# Ch-03

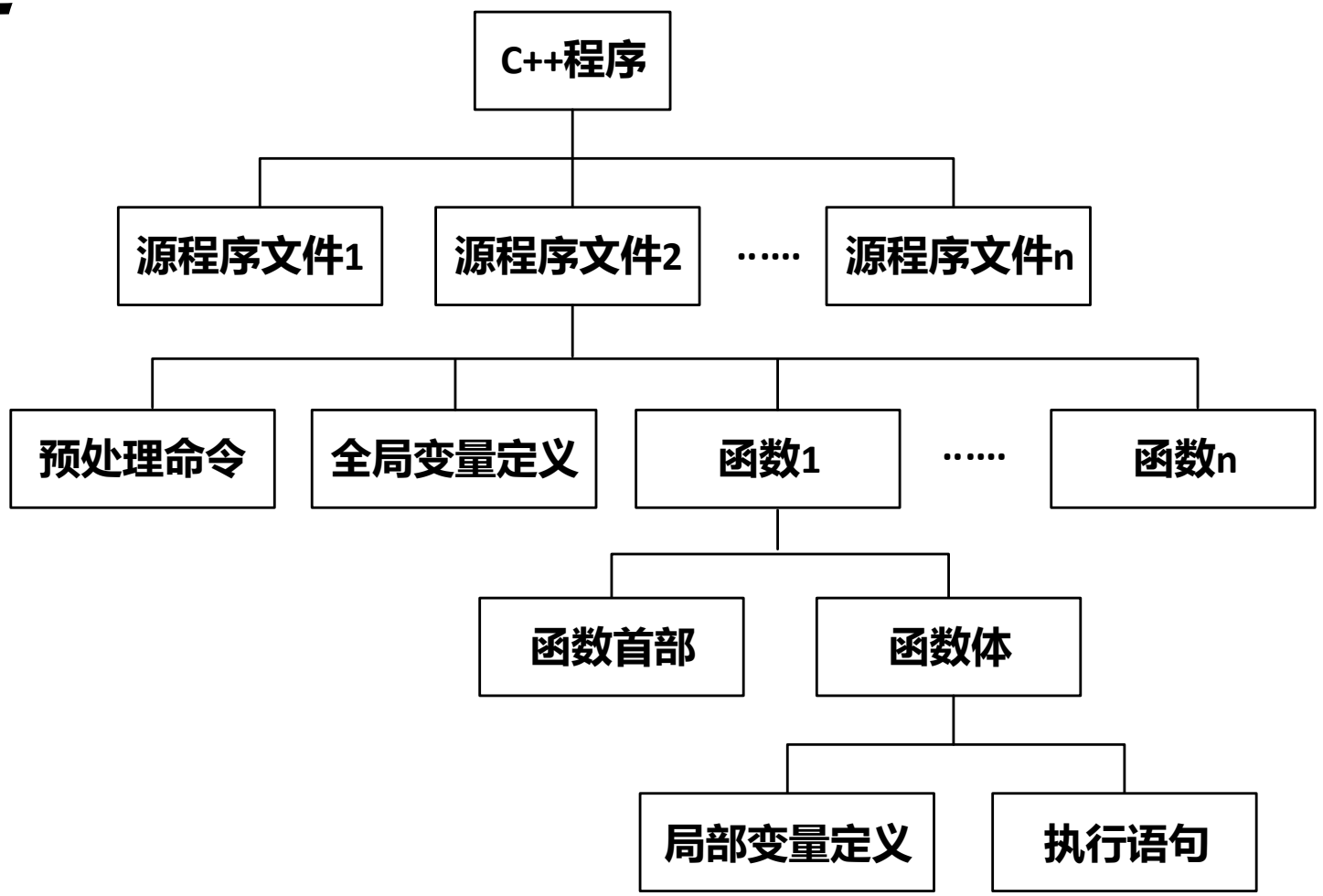
# 程序的三种基本结构

# 主要内容：

- 顺序结构
- 选择结构
- 循环结构

# 3.1 C++语句概述

- C++程序的结构
- C++语句的类型
  - 控制语句
  - 表达式语句
  - 函数调用语句
  - 复合语句
  - 空语句



- 控制语句：用于完成一定的控制功能，C只有9种控制语句：

| | | |
|---|---|---|
| if( ) … else … | for( ) … | while( ) … |
| do … while( ) | continue; | break; |
| switch( ) … | goto; | return; |

