



西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

程序设计基础 Programming in C++

U10G13027/U10G13015

主讲：魏英，计算机学院

- ▶ 3.1 语句
- ▶ 3.2 输入与输出
- ▶ 3.3 程序顺序结构
- ▶ 3.4 程序选择结构
- ▶ 3.5 程序循环结构

3.1 语句

- ▶ 语句（statement）是C++程序的最小单位
- ▶ 程序由一条一条语句组成
- ▶ C++语言语句分为简单语句、复合语句和控制语句

► 1. 表达式语句

表达式; //用分号结束

x=a+b; //赋值语句

t=a, a=b, b=t; //a和b交换

a+b+c; //运算但无实际意义

▶ 2. 函数调用语句

函数调用(实参); //用分号结束

max(a, b); //输出流函数调用语句

▶ 3. 空语句

```
;
```

3.1.2 复合语句

- ▶ 复合语句（compound statement），又称语句块，简称块（block）

```
{ //复合语句
    double s, a=5, b=10, h=8; //局部声明
    s=(a+b)*h/2.0;
    cout<< "area=" <<s<<endl;
} //复合语句不需要分号结尾
```

- ▶ 可以在程序中编写注释（comments），有两种形式：
- ▶ ① /*.....*/块注释语法形式：

```
/*  
..... 注释内容  
*/
```

- ▶ ② //行注释语法形式：

```
//..... 注释内容
```


- ▶ （1）多数情况下，在一个程序行里只写一个语句，这样的程序写法清晰，便于阅读、理解和调试。
- ▶ （2）注意使用空格或TAB来作合理的间隔、缩进、对齐，使程序形成逻辑相关的块状结构，养成优美的程序编写风格。
- ▶ （3）C++语言允许在一行里写多个语句 。

3.2 输入与输出

- 所谓**输入**是指从外部输入设备（如键盘、鼠标等）向计算机输入数据，**输出**是指从计算机向外部输出设备（如显示器、打印机等）输出数据。



3.2 输入与输出

- ▶ C++语言输入输出操作是用流对象（stream）实现的。
- ▶ 若在程序中使用流对象cin和cout，应该用文件包含预处理命令将标准输入输出流库的头文件<iostream>包含到源文件中。

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

3.2.1 输入流与输出流

► cout和cin对象的使用

```
cout<<表达式1<<表达式2<<……;
```

```
cin>>变量1>>变量2>>……;
```

```
int x, y;  
cin>>x>>y;    //键盘输入  
cout<< “x=” <<x<< “, y=” <<y<<endl; //输出到显示器上
```

3.2.1 输入流与输出流

- ▶ cin输入时，为了分隔多项数据，默认要求在键盘输入数据之间使用空格、Tab键、回车作为分隔符。

```
#include <iostream>
using namespace std;
int main()
{
    char c1, c2, c3;
    cin>>c1>>c2>>c3;
    cout<<"c1="<<c1<<", c2="<<c2<<", c3="<<c3<<endl;
    return 0;
}
```

123 ↵

c1=1,c2=2,c3=3

► 2. 格式控制

有时希望数据按指定的格式输入输出，如要求以十六进制或八进制形式输入一个整数，或希望输出的实型数只保留两位小数等。有两种方法可以达到这样的要求：

- (1) 在输入输出流中使用控制符进行格式控制。使用这种方法，要在程序中加入<iomanip>头文件。

具体格式控制符参照教材P75表3-3

3.2.1 输入流与输出流

例3.1 使用cin和cout输入输出数据

```
1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4  int main()
5  {
6      bool v; int a,m,n;
7      double x,y,z,p,f; float f1;
8      cin>>boolalpha>>v;    //输入: true
9      cin>>oct>>a>>hex>>m>>dec>>n; //输入: 144 46 -77
10     cin>>p>>f>>f1>>x>>y>>z;
11     cout<<v<< '\t' <<boolalpha<<v<< '\t' <<noboolalpha<<v<<endl;
12     cout<<a<< '\t' <<p<<endl<<a*p<<endl;
13     cout<<hex<<m<< '\t' <<oct<<m<< '\t' <<dec<<m<<endl;
14     cout<<showbase<<hex<<m<< '\t' <<oct<<m<< '\t' <<dec<<m<<endl;
15     cout<<precision(5); cout<<x<< '\t' <<y<< '\t' <<z<<endl;
16     cout<<fixed<<x<< '\t' <<y<< '\t' <<z<<endl;
```

3.2.1 输入流与输出流

```
17  cout<<scientific<<x<< '\t' <<y<< '\t' <<z<<endl;
18  cout<<left<<setw(6)<<n<< '\t' <<internal<<setw(6)<<n<< '\t' ;
19  cout.width(6);  cout<<right<<n<<setfill( '0' )<<endl;
20  cout<<setw(10)<<77<< '\t' <<setfill( 'x' )<<setw(10)<<77<<endl;
21  cout<<setprecision(5)<<f1<< '\t' <<setprecision(9)<<f1<<endl;
22  cout<<fixed<<setprecision(5)<<f<< '\t' <<setprecision(9)<<f<<endl;
23  cout<<scientific<<setprecision(5)<<f<< '\t' <<setprecision(9)<<f<<endl;
24  cout<<setiosflags( ios_base::floatfield|ios_base::showpoint );
25  cout<<setprecision(0)<<f<< '\t' <<setprecision(9)<<f<<endl;
26  cout<<showpos<<1<< '\t' <<0<< '\t' <<-1<<endl;
27  cout<<noshowpos<< 1<< '\t' <<0<< '\t' <<-1<<endl;
28  return 0;
29 }
```


- ▶ 1. 字符输出putchar函数
- ▶ putchar函数的作用是向显示终端输出一个字符，一般形式为：

```
putchar(c);
```

- ▶ 如：char c='a';
putchar(c); 等价于 cout<<c;
等价于 putchar(97);

- ▶ 2. 字符输入getchar函数
- ▶ getchar函数的作用是从键盘终端输入一个字符，一般形式为：

```
getchar()
```

3.2.2 字符输入与输出

例:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {   char c1, c2, c3;
5      c1=getchar();
6      c2=getchar();
7      c3=getchar();
8      putchar(c1);putchar(c2);putchar(c3);
9      return 0;
10 }
```

a ↙
bc ↙
a
b

3.2.3 格式化输出

- ▶ 使用printf函数和scanf函数进行输入输出，需要在程序中包含<stdio.h>头文件。
- ▶ 1. 用printf函数来进行输出

```
printf(格式控制, 输出项列表.....);
```

```
#include <stdio.h>
int main()
{   int i=10;   double f=10.0;   char c='a';
    printf("i=%d, f=%lf, c=%c\n", i, f, c);
    return 0;
}
```

► 2. 用scanf函数来进行输入

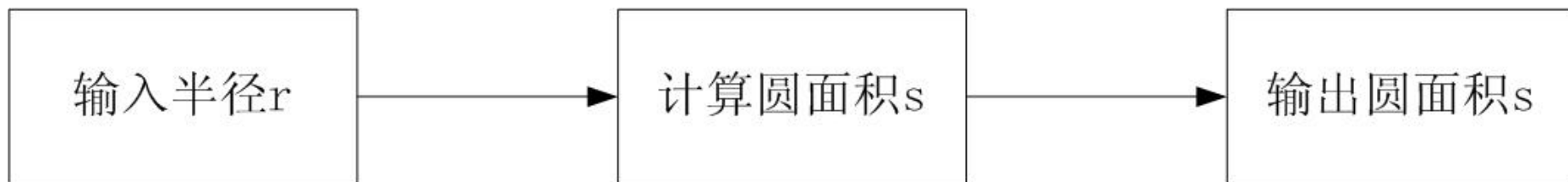
scanf(格式控制, 输入项列表.....);

