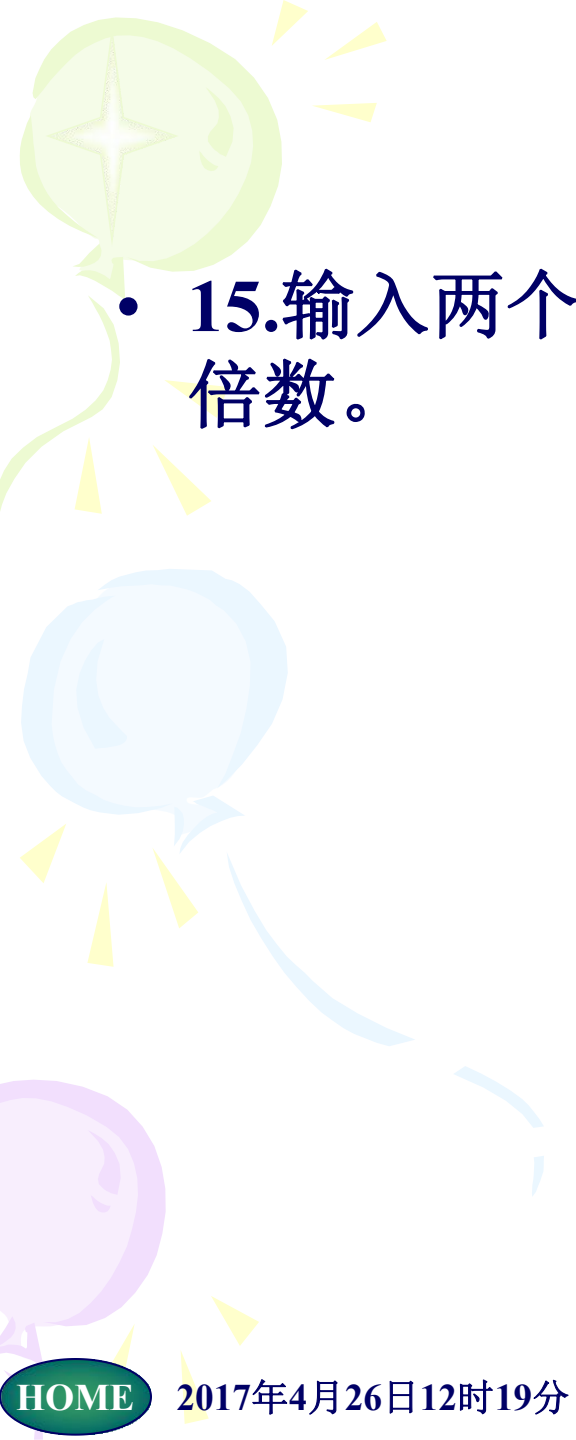
```
s=-s;  
t=s/n;  
}  
pi=pi*4;  
cout<<"pi="<<setiosflags(ios:: fixed)<<setprecision(6)<<pi  
<<endl;  
return 0;  
}
```

字符直角三角形

- `#include<iostream>`
- `using namespace std;`
- `int main()`
- `{`
- `for(int i=1; i<=10; ++i)`
- `{`
- `for(int j=1; j<=i; ++j)`
- `cout<<"M";`
- `cout<<endl;`
- `}`
- `return 0;`
- `}`

字符倒三角形

```
• #include<iostream>
• using namespace std;
• int main()
• {
•     for(int i=1; i<=10; ++i)
•     {
•         for(int j=1; j<=i-1; ++j)
•             cout<<" ";
•         for(int k=1; k<=21-2*i; ++k)
•             cout<<"M";
•         cout<<endl;
•     }
•     return 0;
• }
```

- 
- 15.输入两个整数，求其最大公约数 m 和 n 最小公倍数。

		最大公约数	最小公倍数
联系	组成	都是由质因数确定的	
	求法	都用短除法的形式进行分解，直到两个商是互质数为止	
	关系	最小公倍数是最大公约数的倍数	
区别	来源	源于公约数。公约数的个数是有限的，最小的公约数是1。它是其中最大的。	源于公倍数。公倍数的个数是无限的，没有最大的公倍数。它是其中是小的。
	组成	公有质因数的乘积	公有质因数与各自独有质因数和乘积
	求法	把所有的除数乘起来	把除数和商全乘起来
	大小	小于或等于较小的一个数	大于或等于较大的一个数



	求两个数的最大公约数	求两个数的最小公倍数
相同点	用短除的形式分解质因数,直到两个商是互质数为止	同左
不同点	把所有的除数乘起来	把所有的除数和商乘起来

最大公约数求法

- 辗转相除法

如果两个数比较大，而且都很难分解质因数，就可以用大数除以小数，再用上次运算中的除数除以余数，如此反复除下去，直到能整除为止，最后一个除数就是两数的最大公约数。例如：求636和376的最大公约数。 $636 \div 376 = 1 \dots 260$ ， $376 \div 260 = 1 \dots 116$ ， $260 \div 116 = 2 \dots 28$ ， $116 \div 28 = 4 \dots 4$ ， $28 \div 4 = 7$ ，所以636和376的最大公约数就是4。

- 求两个整数的最大公约数、最小公倍数

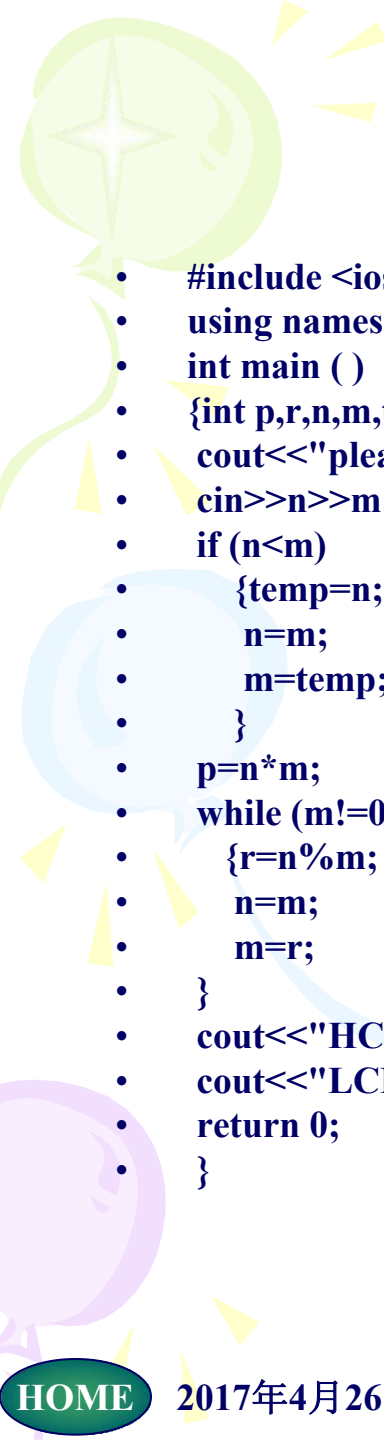
分析：求最大公约数的算法思想：(最小公倍数=两个整数之积/最大公约数)