- 表达式语句：由表达式加分号构成。
- 函数调用语句：由函数调用加分号构成。
- 复合语句：可以用{ }把一条或多条语句括起来构成复合语句。
- 空语句：只由一个分号构成的语句，即 ；。

◆ 注意，除了复合语句以为的其他语句必须用分号结尾。

# 3.2 顺序结构

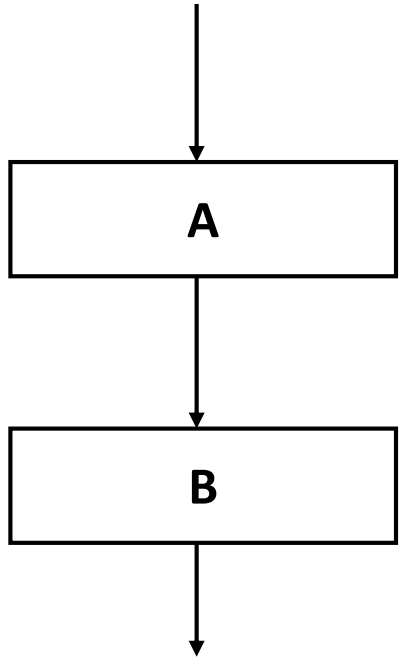
**按照语句出现的前后顺序，线性地依次执行程序中的每一条语句，而且每条语句仅执行一次。**

## 3.2.1 顺序结构举例

**例3.1**

```
/*
 * 输入三角形边长，求面积
 */

#include <cmath>
#include <iostream>

using namespace std;

int main()
{
    float a, b, c, s, area;

    cin >> a >> b >> c;

    s = 1.0/2*(a+b+c);
    area = sqrt(s*(s-a)*(s-b)*(s-c));

    cout << "a=" << a << " b=" << b << " c=" << c << " s=" << s << endl;
    cout << "area=" << area << endl;

    return 0;
}
```

```
3 4.5 7.3
a=3 b=4.5 c=7.3 s=7.4
area=3.07285
```



# 3.2.2 C++的输入输出

- C++语言本身不提供输入输出语句，输入和输出操作是通过标准库中定义的流对象和运算符实现的。
- 流是对数据从一个对象流动到另一个对象的过程的抽象。
- 预定义的流对象
  - cin：处理标准输入，即键盘输入；
  - cout：处理标准输出，即屏幕输出；
  - cerr：处理标准出错信息，提供不带缓冲区的输出；
  - clog：处理标准出错信息，提供带缓冲区的输出；
- 预定义的流运算符
  - << ：流插入运算符
  - >> ：流提取运算符

## 3.2.2.1 C++的标准输入输出

### □ C++的标准输入

cin >> 变量1 >> 变量2 >> … >> 变量n

例如：int a, b; cin >> a >> b;

### □ C++的标准输出

cout << 表达式1 << 表达式2 << … << 表达式n

例如：int c = 3, d = 27; cout << c << d <<endl;