```
#include <stdio.h>
int main()
{   int i;   double f;   char c;
    scanf( "%d%lf%c" , &i, &f, &c)
    printf("i=%d, f=%f, c=%c\n", i, f, c);
    return 0;
}
```

3.3 程序顺序结构

- ▶ 通常情况下，语句以其出现的顺序执行
- ▶ 一个语句执行完会自动转到下一个语句开始执行，这样的执行称为顺序执行。
- ▶ 顺序执行的次序是很重要的

图3.2 求圆面积的执行次序



- ▶ 从问题求解的一般过程来看，还需要跳转执行。
- ▶ ①选择语句：if语句、switch语句；
- ▶ ②循环语句：while语句、do语句、for语句；
- ▶ ③跳转语句：goto语句（不建议使用）、break语句、continue语句、return语句。

- ▶ 1. if语句
- ▶ 2. switch语句

3.4.1 if语句

- ▶ if语句的作用是计算给定的表达式，根据结果选择执行相应的语句。语句形式有两种：
- ▶ ①if形式：

```
if (表达式) 语句1;
```

```
如： if ( a>b ) t=a, a=b, b=t;
```

- ▶ ②if-else形式：

```
if (表达式) 语句1; else 语句2;
```

```
如： if ( x>=y ) cout<<x<<endl;  
      else  cout<<y<<endl;
```

3.4.1 if语句

- ▶ (1) if语句中的子语句既可以是简单语句，又可以是复合语句或控制语句，但必须是“一个语句”的语法形式

```
1  if (a>b)          //此段程序代码编译有错误
2      x=a+b; y=a-b;
3  else
4      x=a-b; y=a+b;
```

```
1  if (a>b) {        //此段程序代码正确
2      x=a+b; y=a-b;
3  }
4  else {
5      x=a-b; y=a+b;
6  }
```

3.4.1 if语句



► 总是交换的代码（不好的code风格）：

```
1  if ( a>b )      ;
2  {
3      t=a;
4      a=b;
5      b=t;
6  }
```

```
1  if ( a>b ) {      ;
2      t=a;
3      a=b;
4      b=t;
5  }
```

3.4.1 if语句

- ▶ (2) if语句的表达式一般为关系、逻辑运算表达式，但也可以为其他表达式，但按逻辑值来理解：

```
a=5, b=2;  
if ( a ) x=a*10;    //等价于a!=0
```

3.4.1 if语句

例3.2 计算三角形的面积

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main()
5  {
6      double a, b, c;
7      cin>>a>>b>>c;
8      //判断三边长是否构成三角形
9      if (a+b>c && a+c>b && b+c>a) {
10         double s, t;
11         t=(a+b+c)/2.0;
12         s=sqrt(t*(t-a)*(t-b)*(t-c));
13         cout<< "area=" <<s<<endl;
14     }
15     else cout<< "error" <<endl;
16     return 0;
17 }
```

3 4 5
area=6

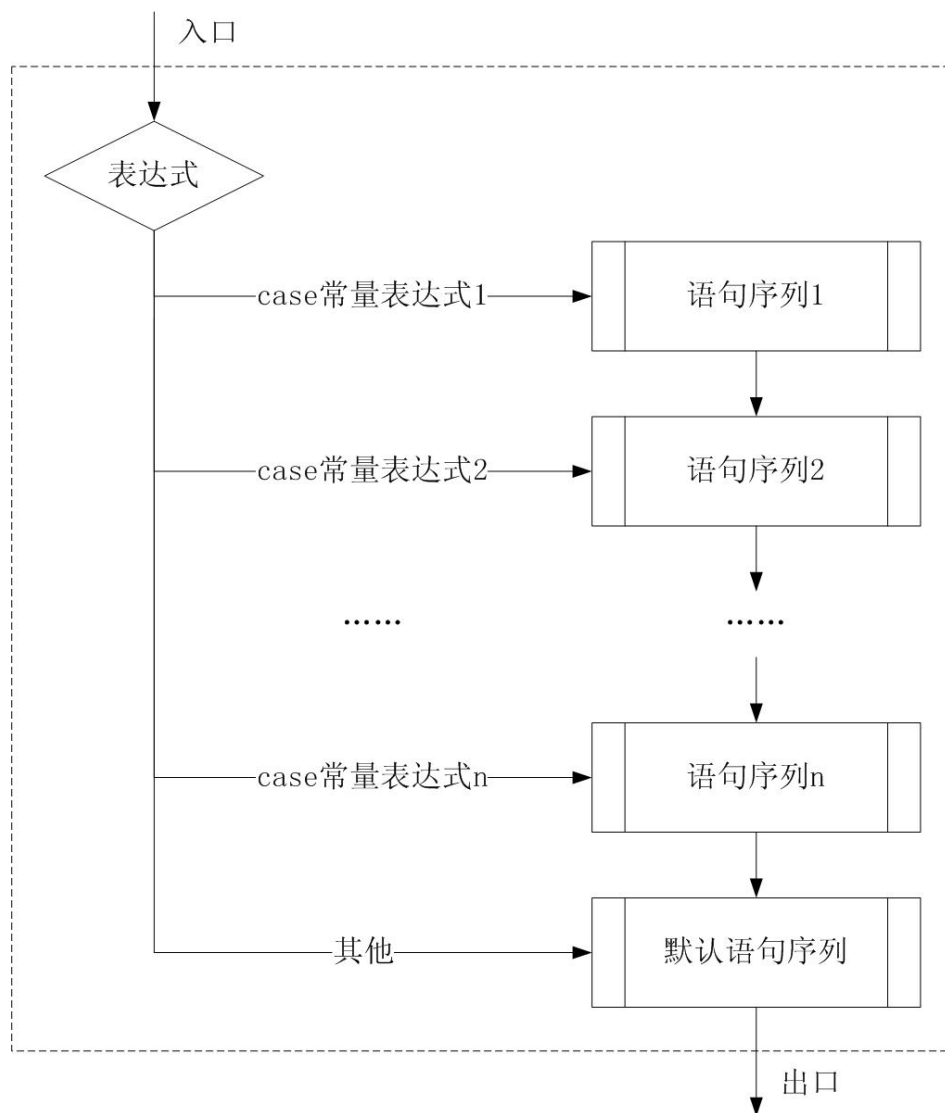
- ▶ switch语句
- ▶ switch语句的作用是计算给定的表达式，根据结果选择从哪个分支入口执行，语句形式为：

```
switch ( 表达式 ) {  
    case 常量表达式1 :… 语句序列1  
    case 常量表达式2 :… 语句序列2  
        ⋮  
    case 常量表达式n :… 语句序列n  
    default           : …默认语句序列  
}
```

3.4.2 switch语句

图3.4 switch语句执行流程

```
switch ( 表达式 ) {  
    case 常量表达式1 : 语句序列1  
    case 常量表达式2 : 语句序列2  
        ⋮  
    case 常量表达式n : 语句序列n  
    default           : 默认语句序列  
}
```



- ▶ switch语句的使用说明：
 - ▶ （1）switch语句中case分支的语句序列可以是一个语句，也可以是任意多的语句序列，也可以没有语句；
 - ▶ （2）如果case后没有语句，则一旦执行到这个case分支，什么也不做，继续往下执行。

3.4.2 switch语句

```
1  switch (n) {  
2      case 7 : cout<< “step5” <<endl;  
3      case 6 :  
4      case 5 : cout<< “step4” <<endl;  
5      case 4 :  
6          {  
7              cout<< “step3” << “step2” <<endl;  
8          }  
9      case 2 : cout<< “step1” <<endl;  
10     default: cout<< “step0” <<endl;  
11 }
```