(1) 对于已知两数 m , n , 使得 $m > n$;

(2) m 除以 n 得余数 r ;

(3) 若 $r=0$, 则 n 为求得的最大公约数, 算法结束; 否则执行(4);

(4) $m \leftarrow n$, $n \leftarrow r$, 再重复执行(2)。



```

• #include <iostream>
• using namespace std;
• int main ()
• {int p,r,n,m,temp;
•   cout<<"please enter two positive integer numbers n,m:";
•   cin>>n>>m;
•   if (n<m)
•   {temp=n;
•     n=m;
•     m=temp;           //把大数放在n中, 小数放在m中
•   }
•   p=n*m;              //先将n和m的乘积保存在p中, 以便求最小公倍数时用
•   while (m!=0)        //求n和m的最大公约数
•   {r=n%m;
•     n=m;
•     m=r;
•   }
•   cout<<"HCF="<<n<<endl;  //Highest common factor, Greatest common divisor
•   cout<<"LCD="<<p/n<<endl;  // Least/lowest common denominator , Least common multiple
•   return 0;
• }

```

例3.13 求Fibonacci数列前40个数。这个数列有如下特点：第1、2个数为1、1。从第3个数开始，每个数是其前面两个数之和。即

$$F1=1 \quad (n=1)$$

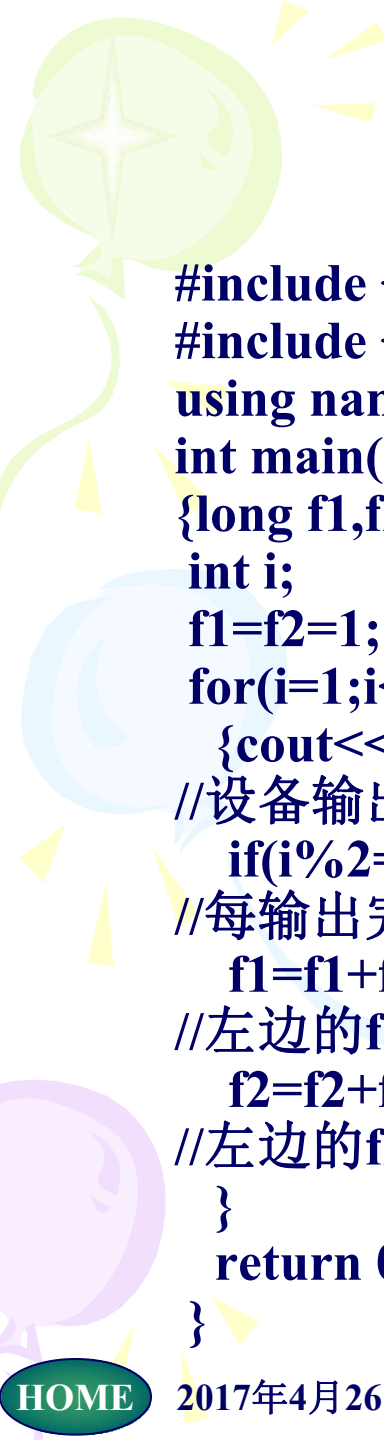
$$F2=1 \quad (n=2)$$

$$Fn=Fn-1+Fn-2 \quad (n \geq 3)$$

这是一个有趣的古典数学问题：有一对兔子，从出生后第3个月起每个月都生一对兔子。小兔子长到第3个月后每个月又生一对兔子。假设所有兔子都不死，问每个月的兔子总数为多少？

可以从书中表3.4看出兔子数的规律。

根据给出的每月兔子总数的关系，可编写程序如下：



```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{long f1,f2;
 int i;
 f1=f2=1;
 for(i=1;i<=20;i++)
 {cout<<setw(12)<<f1<<setw(12)<<f2;
 //设备输出字段宽度为12，每次输出两个数
 if(i%2==0) cout<<endl;
 //每输出完4个数后换行，使每行输出4个数
 f1=f1+f2;
 //左边的f1代表第3个数，是第1、2个数之和
 f2=f2+f1;
 //左边的f2代表第4个数，是第2、3个数之和
 }
 return 0;
}
```

例3.14 找出100~200间的全部素数。

$m=i \times j$ 假定 $i \leq j$,

则 $i^2 \leq i \times j = m \leq j^2$

即 $i^2 \leq m \leq j^2$

即 $i \leq \sqrt{m} \leq j$

编写程序如下:

```
#include <iostream>
```

```
#include <cmath>
```

```
#include <iomanip>
```

```
using namespace std;
```

```
int main()
```

```
{int m,k,i,n=0;
```

```
    bool prime;    //定义布尔变量prime
```