## 口 说明：

1. C++使用流对象和流运算符进行输入输出前，必须在程序开始包含相关的头文件：#include <iostream>

   - 在C++中，若使用标准库中的内容，就要用预处理命令“#include”将相关的“头文件”包含到源文件中。

2. 在缺省的情况下，cin自动过滤输入中的空白符，换言之，cin不能将输入的空白符赋给字符型变量。

3. 若要得到输入中的空白符，则调用cin的成员函数get()：

   cin.get(字符型变量)；

   - cin.get()从输入缓冲区中取出一个字符，并将它赋给字符型变量。而结束输入时的回车符将保留在缓冲区中。

# 3.2.2.2 C++标准输入输出中的控制符

- 1.使用控制符控制输出格式 ,注意需要iomanip头文件

| 控制符 | 作用 |
| --- | --- |
| dec\|hex\|oct | 设置整数的基数为10,16,8 |
| setbase(n) | 设置整数的基数为n(n只能是16，10，8之一) |
| setfill(c) | 设置填充字符c，c可以是字符常量或字符变量 |
| setw(n) | 设置字段宽度为n位。 |
| setprecision(n) | 设置实数的精度为n位。在以一般十进制小数形式输出时，n代表有效数字。在以fixed(固定小数位数)形式和scientific(指数)形式输出时，n为小数位数。默认的流输出数值有效位是6 |
| setiosflags(ios::fixed) | 设置浮点数以固定的小数位数显示。 |
| setiosflags(ios::scientific) | 设置浮点数以科学计数法(即指数形式)显示。 |
| resetiosflags(标志位) | 终止已设置的输出格式状态，在括号中应指定内容。 |

- 2、可以设置的标志位如下表所示：

| 格式标志 | 作用 |
| --- | --- |
| ios::left\|ios::right | 输出数据在本域宽范围内左对齐,右对齐 |
| ios::dec\|ios::oct\|ios::hex | 设置整数的基数为10,8,16 |
| ios::uppercase | 在以科学计数法输出E和十六进制输出字母X时，以大写表示 |
| ios::showpos | 输出正数时，给出"+"号。 |

### 3.2.3 C中的输入输出函数

使用这些函数需要在程序开头 #include <cstdio> 或 #include <stdio.h>

#### 3.2.3.1 字符输入输出函数

**□ 字符输出函数——putchar()**

- 调用形式：

```
putchar(c)
```

- 参数：c可以是单个的字符型常量/变量、0~127之间的十进制整数或表达式。
- 功能：把字符输出到显示器上。
- 返回值：正常，返回字符的ASCII码值；出错，返回EOF(值为-1)。

## □ 字符输入函数——getchar()

- 调用形式：

```
getchar()
```

- 功能：从键盘缓冲区读入一个字符。
- 返回值：读入的字符。
- 说明：
  - getchar()只接受一个字符。若输入多个字符（按回车后才开始接收字符），则多余字符无效。
  - 用getchar()得到的字符可以赋给字符型变量、整型变量或作为表达式的操作数。

# 例3.2 字符输入输出函数的使用

```cpp
#include <cstdio>
using namespace std;

int main()
{
    printf("输出控制字符和转义字符\n");
    putchar('\101');
    putchar('\n');
    putchar('\\');
    putchar('\n');

    int c;
    printf("Enter a character: ");
    c = getchar();
    putchar(c);
    putchar('\n');

    while(getchar() != '\n') continue;

    printf("Enter a capital letter: ");
    putchar(getchar() + 32);

    return 0;
}
```

```
输出控制字符和转义字符
A
\
Enter a character: k
k
Enter a capital letter: Y
y
```

### 3.2.3.2 格式化的数据输入/输出函数

**口 格式化输出函数——printf()**

- **调用形式：**

| printf(格式控制, 输出表列) |
|---|

  - **功能：按照“格式控制”指定的格式向显示器输出数据（输出表列）。**
  - **返回值：正常，返回输出字节数；出错，返回EOF（值为-1）。**
  - **说明：**
    - ✓ **输出表列：要输出的数据（可以没有，多个时以“,”分隔）**
    - ✓ **格式控制：包含两种信息（1）格式说明：%[修饰符]格式字符，用于指定输出格式，将数据转换为指定的格式 输出。如：%d、%f；（2）普通字符或转义序列：原样输出的字符(包括转义字符)。**