- ▶ (3) switch语法中各个case分支和default分支的出现次序在语法上没有规定，但次序的不同安排会影响执行结果。

```
1 //①程序A
2 switch (n) {
3     case 1 : cout<<"1";
4     case 2 : cout<<"2";
5     default: cout<<"0";
6 }
```

```
1 //②程序B
2 switch (n) {
3     default: cout<<"0";
4     case 1 : cout<<"1";
5     case 2 : cout<<"2";
6 }
```

- ▶ （4）switch语法中default分支是可选的，若没有default分支且没有任何case标号的值相等时，switch语句将什么也不做，直接执行后续语句。
- ▶ （5）switch语句的分支表达式可以是C++语言的任意表达式，但其值必须是整数（含字符类型）、枚举类型。

- ▶ （6）switch语法中的case后的表达式必须是常量表达式且互不相同，即为整型、字符型、枚举类型的常量值，但不能包含变量。
- ▶ 例如：若c是变量，如：“case c>='a' && c<='z': ”的写法是错的。
- ▶ （7）case分支后面的冒号是必须的，即使没有后面的语句序列。

- ▶ （8）在switch语句中任意位置上，只要执行到break语句，就结束switch语句的执行，转到后续语句。

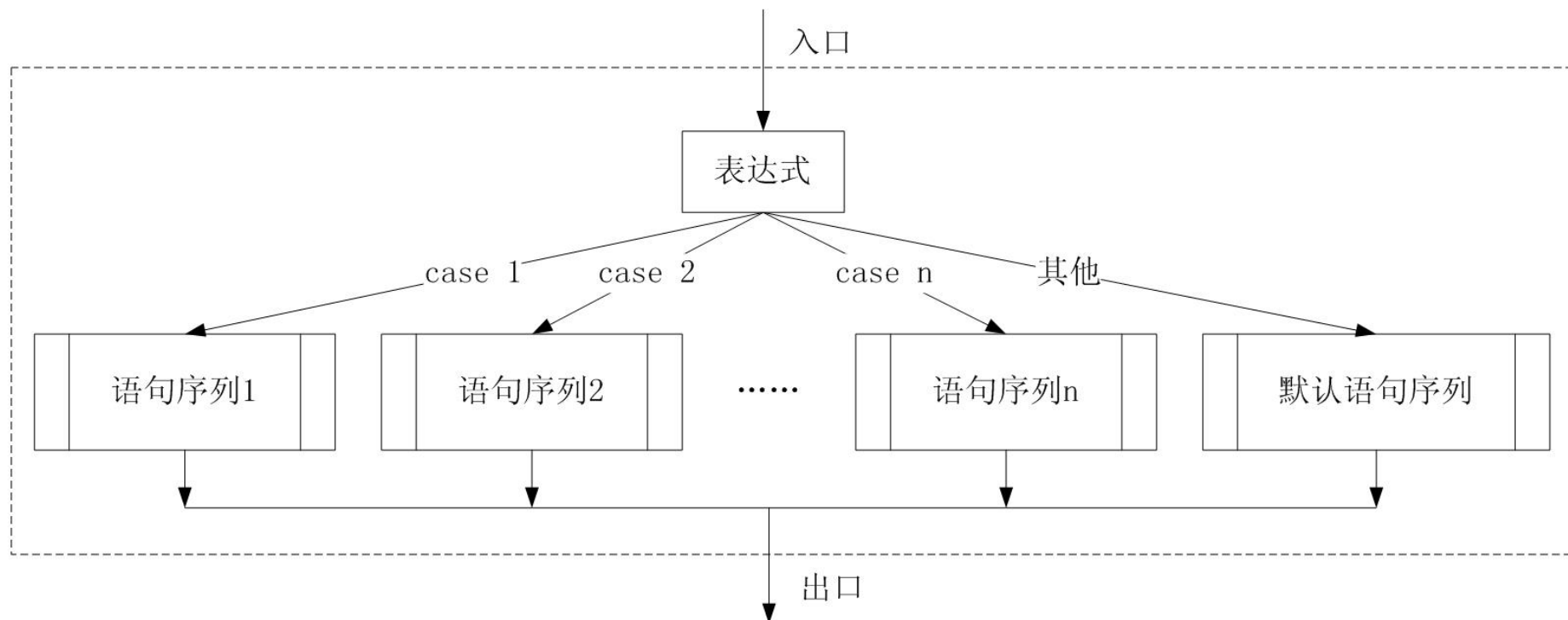
```
break;
```

- ▶ 更常见的switch结构应该如下，它提供了程序多分支选择执行流程。

```
switch ( 表达式 ) {  
    case 常量表达式1:…语句序列1;break;  
    case 常量表达式2:…语句序列2;break;  
    ⋮  
    case 常量表达式n:…语句序列n;break;  
    default           :…默认语句序列  
}
```

3.4.2 switch语句

图3.5 结构化的switch流程



例：按照考试成绩的等级输出百分制分数段

```
int main( )
{   char g;
    cin>>g;
    switch( g )
    {   case 'A':cout<<"85~100\n"; break;
        case 'B':cout<<"70~84\n" ; break;
        case 'C':cout<<"60~69\n" ; break;
        case 'D':cout<<"<60\n" ; break;
        default: cout<<"error\n" ;
    }
    return 0;
}
```