for(m=101;m<=200;m=m+2) //判别m是否为素数，m由101变化到200，增量为2

{prime=true; //循环开始时设prime为真，即先认为m为素数

k=int(sqrt(m)); //用k代表根号m的整数部分

for(i=2;i<=k;i++) //此循环作用是将m被2~根号m除，检查是否能整除

if(m%i==0) //如果能整除，表示m不是素数

{ prime=false; //使prime变为假

break; //终止执行本循环

}

if (prime) //如果m为素数

{cout<<setw(5)<<m; //输出素数m，字段宽度为5

n=n+1; //n用来累计输出素数的个数

}

if(n%10==0) cout<<endl; //输出10个数后换行

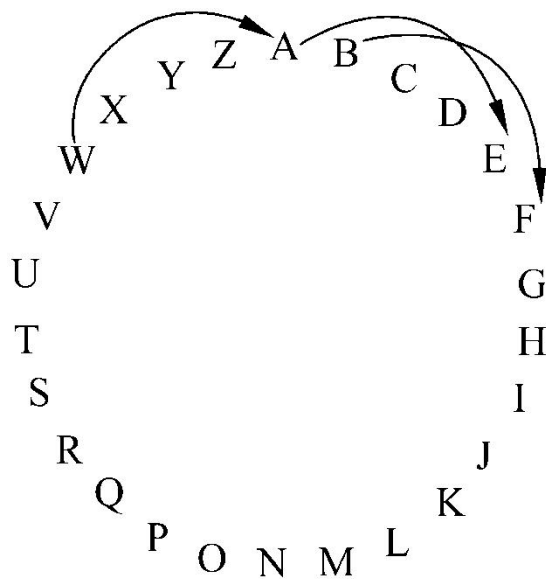
}

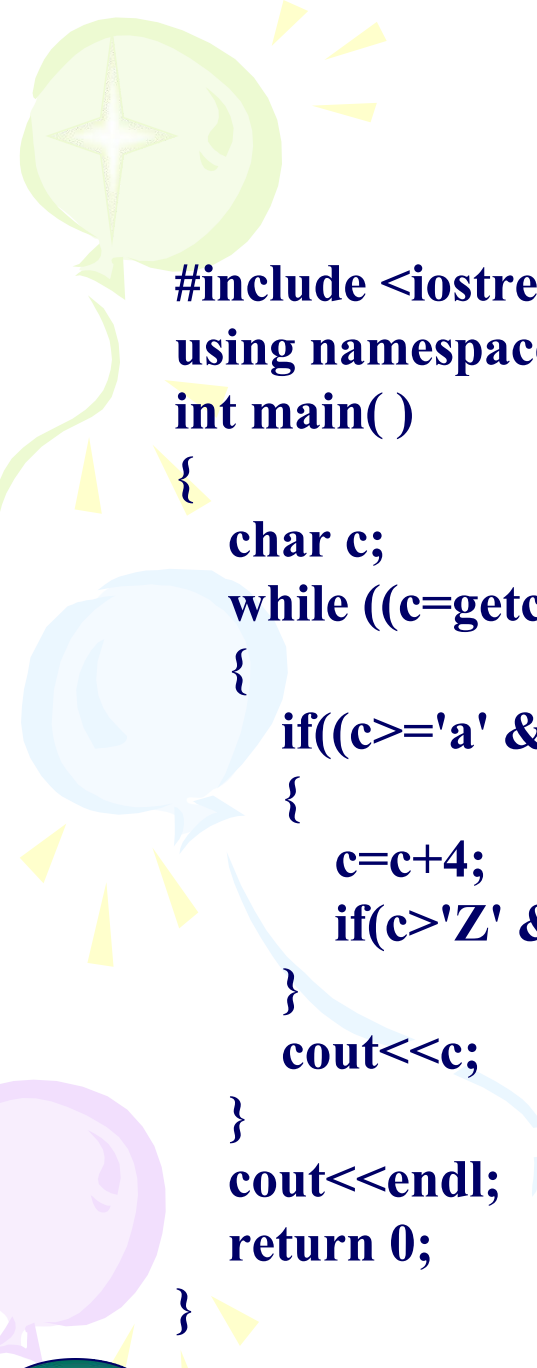
cout<<endl; //最后执行一次换行

return 0;

}

例3.15 译密码。为使电文保密，往往按一定规律将电文转换成密码，收报人再按约定的规律将其译回原文。例如，可以按以下规律将电文变成密码：将字母A变成字母E，a变成e，即变成其后的第4个字母，W变成A，X变成B，Y变成C，Z变成D。见图3.20。字母按上述规律转换，非字母字符不变。如"Wonderful!"转换为"Asrhivjyp!"。输入一行字符，要求输出其相应的密码。





```
#include <iostream>  
using namespace std;  
int main( )  
{  
    char c;  
    while ((c=getchar( ))!='\n')  
    {  
        if((c>='a' && c<='z') || (c>='A' && c<='Z'))  
        {  
            c=c+4;  
            if(c>'Z' && c<='Z'+4 || c>'z') c=c-26;  
        }  
        cout<<c;  
    }  
    cout<<endl;  
    return 0;  
}
```

while sentence

- Syntax Form

`while (expression) sentence`

- Executing Order

Firstly the expression's value is judged, if it is true, then execute the next sentence

do-while sentence

- Syntax Form

`do {sentence} while (expression)`

- Executing Order

Firstly executing the sentence once, then judge the expression's value is true or false.

for 语句

● Syntax Form:

for(exp1; exp2; exp3) sentence

● break and continue sentence

◆ break

Making program exit from the whole cycle, and proceed to execute the following sentence.

◆ continue

Making program exit from the current cycle, and proceed to execute the next cycle.

作 业

- 课本习题
- P85 第3章
- 12、13、16、17
- 18、19、20、21

习题18

```
• #include <iostream>
• using namespace std;
• int main ( )
• {
•     float s=0,t=1;
•     int n;
•     for (n=1;n<=20;n++)
•     {
•         t=t*n;    // 求n!
•         s=s+t;    // 将各项累加
•     }
•     cout<<"1!+2!+...+20!="<<s<<endl;
•     return 0;
• }
```

习题19

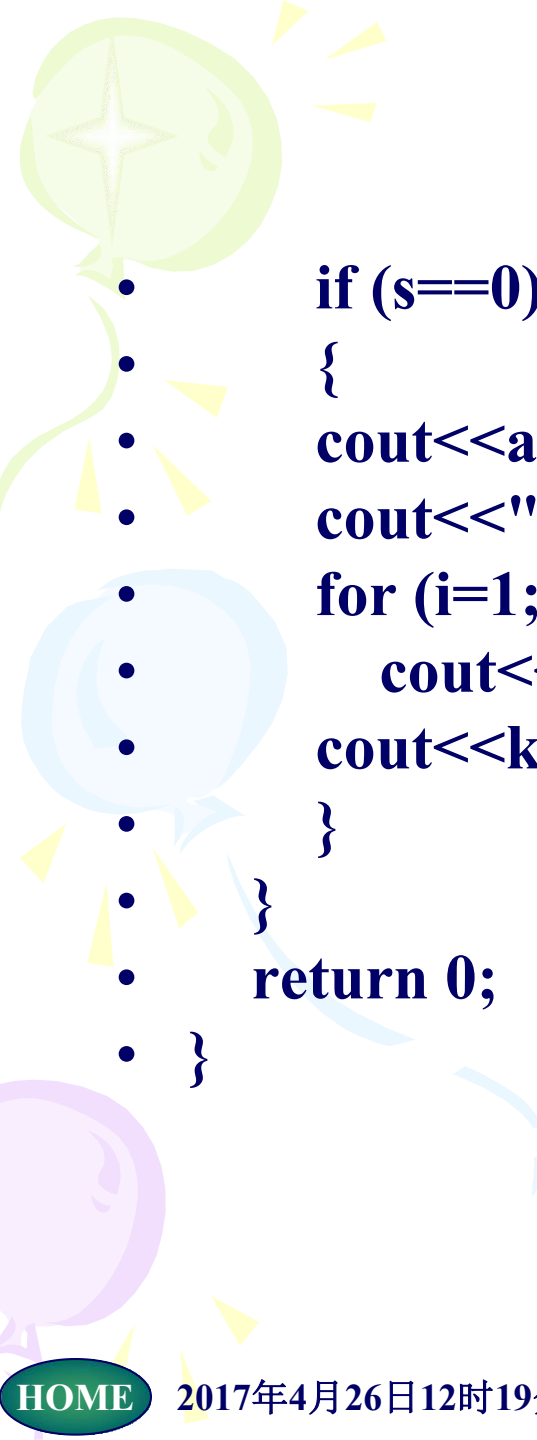
```
• #include <iostream>
• using namespace std;
• int main ( )
• {
•     int i,j,k,n;
•     cout<<"narcissus numbers are:"<<endl;
•     for (n=100;n<1000;n++)
•     {
•         i=n/100;
•         j=n/10-i*10;
•         k=n%10;
•         if (n == i*i*i + j*j*j + k*k*k)
•             cout<<n<<" ";
•     }
•     cout<<endl;
•     return 0;
• }
```