- “格式控制”详解

## 1、格式字符说明

| 格式字符 | 含义 |
| --- | --- |
| d, i | 以有符号十进制整数形式输出 |
| u | 以无符号十进制整数形式输出 |
| x, X | 以无符号十六进制整数形式输出 |
| o | 以无符号八进制整数形式输出 |
| c | 输出一个字符 |
| s | 输出一个字符串 |
| f | 输出小数形式的浮点数（默认6位小数） |
| e, E | 输出指数形式的浮点数（小数部分默认6位小数） |
| g, G | 以e和f中宽度较短的格式进行输出，且不输出无意义的0 |
| p | 以无符号十六进制整数形式输出指针 |
| %% | 输出百分号本身 |

## 2、修饰字符说明

| 修饰字符 | 含义 |
| --- | --- |
| m | 指定输出数据的域宽：若数据长度小于m，在指定域宽内右对齐（即左侧补空格）；否则按实际长度输出。 |
| .n | 指定输出位数：1）对浮点数，用于指定小数点后的输出位数；<br>2）对于字符串，用于指定实际的输出位数。 |
| - | 输出数据在指定域宽内左对齐（即右侧补空格）；否则，默认右对齐。 |
| + | 在有符号数前面显示正号。 |
| 0 | 在指定域宽内，输出数据右对齐，左侧空位补0。 |
| # | 指定在八进制或十六进制数前面显示前导字符0或0x/0X。 |
| h | 指定按照短整型（short）输出数据，可用于修饰格式字符d、o、x、u。 |
| l | 指定按照长整型（long）输出数据，可用于修饰格式字符d、o、x、u。 |

## □ 格式化输入函数——scanf()

- **调用形式：**

```
scanf(格式控制, 地址表列)
```

- **功能：按照“格式控制”指定的格式从键盘读入数据，并保存到地址表列中对应地址所指定的内存单元中。**

- **说明：**

  - ✓ **地址表列：若干个地址组成的表列，多个时以“,”分隔，但不可以为空。**
  - ✓ **格式控制：包含两种信息**
  - ➢ **1、格式说明：%[修饰符]格式字符，用于指定输入数据的格式。**
  - ➢ **2、普通字符：包括空白字符和非空白字符。**

- “格式控制”详解

## 1、格式字符说明

| 格式字符 | 含义 |
| --- | --- |
| d, i | 从键盘读入有符号十进制整数 |
| u | 从键盘读入无符号十进制整数 |
| x, X | 从键盘读入无符号十六进制整数 |
| o | 从键盘读入无符号八进制整数 |
| c | 从键盘读入一个字符 |
| s | 从键盘读入一个字符串 |
| f | 从键盘读入浮点数，可以用小数形式或指数形式输入 |
| e, E, g, G | 与f作用相同，e/E、f和g/G可互相替换 |

## 2、修饰字符说明

| 修饰字符 | 含义 |
| --- | --- |
| m | 指定读入数据所占的域宽，必须为正整数。 |
| h | 用于读入短整型（short）数据，可用于修饰格式字符d、o、x、u。 |
| l | 1）用于读长整型（long）数据，可用于修饰格式字符d、o、x、u；<br>2）用于读入double型数据，可用于修饰f、e、g。 |
| * | 空读数据，即读数据但不赋值给变量 |

## 3、普通字符说明

| | |
| --- | --- |
| 空白字符 | 1）空白字符会使scanf()函数在读操作时略去输入中的一个或多个空白字符，直到出现非空白字符为止。<br>2）常用的空白字符是space、tab、newline (换行)等。 |
| 非空白字符 | 一个非空白字符会使scanf()函数在读入时剔除掉与这个非空白字符相同的字符。因此输入时在非空白字符的对应位置上必须输入该非空白字符 |

# 3.3 选择结构

## 3.3.1 if语句

### □ if语句的三种基本形式

- **形式1：**

```
if(表达式) 语句
```

**执行过程: 当表达式值非0（即为真）时, 执行语句；否则，不执行语句。**

- **形式2：**

```
if(表达式) 语句1
else 语句2
```

**执行过程: 当表达式值非0（即为真）时, 执行语句1；否则，执行语句2。**

- **形式3：**

```
if(表达式) 语句1
else if(表达式2) 语句2
else if(表达式3) 语句3
…
else if(表达式n) 语句n
else 语句n+1
```