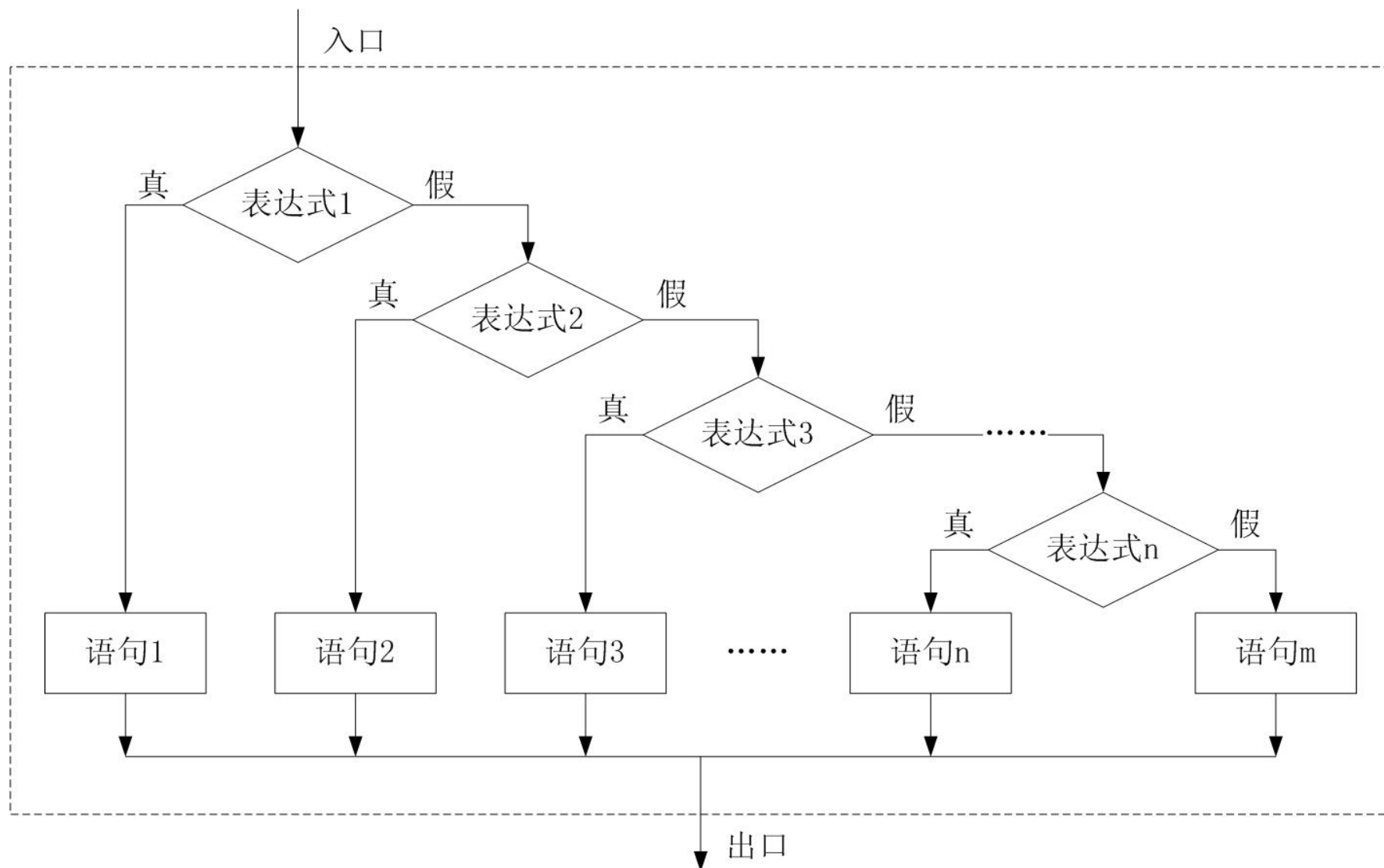

3.4.3 选择结构的嵌套

- ▶ 1. if语句的嵌套
- ▶ (1) 第一种形式，在else分支上嵌套if语句，语法形式为：

```
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 if ( 表达式n ) 语句n  
else 语句m
```

3.4.3 选择结构的嵌套

图3.6 嵌套if语句第一种形式的执行流程



3.4.3 选择结构的嵌套

例3.3 编程输出成绩分类

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int score;
6      cin>>score;
7      if (score >= 90 ) cout<< "A" <<endl;
8      else if (score >= 80 ) cout<< "B" <<endl;
9      else if (score >= 70 ) cout<< "C" <<endl;
10     else if (score >= 60 ) cout<< "D" <<endl;
11     else cout<< "E" <<endl;
12     return 0;
13 }
```

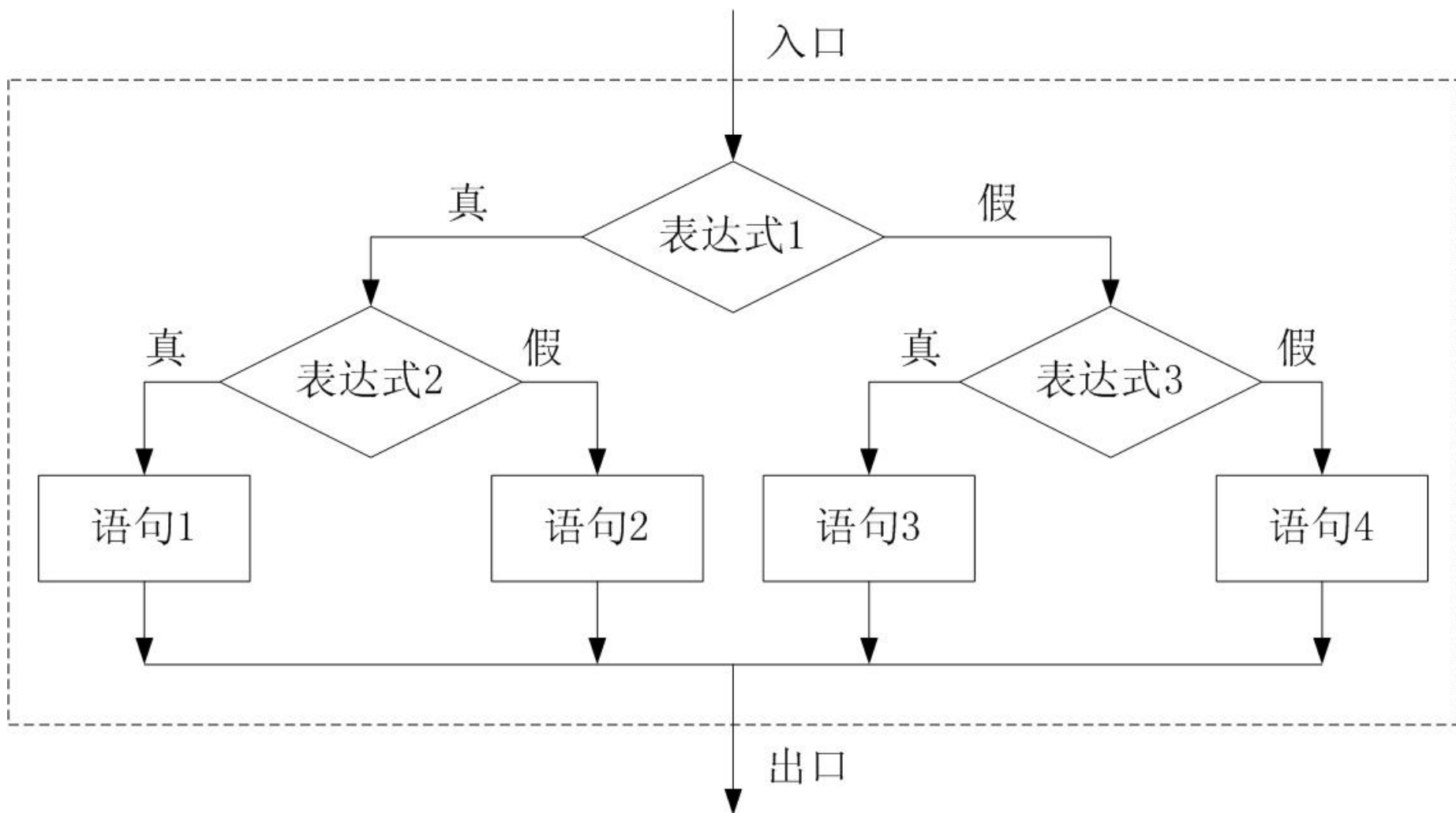
3.4.3 选择结构的嵌套

- ▶ (2) 第二种形式，在if和else分支上嵌套if语句，语法形式为：

```
if ( 表达式1 )  
    if ( 表达式2 ) 语句1  
    else 语句2  
else  
    if ( 表达式3 ) 语句3  
    else 语句4
```

3.4.3 选择结构的嵌套

图3.7 嵌套if语句第二种形式的执行流程



3.4.3 选择结构的嵌套

- ▶ 注意：嵌套的if语句可以实现多路分支。在结构上else语句总和距离自己最近的if相匹配。我们可以通过加“{ }”来改变else的层位，从而改变程序的执行流程。

```
int x=10,y=0,
```

比较：

```
if (x)
if (y) cout<<"1";
else cout<<"2";
```



输出：2

与

```
if (x)
{ if (y) cout<<"1"; }
else cout<<"2";
```



没有输出

3.4.3 选择结构的嵌套

例： 计算函数

$$y = \begin{cases} x-1 & (x < -3) \\ \sqrt{9-x^2} & (-3 \leq x \leq 3) \\ x+1 & (x > 3) \end{cases}$$

```
#include <cmath>
int main( )
{
    double x, y;
    cin>>x;
    if(x<-3.0)  y=x-1.0;
    else
        if(x>= -3.0 && x<=3.0)
            y=sqrt(9.0-x*x);
        else
            y=x+1;
    cout<< "x=" <<x<< ", y=" <<y<<endl;
    return 0;
}
```

- ▶ 2. switch语句的嵌套
- ▶ switch语句是可以嵌套的：