习题20

```
• #include <iostream>
• using namespace std;
• int main( )
• {
•     int m,s,i;
•     for (m=2;m<1000;m++)
•     {
•         s=0;
•         for (i=1;i<m;i++)
•             if ((m%i)==0) s=s+i;
•         if(s==m)
•         {
•             cout<<m<<" is a完数"<<endl;
•             cout<<"its factors are:";
•             for (i=1;i<m;i++)
•                 if (m%i==0) cout<<i<<" ";
•             cout<<endl;
•         }
•     }
•     return 0;
• }
```

习题20-1

```
• #include <iostream>
• using namespace std;
• int main( )
• {
•     int k[11];
•     int i,a,n,s;
•     for (a=2;a<=1000;a++)
•     {
•         n=0;
•         s=a;
•         for (i=1;i<a;i++)
•             if ((a%i)==0)
•             {
•                 n++;
•                 s=s-i;
•                 k[n]=i;           // 将找到的因子赋给k[1]---k[10]
•             }
•     }
```



```
• if (s==0)
• {
• cout<<a<<" is a 完数"<<endl;
• cout<<"its factors are:";
• for (i=1;i<n;i++)
•     cout<<k[i]<<" ";
• cout<<k[n]<<endl;
• }
• }
• return 0;
• }
```

习题22

```
• #include <iostream>
• using namespace std;
• int main( )
• {
•     int day,x1,x2;
•     day=9;
•     x2=1;
•     while(day>0)
•     {
•         x1=(x2+1)*2;    // 第1天的桃子数是第2天桃子数加1后的2倍
•         x2=x1;
•         day--;
•     }
•     cout<<"total="<<x1<<endl;
•     return 0;
• }
```

习题23

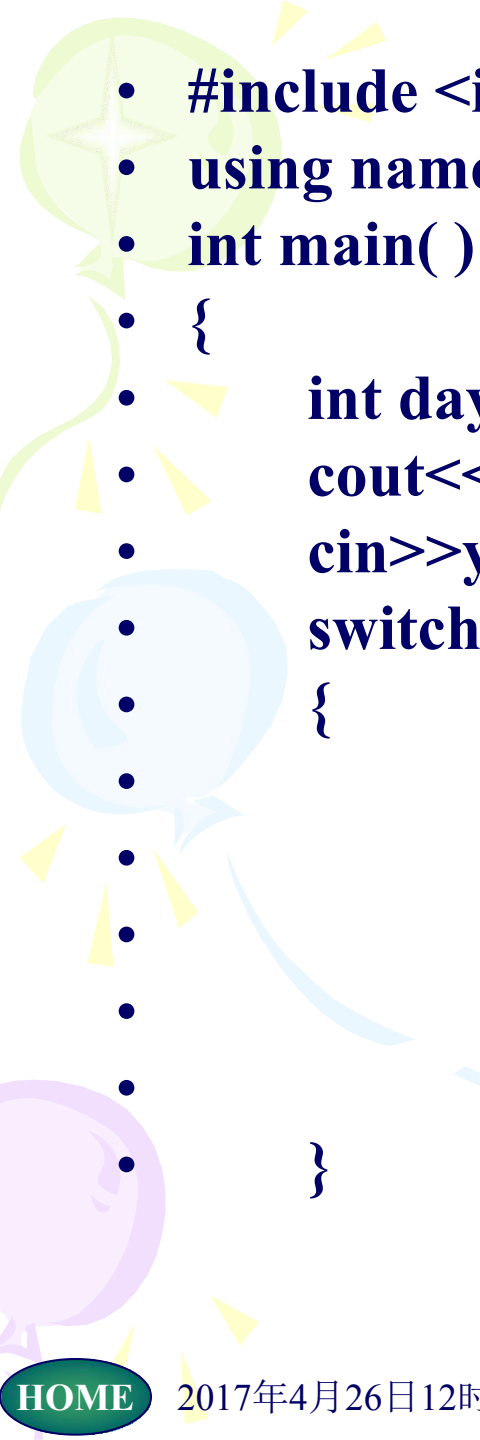
```
• #include <iostream>
• #include <cmath>
• using namespace std;
• int main()
• {
•     float a,x0,x1;
•     cout<<"enter a positive number:";
•     cin>>a;      // 输入a的值
•     x0=a/2;
•     x1=(x0+a/x0)/2;
•     do
•     {
•         x0=x1;
•         x1=(x0+a/x0)/2;
•     }
•     while(fabs(x0-x1)>=1e-5);
•     cout<<"The square root of "<<a<<" is "<<x1<<endl;
•     return 0;
• }
```

课堂练习

1. 输入某年某月某日，判断这一天是这一年的第几天？
2. 一个5位数，判断它是不是回文数。即12321是对称（回文）数，个位与万位相同，十位与千位相同。
3. 有1、2、3、4个数字，能组成多少个互不相同且无重复数字的三位数？都是多少？
4. 一个整数，它加上100后是一个完全平方数，再加上168又是一个完全平方数，请问该数是多少？
5. 求 $s=a+aa+aaa+aaaa+aa...a$ 的值，其中a是一个数字。例如 $2+22+222+2222+22222$ (此时共有5个数相加)，几个数相加由键盘控制。
6. 顺序英文字符三角形