- **执行过程:**

**（1）从上向下逐一对if后面的表达式进行检测；**

**（2）当某一个表达式的值为非0时，执行与此有关的子句中的语句，if语句的其余部分略过；**

**（3）如果所有表达式的值都是0，则执行最后的else子句；如果没有最后的else子句，则不进行任何操作。**

◆ if语句的注意问题：

- 表达式可以是逻辑表达式、关系表达式、算术表达式。
- 语句必须以分号结束。
- 若语句不止一条，则必须用{ }括起来，但{ }外不加分号。

## □ if语句的嵌套

- 嵌套的一般形式

```
if (条件1)
      if (条件2)  语句1      } 内嵌if
      else        语句2
else
      if(条件3)   语句3      } 内嵌if
      else        语句4
```

```
if (条件1)
{
      if (条件2)  语句1      } 内嵌if
      else        语句2
}
else
{
      if(条件3)   语句3      } 内嵌if
      else        语句4
}
```

- **if ~ else 配对原则：**

**缺省{ }时，else总是和它上面离它最近的未配对的if 配对。如果要改变这种配对规则，则需要在合适的位置加上{ }。**

```
if (条件1)
   if (条件2) 语句1
   else       语句2
else
   语句3
```

```
if (条件1)
   if (条件2)  语句1
else
   if (条件3)  语句2
   else        语句3
```

```
if (条件1)
{
   if (条件2)  语句1
}
else
   if (条件3)  语句2
   else        语句3
```

### 3.3.2 switch语句

**口 一般形式**

```
switch(表达式)
{
    case C1: 语句1 [break;]
    case C2: 语句2 [break;]
    …
    case Cn: 语句n [break;]
    [default: 语句n+1]
}
```

**执行过程：**

当表达式的值与某一个 case 后面的常量表达式的值相等时，就执行此 case 后面的语句；若所有 case 后的常量表达式的值都没有与表达式的值匹配，则执行 default 后面的语句。

◆ **说明：**

- **C1，C2，…，Cn是常量表达式，且值必须互不相同；**
- **常量表达式起语句标号作用，必须用break跳出；**
- **case后可包含多个可执行语句，且不必加{ }；**
- **switch可嵌套；**
- **多个case可共用一组执行语句。**

# 例3.3 根据成绩范围，打印成绩等级

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

int main()
{
    int score, num;
    char grade;

    cout << "Enter your score: " << endl;
    cin >> score;

    num = score / 10;
    switch(num) {
        case 10:
        case 9: grade = 'A'; break;
        case 8: grade = 'B'; break;
        case 7: grade = 'C'; break;
        case 6: grade = 'D'; break;
        default: grade = 'E';
    }

    cout << "score = " << score << " " << "grade = " << grade << endl;
    return 0;
}
```

```
Enter your score:
98
score = 98 grade = A
```

# 例3.4 switch语句的嵌套使用

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

int main()
{
    int x = 1, y = 0, a = 0, b = 0;

    switch(x) {
        case 1:
            switch(y) {
                case 0: a++;  break;
                case 1: b++;  break;
            }
        case 2:  a++; b++; break;
        case 3:  a++; b++;
    }

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;

    return 0;
}
```

```
a = 2
b = 1
```

# 3.4 循环结构

**实现循环结构的三种语句：**

- **while语句**
- **do-while语句**
- **for语句**

### 3.4.1 while语句

**口 一般形式：**

```
while(表达式)  语句
```

**口 功能：**

对表达式求值，若值为真则执行循环体。重复上述过程，直到表达式值为假时退出循环。

**口 说明：**

- 循环体有可能一次也不执行。
- 若循环体包含一条以上的语句，应以复合语句{ }形式出现。
- 无限循环: while(1) 循环体;
- 循环控制条件可多样，如while (i<=100)和while(getchar()!='\n') n++;。
- 循环体可为空，如：while((c=getchar( ))!='A') ;。