```
1  int a=15, b=21, m=0;
2  switch(a%3) {
3      case 0: m++;
4          switch(b%2) {
5              default: m++;
6              case 0 : m++; break;
7          }
8      case 1: m++;
9  }
```


3.4.4 选择结构程序举例

例3.5 输入某天的日期，输出第二天的日期

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int y, m, d, Days;
6      cin>>y>>m>>d;  //输入日期
7      switch(m) { //计算每月的天数
8          case 2 :
9              Days=28;
10             if((y%4==0&& y%100!=0) || (y%400==0)) Days++;
11             break;
12             case 4 : case 6 :
13             case 9 : case 11 : Days=30; break;
```

3.4.4 选择结构程序举例

例3.5

```
14         default: Days=31; //其余月份为31天
15     }
16     d++;
17     if (d>Days) d=1, m++; //判断月末
18     if (m>12) m=1, y++; //判断年末
19     cout<<y<< “-” <<m<< “-” <<d<<endl;
20     return 0;
21 }
```

► 比较一下三种多路分支结构：

```
if ( 表达式1 )  
    if ( 表达式2 ) 语句1  
    else 语句2  
else  
    if ( 表达式3 ) 语句3  
    else 语句4
```

```
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 if ( 表达式n ) 语句n  
else 语句m
```

```
switch ( 表达式 ) {  
    case 常量表达式1:…语句序列1;break;  
    case 常量表达式2:…语句序列2;break;  
    :  
    case 常量表达式n:…语句序列n;break;  
    default          :…默认语句序列  
}
```

- ▶ **循环**：就是在满足一定条件时，重复执行一段程序。
- ▶ 1. while语句
- ▶ 2. do while语句
- ▶ 3. for语句

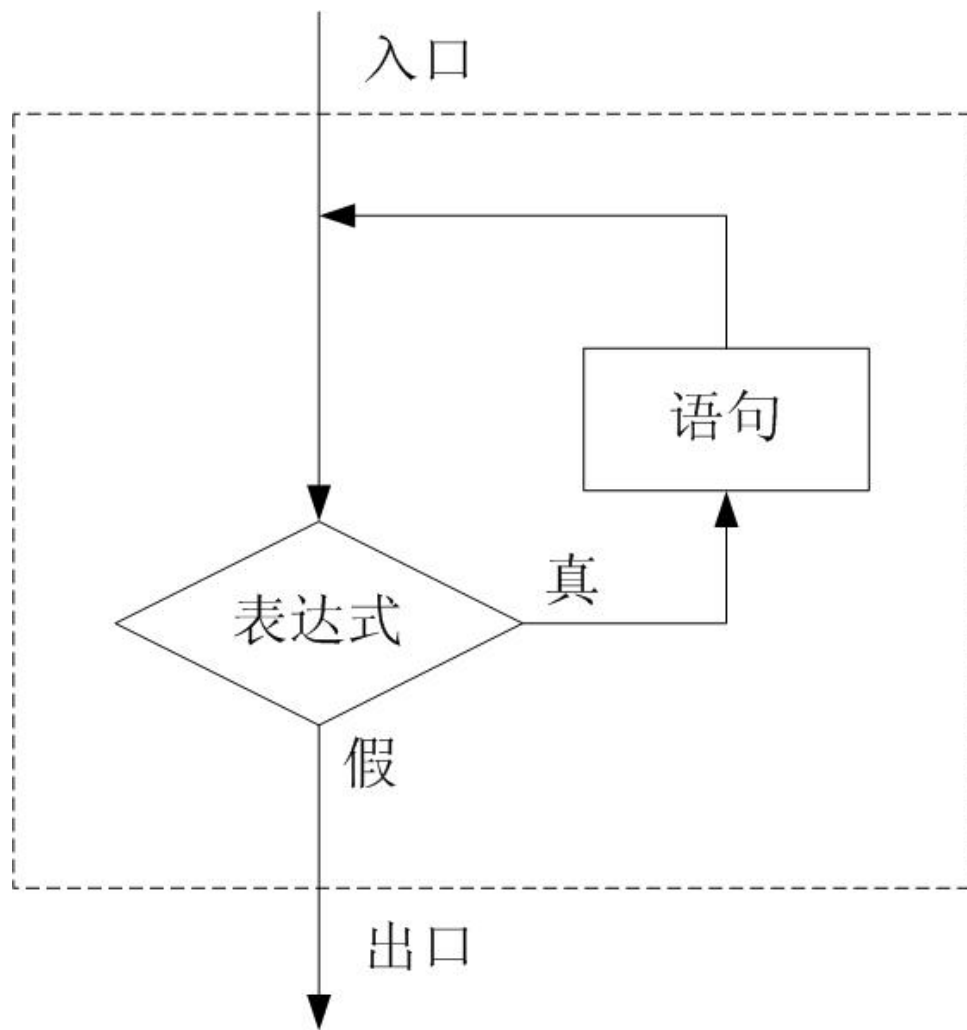
▶ while语句

`while` (表达式) 语句;

- ▶ 其中的语句称为子语句，又称循环体，圆括号内的表达式称为循环条件。

3.5.1 while语句

图3.9 while语句执行流程



(a)

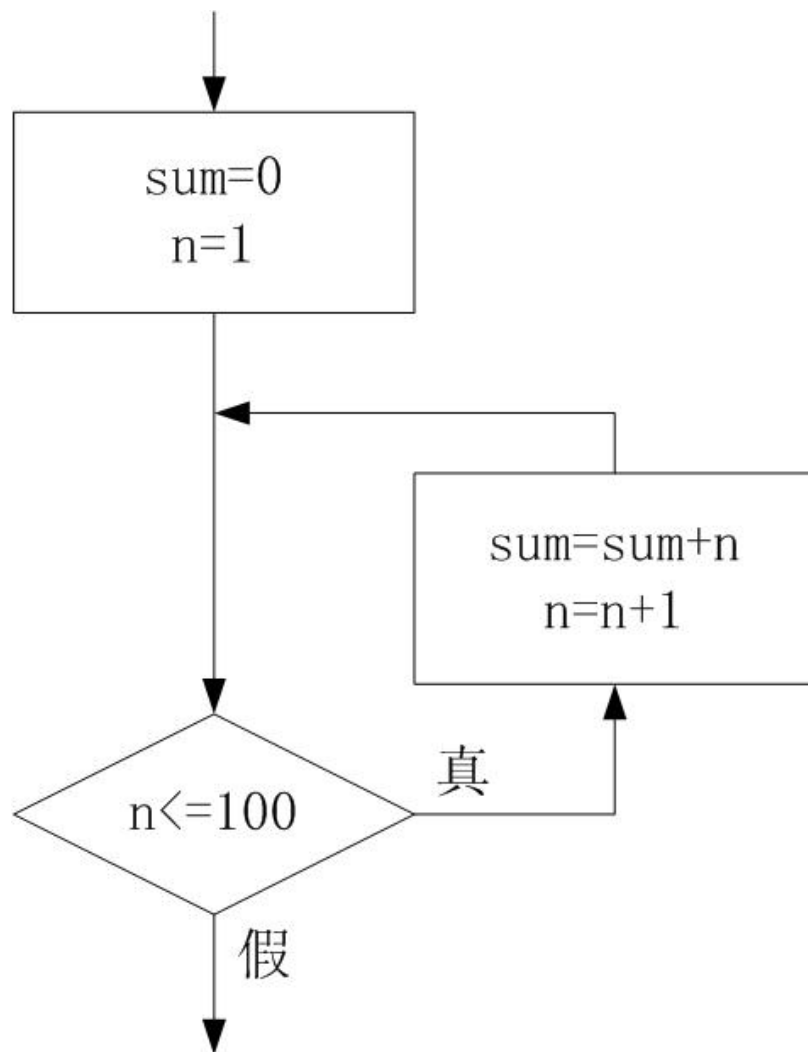
3.5.1 while语句

图3.9 while语句执行流程

▶ 【例3.7】

▶ 求 $s = \sum_{n=1}^{100} n$

▶ 即 $s = 1 + 2 + 3 + \dots + 100$

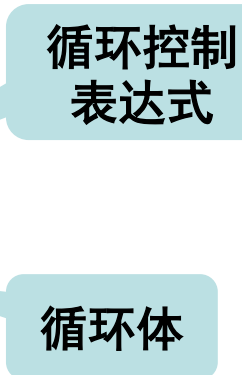


(b)