课堂练习

- 题目1：输入某年某月某日，判断这一天是这一年的第几天？
- 1.程序分析：以3月5日为例，应该先把前两个月的加起来，然后再加上5天即本年的第几天，特殊情况，闰年且输入月份大于3时需考虑多加一天。
- 2.程序源代码：



```
• #include <iostream>
• using namespace std;
• int main( )
• {
•     int day,month,year,sum,leap;
•     cout<< "\nplease input year,month,day\n";
•     cin>>year>>month>>day;
•     switch(month)/*先计算某月以前月份的总天数*/
•     {
•         case 1:sum=0;break;
•         case 2:sum=31;break;
•         .....
•         case 12:sum=334;break;
•         default:cout<<"data error";break;
•     }
```

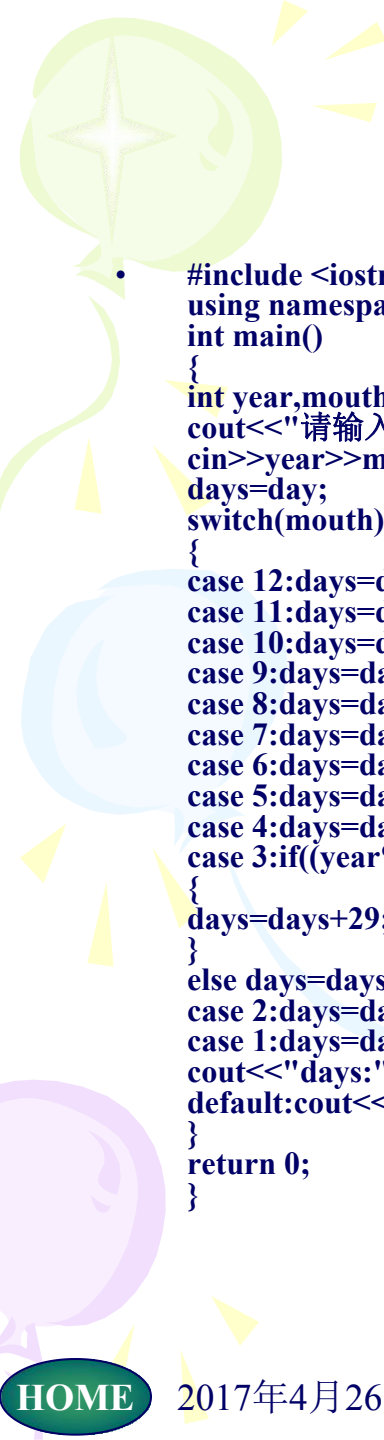
```

• #include <iostream>
• using namespace std;
• int main( )
• {
•     int day,month,year,sum,leap;
•     cout<< "\nplease input year,month,day\n";
•     cin>>year>>month>>day;
•     switch(month)/*先计算某月以前月份的总天数*/
•     {
•         case 1:sum=0;break;
•         case 2:sum=31;break;
•         case 3:sum=59;break;
•         case 4:sum=90;break;
•         case 5:sum=120;break;
•         case 6:sum=151;break;
•         case 7:sum=181;break;
•         case 8:sum=212;break;
•         case 9:sum=243;break;
•         case 10:sum=273;break;
•         case 11:sum=304;break;
•         case 12:sum=334;break;
•         default:cout<<"data error";break;
•     }

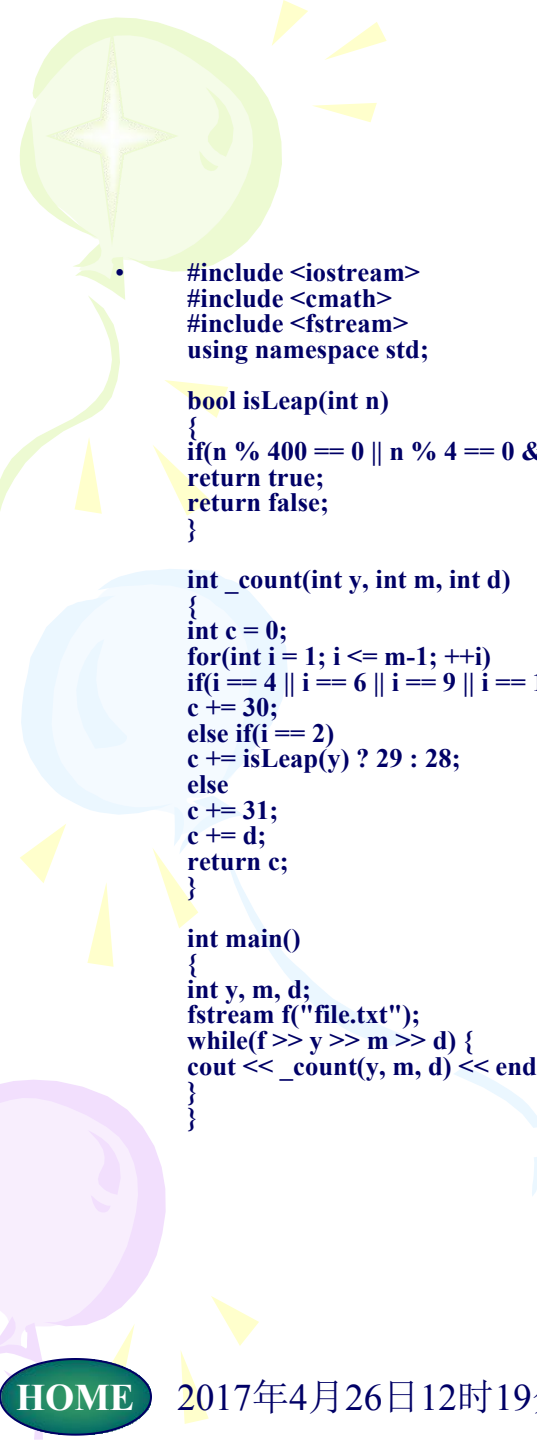
```



```
•    sum=sum+day; /*再加上某天的天数*/
•    if(year%400==0||(year%4==0&&year%100!=0))
/*判断是不是闰年*/
•        leap=1;
•    else
•        leap=0;
•    if(leap==1&&month>2)
/*如果是闰年且月份大于2,总天数应该加一天*/
•        sum++;