### 3.4.2 do-while语句

**口 一般形式：**

```
do
    语句
while(表达式);
```

**口 功能：**

**先执行循环体，然后对表达式求值。若值为真，则再次执行循环体，否则退出循环。**

# 例3.5 求 $\sum_{n=i}^{100} n$

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

int
main()
{
    int i,sum = 0;

    cin >> i;

    do {
        sum += i;
        i++;
    }
    while(i <= 100);

    cout << "sum = " << sum << endl;

    return 0;
}
```

```
1
sum = 5050
```

```
110
sum = 110
```

```cpp
#include <iostream>

int main()
{
    int i, sum = 0;

    cin >> i;

    while(i <= 100) {
        sum = sum + i;
        i++;
    }

    cout << "sum = " << sum << endl;

    return 0;
}
```

```
1
sum = 5050
```

```
110
sum = 0
```

### 3.4.3 for语句

**口 一般形式：**

```
for(表达式1; 表达式2; 表达式3)  语句
```

**口 执行过程：**

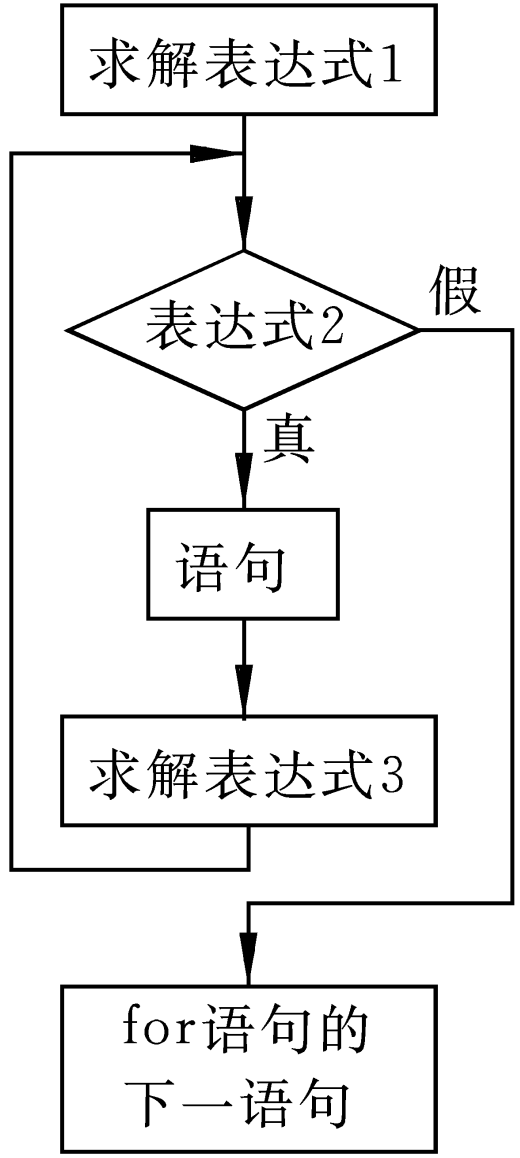
(1) 先求解表达式1。

(2) 求解表达式2，若其值为真(非0)，则执行for语句中的循环体语句，然后执行下面第(3)步；若其值为假(0)，则结束循环，转到第(5)步。

(3) 求解表达式3。

(4) 转回第(2)步继续执行。

(5) 循环结束，执行for语句后面的语句。



## 常用形式