3.5.1 while语句

例3.7

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int n=1, sum=0;
6      while (n<=100) {
7          sum=sum+n;
8          n=n+1;
9      }
10     cout<< "sum=" <<sum<<endl;
11     return 0;
12 }
```



The diagram consists of two light blue callout boxes. The first box, labeled '循环控制表达式' (Loop Control Expression), has a pointer to the condition 'n<=100' in the while statement on line 6. The second box, labeled '循环体' (Loop Body), has a pointer to the block of code between lines 7 and 8, which is enclosed in curly braces.

注意:

①若循环体包含一条以上的语句，应以复合语句形式出现。

②循环前，必须给循环控制变量赋初值。

③在循环体里面，必须有改变循环控制变量值的语句。

- ▶ while语句的说明。
- ▶ (1)由于while语句先计算表达式的值，再判断是否循环，所以如果表达式的值一开始就为假，则循环一次也不执行，失去了循环的意义。
- ▶ (2)while语句循环条件可以是C++语言的任意表达式。通常情况下，循环条件是关系表达式或逻辑表达式，应该谨慎出现别的表达式。

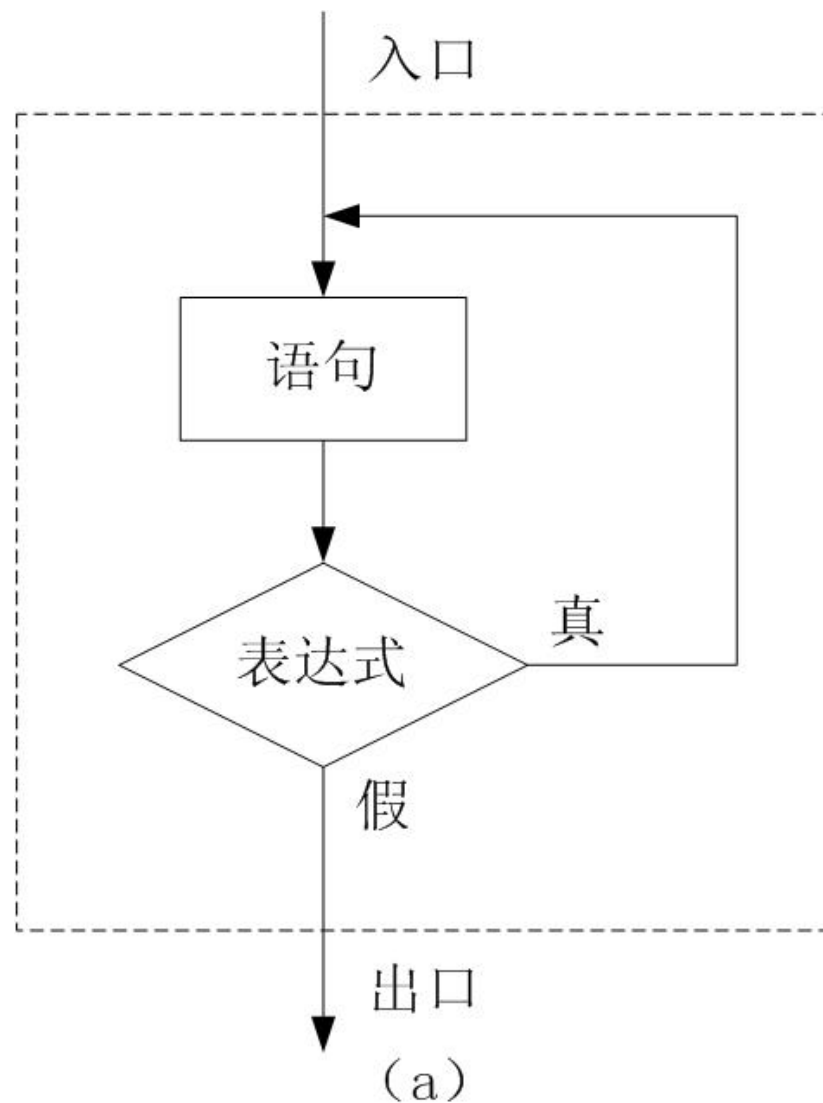
► do while语句

```
do 语句 while ( 表达式 );
```

- 其中的语句称为子语句，又称循环体，圆括号内的表达式称为循环条件。

3.5.2 do语句

图3.10 do-while语句执行流程



- ▶ do语句的说明。
- ▶ （1）do语句的最后必须用分号（；）作为语句结束，循环体的复合语句形式为：

```
do {  
    ... //复合语句  
} while ( 表达式 );
```

- ▶ （2）do语句先执行后判定，while语句则是先判定后执行；do语句至少要执行循环体一次，而while语句可能一次也不执行。
- ▶ （3）do语句结构和while语句结构是可以相互替换的。

3.5.2 do语句

例3.8 连续输入多个数据，计算它们的乘积，当输入0时结束

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int n=1, k=1;
6      do {
7          k=k*n;
8          cin>>n;
9      } while (n!=0); //输入0时结束循环
10     cout<<k<<endl;
11     return 0;
12 }
```

- ▶ for语句

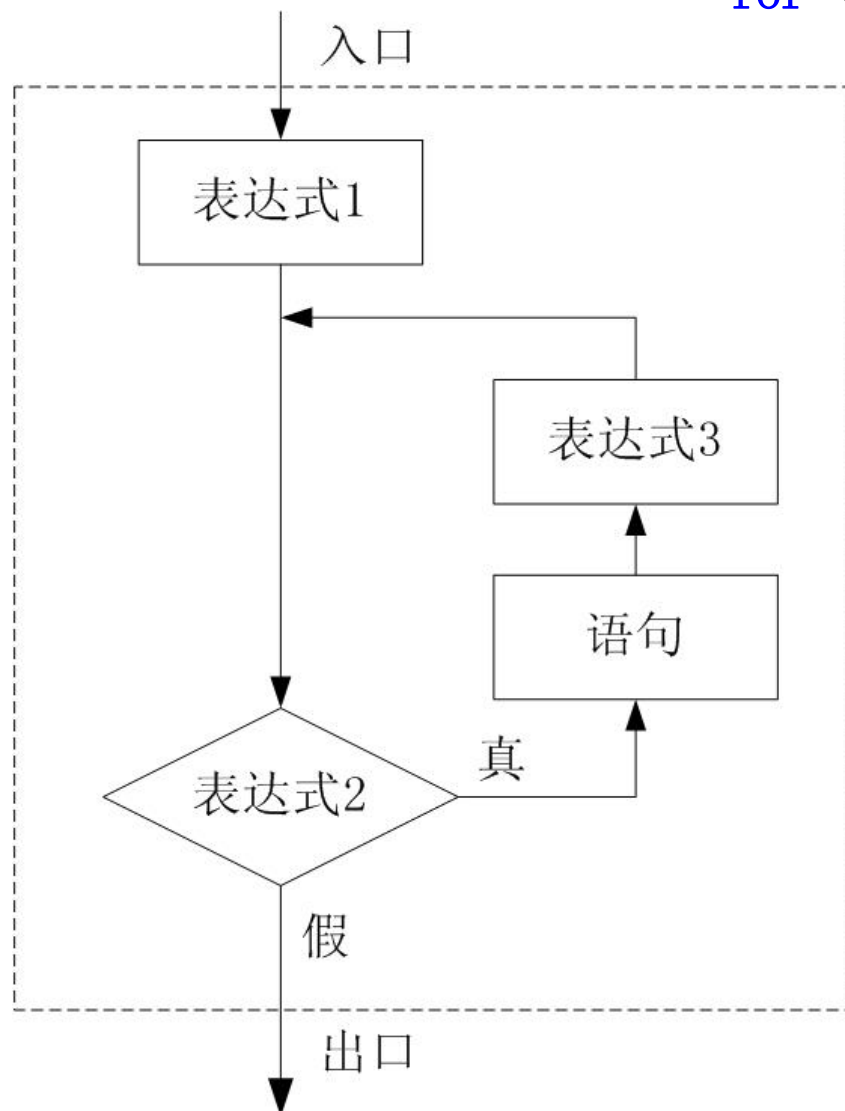
- ▶ for语句有循环初始和循环控制功能，语句形式为：

`for (表达式1; 表达式2; 表达式3) 语句;`

3.5.3 for语句

图3.11 for语句执行流程

for (表达式1; 表达式2; 表达式3) 语句;