•        cout<<"It is the "<<sum<<"th day.\n";
•        return 0;
•    }
```



```
• #include <iostream>
  using namespace std;
  int main()
  {
    int year,mouth,day,days;
    cout<<"请输入年,月,日: ";
    cin>>year>>mouth>>day;
    days=day;
    switch(mouth)
    {
      case 12:days=days+30;
      case 11:days=days+31;
      case 10:days=days+30;
      case 9:days=days+31;
      case 8:days=days+31;
      case 7:days=days+30;
      case 6:days=days+31;
      case 5:days=days+30;
      case 4:days=days+31;
      case 3:if((year%4==0 && year%100!=0)|| year%400==0)
      {
        days=days+29;
      }
      else days=days+28;
      case 2:days=days+31;
      case 1:days=days;
      cout<<"days:"<<days<<endl;break;
      default:cout<<"error!"<<endl;
    }
    return 0;
  }
```



```
• #include <iostream>
  #include <cmath>
  #include <fstream>
  using namespace std;

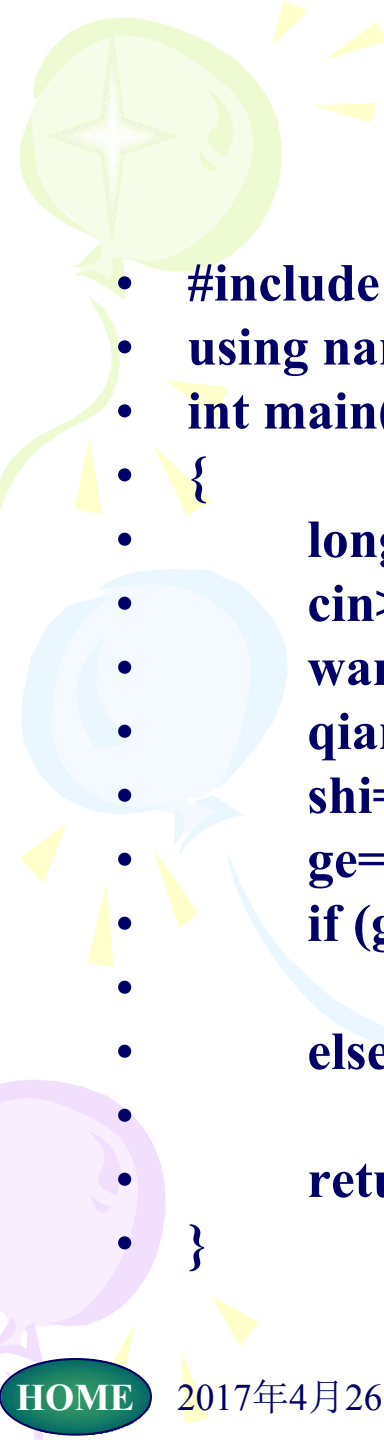
  bool isLeap(int n)
  {
    if(n % 400 == 0 || n % 4 == 0 && n % 100 != 0)
      return true;
    return false;
  }

  int _count(int y, int m, int d)
  {
    int c = 0;
    for(int i = 1; i <= m-1; ++i)
      if(i == 4 || i == 6 || i == 9 || i == 11)
        c += 30;
      else if(i == 2)
        c += isLeap(y) ? 29 : 28;
      else
        c += 31;
    c += d;
    return c;
  }

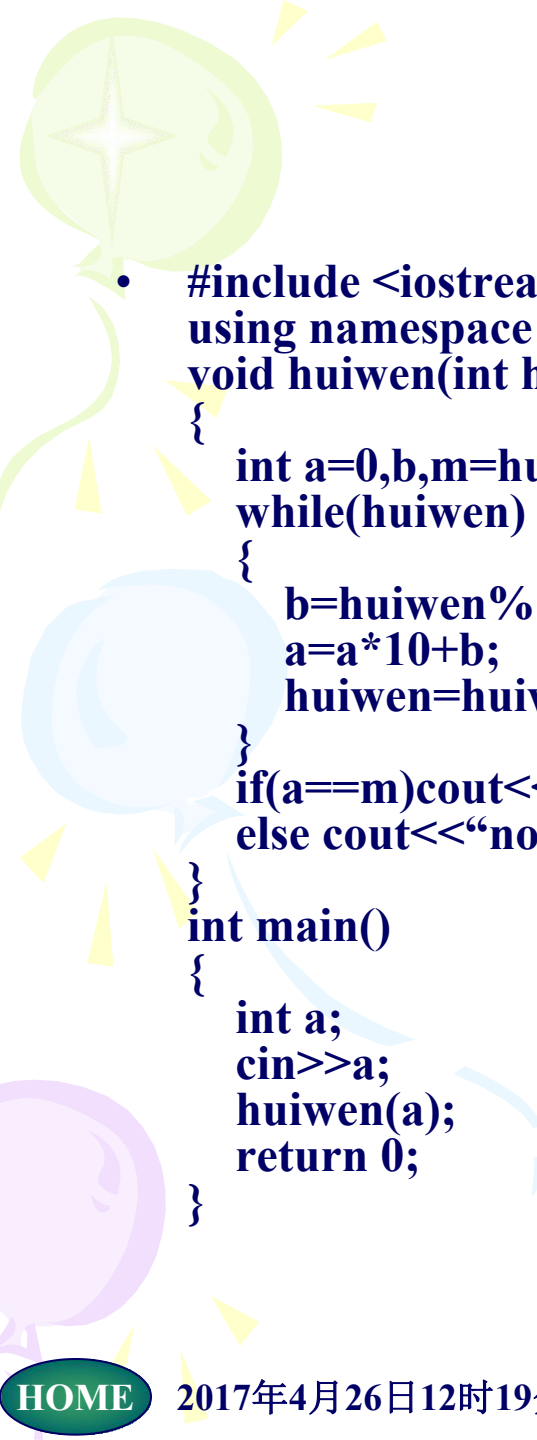
  int main()
  {
    int y, m, d;
    fstream f("file.txt");
    while(f >> y >> m >> d) {
      cout << _count(y, m, d) << endl;
    }
  }
```