```
for(循环变量赋初值;循环条件;循环变量增值)  语句
```

- 例如：for(i=1; i<=100; i++)  sum+=i;

## 说明

- for语句中表达式1、2、3类型任意，都可省略，但分号“;”不能省。

## □ 三种循环的比较

| 语句 | 适用范围 | 说明 |
| --- | --- | --- |
| while | 只知道结束条件而无法确定循环次数的情况下 | 1、循环变量初始化在循环语句之前完成；<br>2、循环体中应包含使循环结束的语句；<br>3、可用break和continue语句控制。 |
| do-while | 只知道结束条件而无法确定循环次数的情况下，但循环至少执行一次 | |
| for | 已知结束条件或循环次数的情况下 | 1、使用方式最为灵活；<br>2、可用break和continue语句控制。 |

### 3.4.4 循环的嵌套

## 口 定义

**一个循环体内又包含了另一个完整的循环结构，称为循环的嵌套。**

- **三种循环可以互相嵌套，层数不限。**

(1)
```
while( )
{   ……
    while()
    {   ……
    }
  ……
}
```

(2)
```
do
{   ……
    do
    {   ……
    }while( );
  ……
}while( );
```

(3)
```
for( )
{   ……
    for(; ;)
    {   ……
    }
……
}
```

(4)
```
while( )
{   ……
    do
    {   ……
    }while( );
……
}
```

(5)
```
for( ; ;)
{   ……
    while()
    {   ……
    }
  ……
}
```

(6)
```
do
{   ……
    for(; ;)
    {   ……
    }
  ……
}while( );
```

## 3.4.5 break语句、continue语句和goto语句

### 口 break语句

- **使用形式**

```
break;
```

- **功能**

  **用在循环语句和switch语句中，用于终止并跳出整个语句结构。**

- **说明：**

  - **break只能终止并跳出包含它的本层语句结构。**
  - **break只能用于循环语句和switch语句。**

## 例3.6 若输入英文字母，则原样输出；输入其他字符不理会，直到输入q键结束程序。

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

int main()
{
    char ch;
    while(1) {
        cout << "Input a letter ('Q' or 'q' to quit): " << endl;
        cin >> ch;

        if((ch == 'Q') || (ch == 'q')) {
            cout << "Quit!" << endl;
            break;
        } else if((ch>='A' && ch<='Z') || (ch>='a' && ch<='z')) {
                cout << ch << endl;
        }
    }

    return 0;
}
```

```
Input a letter (’Q’ or ’q’ to quit): g
g
Input a letter (’Q’ or ’q’ to quit): e
e
Input a letter (’Q’ or ’q’ to quit): b
b
Input a letter (’Q’ or ’q’ to quit): k
k
Input a letter (’Q’ or ’q’ to quit): q
Quit!
```

## □ continue语句

- **使用形式**

```
continue;
```

- **功能**

  **在循环语句中用于结束本次循环，即跳过循环体中尚未执行的语句，下一次循环条件的判断。**

# 例3.7 把100~200之间不能被3整除的数输出

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

int main()
{
    int i, cnt = 0;
    for(i=100; i<=200; i++) {
        if(i%3 == 0)
            continue;

        cout << setw(5) << setiosflags(ios::left) << i;
        if(++cnt == 10) {
            cout << endl;
            cnt = 0;
        }
    }

    return 0;
}
```

```
100  101  103  104  106  107  109  110  112  113
115  116  118  119  121  122  124  125  127  128
130  131  133  134  136  137  139  140  142  143
145  146  148  149  151  152  154  155  157  158
160  161  163  164  166  167  169  170  172  173
175  176  178  179  181  182  184  185  187  188
190  191  193  194  196  197  199  200
```

## □ goto语句

- **使用形式**

```
        goto    语句标号;

        …

标号: 语句;
```

**其中，语句标号：1）用于定义程序中的某个位置；2）用标识符表示，开头不能是数字；3）只能加在可执行语句前面。**

- **功能：无条件地跳转到语句标号指向的语句去执行。**
- **说明：（1）从多层嵌套中跳出到最外层；（2）可用于各种结构；（3）只能在函数体中跳转，即不能从函数体中跳出。**

## 例3.8 判断素数

- 什么是素数----只能被自身和1整除的自然数。
- 判断方法----让m依次被2, 3, 4, ..., $\sqrt{m}$除，如果m能被其中的任何一个整数整除，则不是素数。

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

int main()
{
    int m, i, k;

    cin >> m;
    k = sqrt(m);

    for(i=2; i<=k; i++)
        if(m%i == 0)  break;

    if(i > k)  cout << m << " is a prime number" << endl;
    else   cout << m << " is not a prime number" << endl;

    return 0;
}
```

```
19
19 is a prime number
```