整个循环过程中**表达式1**只求解一次；作用是给循环控制变量赋初值。

表达式2相当于是for的循环条件。

表达式3是重复执行的内容；通常是改变循环控制变量值的语句。

► for语句的应用格式

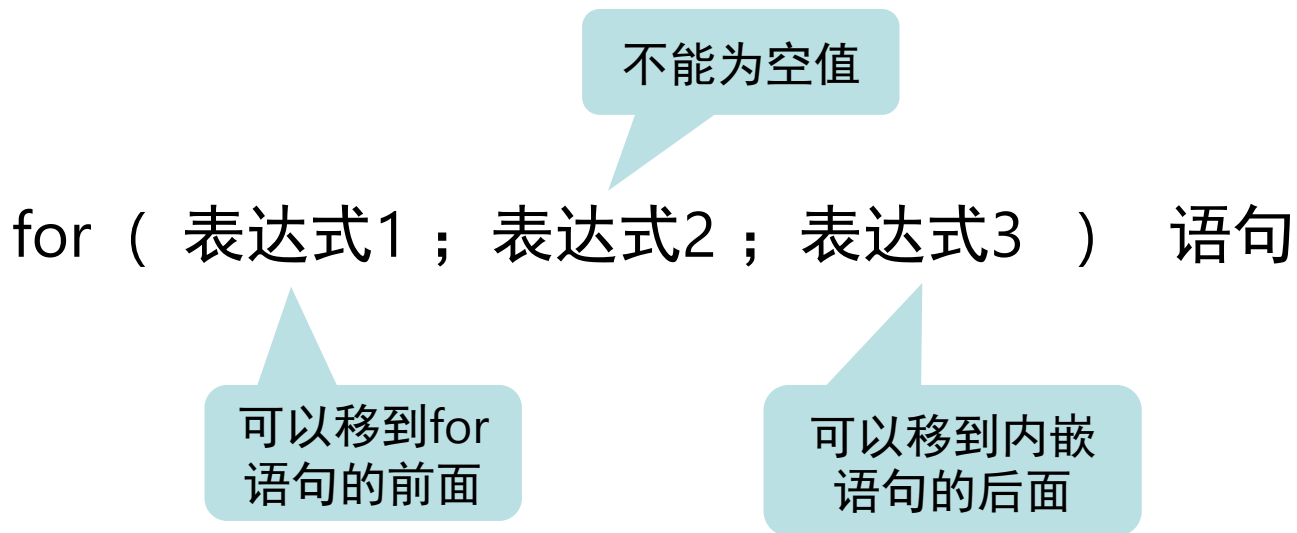
```
for(循环初始; 循环条件; 循环控制) 循环体;
```

► 如：求 $1+2+3+\dots+100$

```
for (n=1, sum=0 ; n<=100 ; n++) sum=sum+n;
```

```
int n=1, sum=0;  
while(n<=100) {  
    sum=sum+n;  
    n=n+1;  
}
```

for循环的其他形式:



```
表达式1;  
for ( ; 表达式2 ; 表达式3 ) 语句
```

```
for ( 表达式1; 表达式2 ; )  
{语句  
  表达式3;  
}
```

▶ 省略表达式1。

```
n=1, sum=0;  
for ( ; n<=100 ; n++) sum=sum+n;
```

▶ 省略表达式3。

```
for (n=1, sum=0; n<=100;) sum=sum+n, n++;
```

▶ 表达式1和表达式3都省略

```
n=1, sum=0  
for (; n<=100 ; ) sum=sum+n, n++;
```

- ▶ break语句
- ▶ break语句的作用是结束switch语句和循环语句的运行，转到后续语句，语法形式为：

```
break;
```

3.5.4 break语句

- ▶ break语句只能用在switch语句和循环语句（while、do、for）中，不得单独使用。
- ▶ 显然，在循环结构中使用break语句的目的就是提前结束循环。

3.5.4 break语句

例3.9 判断一个数m是否是素数

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i, m;
6      cin>>m;
7      //从2到m-1之间逐一检查是否被m整除
8      for (i=2 ; i<=m-1 ; i++)
9          if (m%i==0) break;
10     if(i==m) cout<< "Yes" <<endl;
11     else cout<< "No" <<endl;
12     return 0;
13 }
```

- ▶ 如此，循环语句的结束就有两个手段了：
- ▶ 一是循环条件
- ▶ 二是应用break语句。

3.5.5 continue语句

- ▶ continue语句
- ▶ continue语句的作用是在循环体中结束本次循环，直接进入下一次循环，语句形式为：

```
continue;
```


- ▶ continue语句只能用在循环语句（while、do、for）中，不能单独使用。
- ▶ 在while语句和do语句循环体中执行continue语句，程序会转到“表达式”继续运行，在for语句循环体中执行continue语句，程序会转到“表达式3”继续运行，循环体中余下的语句被跳过了。

► 比较下面两段程序：

```
1  for (n=1, sum=0 ; n<=100 ; n++) {  
2      if (n%2==0) break;  
3      sum=sum+n;  
4  }
```