课堂练习

- 题目2：一个5位数，判断它是不是回文数。即12321是对称（回文）数，个位与万位相同，十位与千位相同。
 - 1.程序分析：学会分解出每一位数。
 - 2.程序源代码：



```
• #include <iostream>
• using namespace std;
• int main( )
• {
•     long ge,shi,qian,wan,x;
•     cin>>x;
•     wan=x/10000;
•     qian=x%10000/1000;
•     shi=x%100/10;
•     ge=x%10;
•     if (ge==wan&&shi==qian)/*个位等于万位并且十位等于千位*/
•         cout<<"this number is a huiwen\n";
•     else
•         cout<<"this number is not a huiwen\n";
•     return 0;
• }
```



- ```
#include <iostream>
using namespace std;
void huiwen(int huiwen) //判断回文数的函数
{
 int a=0,b,m=huiwen;
 while(huiwen)
 {
 b=huiwen%10;
 a=a*10+b;
 huiwen=huiwen/10;
 }
 if(a==m)cout<<"Is hui wen shu"<<endl;
 else cout<<"not hui wen"<<endl;
}
int main()
{
 int a;
 cin>>a;
 huiwen(a);
 return 0;
}
```

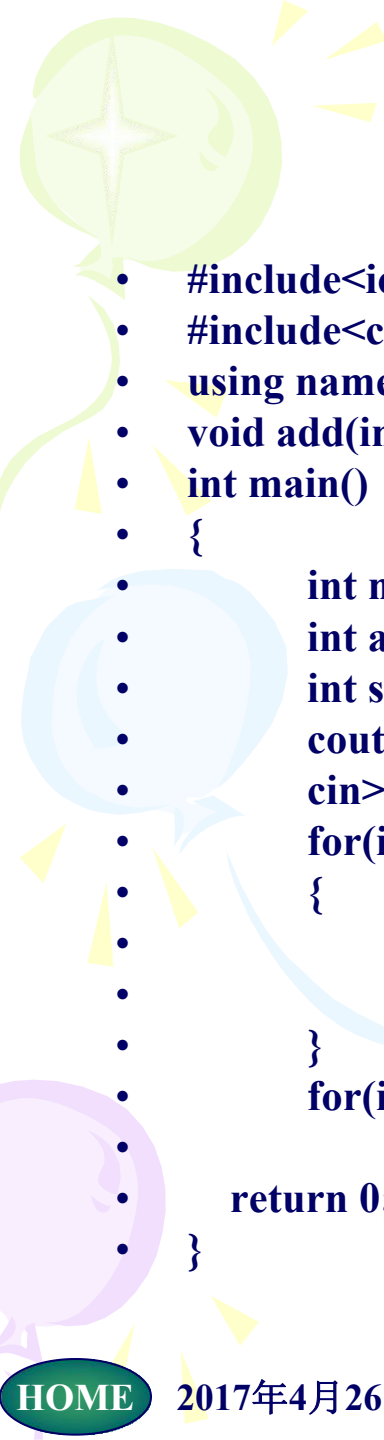
3.有1、2、3、4个数字，能组成多少个互不相同且无重复数字的三位数？都是多少？

```
#include <iostream>
using namespace std;
int main()
{
 int i,j,k;
 int n=0;
 for (i=1;i<=4;i++)
 for (j=1;j<=4;j++)
 for (k=1;k<=4;k++)
 {
 n++;
 cout <<i<<j<<k<<endl;
 }
 cout <<"共有"<<n<<"个"<<endl;
 return 0;
}
```

# 课堂练习

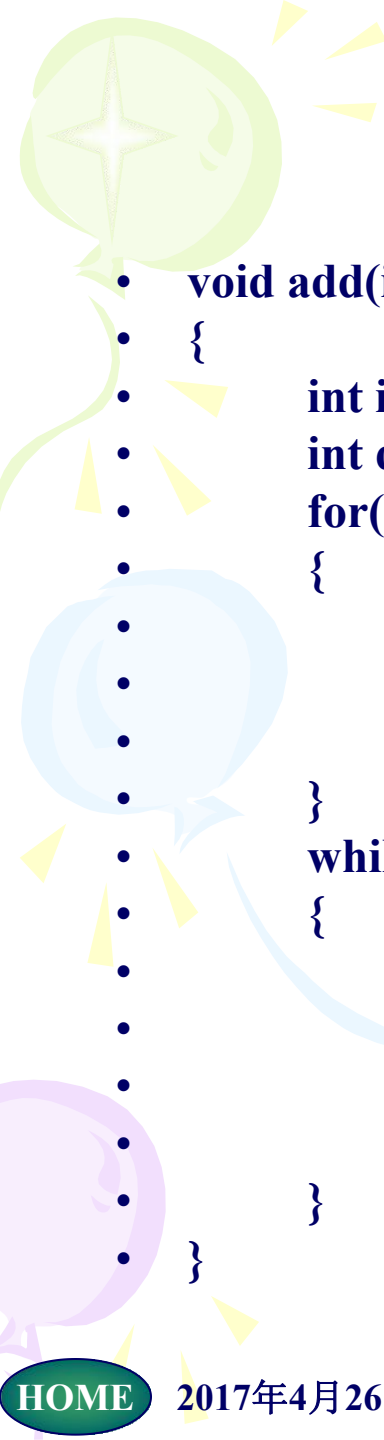
$$s=a+aa+aaa+aaaa+aa...a$$

```
• #include <iostream>
• using namespace std;
• int main ()
• {
• int a,n,i=1,sn=0,tn=0;
• cout<<"a,n=";
• cin>>a>>n;
• while (i<=n)
• {
• tn=tn+a; //赋值后的tn为i个a组成数的值
• sn=sn+tn; //赋值后的sn为多项式前i项之和
• a=a*10;
• ++i;
• cout<<"a+aa+aaa+...="<<sn<<endl;
• }
• return 0;
• }
```



# $s=a+aa+aaa+aaaa+aa...a$

```
• #include<iostream>
• #include<cmath>
• using namespace std;
• void add(int a[],int s[]);
• int main()
• {
• int n, i;
• int a[256]={0};
• int s[256]={0};
• cout<<"请输入N\n ";
• cin>>n;
• for(i=0;i<n;i++)
• {
• a[i] = 2;
• add(a,s);
• }
• for(i=n;i>=0;i--)
• cout<<s[i];
• return 0;
• }
```



# **$s = a + aa + aaa + aaaa + aa...a$**

```
• void add(int a[],int s[])
• {
• int i;
• int d = 0;
• for(i=0;a[i]!=0;i++)
• {
• s[i]=s[i]+a[i]+d;
• d=s[i]/10;
• s[i]=s[i]%10;
• }
• while(d!=0)
• {
• s[i]=s[i]+d;
• d=s[i]/10;
• s[i]=s[i]%10;
• i=i+1;
• }
• }
```

# 课堂练习

## 顺序英文字符三角形

- **#include<iostream>**
- **using namespace std;**
- **int main()**
- **{**
- **for(int i=1; i<=10; ++i)**
- **{**
- **for(int j=1; j<=10-i; ++j)**
- **cout<<" ";**
- **for(char ch='A'; ch<'A'+2\*i-1; ++ch)**
- **cout<<ch;**
- **cout<<"\n";**
- **}**
- **return 0;**
- **}**

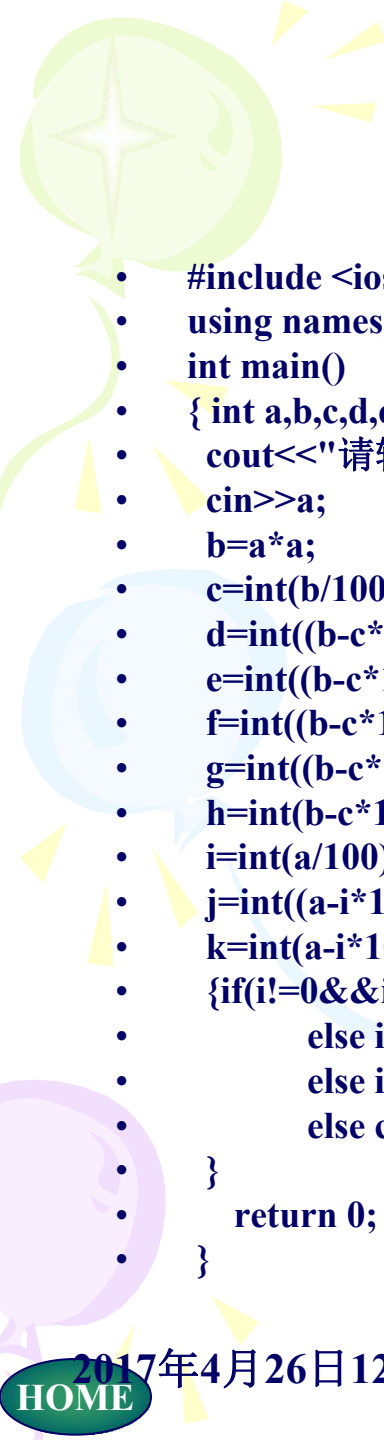
# 顺序英文字符三角形

```
• #include <iostream>
• using namespace std;
• int main()
• {
• int i,j;
• for (int i=1;i<=10;i++)
• {
• for (j=10-i;j>0;j--)
• cout<<' ';
• for (j=1;j<=2*i-1;j++)
• putchar('A'+j-1);
• cout<<endl;
• }
• return 0;
• }
```

# 课堂练习

## 自守数

```
• #include<iostream>
• #include<cmath>
• using namespace std;
• int main()
• {
• int chengfang(int);
• int a,b=0,c,f;
• cin>>a;
• do
• {
• b=b+1;
• for(c=1;b==chengfang(c);c++);
• }while (b<=a);
• f=a*a-a;
• if((f%chengfang(c))==0)
• cout<<a<<" YES "<<"a*a="<<a*a<<endl;
• else
• cout<<a<<" NOT "<<"a*a="<<a*a<<endl;
• return 0;
• }
• int chengfang(int c)
• {
• int i,s=1;
• for (i=1;i<=c;i++)
• s=s*i;
• return s;
• }
```



```

• #include <iostream>
• using namespace std;
• int main()
• { int a,b,c,d,e,f,g,h,i,j,k;
• cout<<"请输入一个0--999的自然数: ";
• cin>>a;
• b=a*a;
• c=int(b/100000); //b的十万位
• d=int((b-c*100000)/10000); //b的万位
• e=int((b-c*100000-d*10000)/1000); //b的千位
• f=int((b-c*100000-d*10000-e*1000)/100); //b的百位
• g=int((b-c*100000-d*10000-e*1000-f*100)/10); //b的十位
• h=int(b-c*100000-d*10000-e*1000-f*100-g*10); //b的个位
• i=int(a/100); //a的百位
• j=int((a-i*100)/10); //a的十位
• k=int(a-i*100-j*10); //a的个位
• {if(i!=0&&i!=f) cout<<a<<"no " <<a<<"*" <<a<<"=" <<b<<endl;
• else if(j!=0&&j!=g) cout<<a<<"no " <<a<<"*" <<a<<"=" <<b<<endl;
• else if(k!=0&&k!=h) cout<<a<<"no " <<a<<"*" <<a<<"=" <<b;
• else cout<<a<<"yes" <<a<<"*" <<a<<"=" <<b<<endl;
• }
• }
• return 0;
• }

```

- **#include<iostream>**
- **using namespace std;**
- **int main()**
- **{int a,b;**
- **cout<<"请输入一个自然数:";**
- **cin>>a;**
- **b=a\*a;**
- **if(b<1||b>9999)**
- **{cout<<"数据输入有误！请重新再输:";**
- **}**
- **else if((b%10==a||b%100==a))**
- **{cout<<a<<' '<<"yes"<<' '<<a<<"\*"<<a<<"="<<b;**
- **}**
- **else**
- **cout<<a<<' '<<"no"<<' '<<a<<"\*"<<a<<"="<<b;**
- **cout<<endl;**
- **return 0;**
- **}**

```

• #include<iostream>
• using namespace std;
• int main()
• {
• int n,a,b,c;
• bool flag=false;
• cout<<"请输入一个自然数:";
• cin>>a;
• b=a*a;
• c=b;
• while(a<1||b>46340)
• {
• cout<<"数据输入有误！请重新再输:";
• cin>>a;
• b=a*a;
• }
• n=1;
• while(c=c/10){n++;}
• cout<<b<<" is "<<n<<" bit number"<<endl;
• c=1;
• for (int i=1;i<n;i++)
• {
• c=c*10;
• if(b%c==a) flag=true;
• else ;
• }
• if(flag)
• cout<<a<<' '<<"yes"<<' '<<a<<"*"<<a<<"="<<b;
• else
• cout<<a<<' '<<"no"<<' '<<a<<"*"<<a<<"="<<b;
• cout<<endl;
• return 0;
• }

```

```

• #include <iostream>
• #include <string>
• #include <ctime>
• #include <fstream>
• using namespace std;

• class Self
• {
• private:
• int a[20];
• int m,n,p;
• public:
• Self(int _m,int _n)
• {
• m=_m;n=_n;
• }
• void process()
• {
• int pf;
• int j=0;
• p=0;
• for(int i=m;i<=n;i++)
• {
• int k=i;
• pf=k*k;
• while(k!=0)
• {
• if((pf%10)!=(k%10))break;
• k=k/10;
• pf=pf/10;
• }
• if(k==0)
• {
• a[j]=i;
• p++;
• j++;
• }
• }
• }
• void print()
• {
• cout<<"自守数的个数: 阢"<<"\n";
• for(int j=0;j<p;j++)
• cout<<a[j]<<"\t";
• cout<<endl;
• }

```