```
1  for (n=1, sum=0 ; n<=100 ; n++) {  
2      if (n%2==0) continue;  
3      sum=sum+n;  
4  }
```

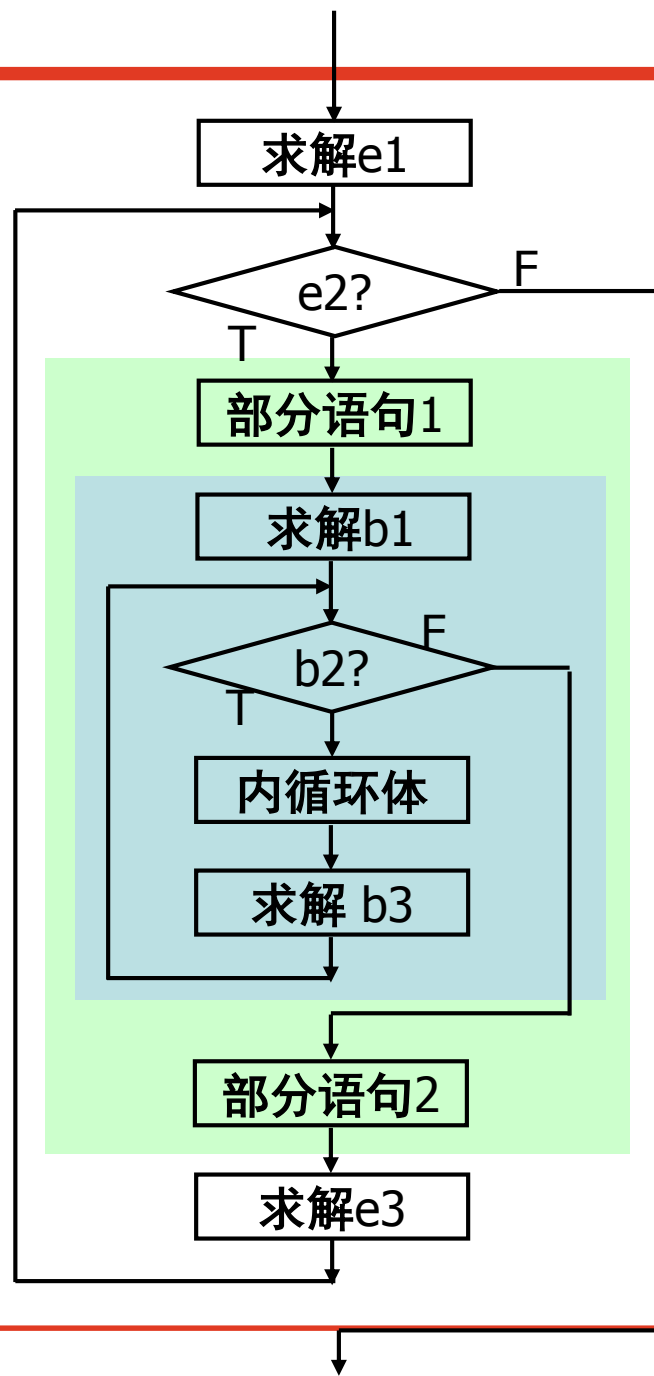
3.5.6 循环结构的嵌套

- ▶ 如果一个循环体内包含又一个循环语句时，就构成了循环的嵌套。
- ▶ C++语言的循环语句（while、do、for）可以互相嵌套，循环嵌套的层数没有限制，可以形成多重循环。
- ▶ 使用多重循环的时候，嵌套的循环控制变量不能相同。

3.5.6 循环结构的嵌套

例：for循环嵌套

```
for(e1; e2; e3)
{
    !
    for(b1; b2; b3)
    {
        !
    }
    !
}
```



循环语句的选用

- 1) 循环次数已知且循环操作规律，选用 for 方便；
- 2) 循环次数未知且循环操作里没有按一定规律变化的量，可采用 while 或 do-while 语句。

例：键盘输入一个偶数，非偶数则要求重输。

```
int main( )  
{ int n;  
  |  
  do { cout<<"Enter n:"<<endl; //循环次数不定  
      cin>>n;  
  }while(n%2!=0);  
  |  
}
```

3.5.7 循环结构程序举例

例：输入5个最多8位的正整数，将各数按位颠倒输出。

```
int main( )
{   int i,n,d;
    for(i=1; i<=5; i++) {           //循环次数确定
        cout<<"Enter a data(1~99999999):";
        cin>>n;
        while(n!=0) {               //循环次数不定
            d=n%10;
            cout<<d;
            n=n/10;
        }
    }
    return 0;
}
```

- ▶ 循环的分类（循环语句的选用）

- ▶ 1. 计数型循环

- ▶ 计数型循环用于处理已知循环次数的循环过程。
- ▶ 控制变量在每次循环时都要发生规律性变化（递增或递减），当控制变量达到预定的循环次数时，循环就结束。
- ▶ 计数型循环常使用for语句。

【例3.10】 求

$$\sum_{n=1}^{10} n!$$

相当于：求 $S=1!+2!+3!+\dots+10!$

3.5.7 循环结构程序举例

例3.10 求 $S=1!+2!+3!+\dots+10!$

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int s, n, t;
6      for(s=0, t=1, n=1; n<=10; n++)
7          t = t * n, s = s + t; //t为n!
8      cout<<s<<endl;
9      return 0;
10 }
```

▶ 2. 条件型循环

- ▶ 条件型循环用于处理循环次数未知的循环过程，称为“不定次数循环”。
- ▶ 在条件型循环中，由于事先不能准确知道循环的次数，因此循环控制是由条件来判定的。在每次循环时检测这个条件，当条件一旦满足，循环就结束。
- ▶ 条件型循环常使用while语句和do while语句。

3.5.7 循环结构程序举例

例3.11 求 π 的近似值 $\pi/4 \approx 1 - 1/3 + 1/5 - 1/7 + 1/9 - \dots$ ，直到最后一项的绝对值小于 10^{-7} 为止。

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4  int main()
5  {
6      double s=1, pi=0, n=1, t=1;
7      while(fabs(t)>1e-7)
8          pi=pi+t, n=n+2, s=-s, t=s/n;
9      cout<<pi*4<<endl;
10     return 0;
11 }
```

3.5.7 循环结构程序举例

例3.12 从键盘输入一行字符，直到输入回车时结束输入。统计其中的字母、数字和空格个数。

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int c, a=0, n=0, s=0;
6      while((c=cin.get()) != '\n')
7          if((c>='A' && c<='Z') ||
8              (c>='a' && c<='z')) a++;
9          else if (c>='0' && c<='9') n++;
10         else if (c==' ') s++; //空格
11         cout<<a<<"", "<<n<<"", "<<s<<endl;
12         return 0;
13     }
```

▶ 3. 枚举算法

- ▶ 枚举法，也称为穷举法，是指从可能的集合中一一枚举各个元素，用给定的约束条件判定哪些是无用的，哪些是有用的。能使命题成立者，即为问题的解。
- ▶ 采用枚举算法求解问题的基本思路为：
 - ▶ （1）确定枚举对象、枚举范围和判定条件；
 - ▶ （2）一一枚举可能的解，验证是否是问题的解。

【例3.14】百钱买百鸡问题：有人有一百块钱，打算买一百只鸡。公鸡一只5元，母鸡一只3元，小鸡3只1元，求应各买多少？

3.5.7 循环结构程序举例

例3.13 三重循环实现百钱百鸡问题。

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int x, y, z;
6      for (x=0; x<=20; x++)
7          for (y=0; y<=33; y++)
8              for (z=0; z<=100; z++)
9                  if (z%3==0&& x+y+z==100&&
10                      5*x+3*y+z/3==100)
11                      cout<<"公鸡="<<x<<"", 母鸡="<<y
12                      <<"", 小鸡="<<z<<endl;
11      return 0;
12  }
```

3.5.7 循环结构程序举例

例3.13 二重循环实现百钱百鸡问题。

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int x, y, z;
6      for (x=0; x<=20; x++)
7          for (y=0; y<=33; y++) {
8              z=100-x-y;
9              if (z%3==0 && 5*x+3*y+z/3==100)
10                 cout<<"公鸡="<<x<<", 母鸡= "
11                    <<y<<", 小鸡="<<z<<endl;
12         }
13     return 0;
14 }
```


▶ 4. 迭代算法

- ▶ 迭代法是一种不断用变量的旧值递推新值的求解方法。
- ▶ 采用迭代算法求解问题的基本思路为：
 - ▶ （1）确定迭代变量。
 - ▶ （2）建立迭代关系式。
 - ▶ （3）对迭代过程进行控制。

【例3.14】 求斐波那契(Fibonacci)数列前40个数。斐波那契数列公式为：

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

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

$$f(n) = f(n-1) + f(n-2) \quad (n > 2)$$

确定迭代变量：f(n)

建立迭代关系式：f(n)=f(n-1)+f(n-2)

对迭代过程进行控制：Fibonacci数列的前40个数

3.5.7 循环结构程序举例

例3.14

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i, f1=0, f2=1, fn; //迭代变量
6      for(i=1; i<=40; i++) { //迭代次数
7          fn = f1 + f2; //迭代关系式
8          f1 = f2 , f2 = fn; //f1和f2迭代前进
9          cout<<f1<<endl;
10     }
11     return 0;
12 }
```

CP[®]程序设计