第三章　Java面向过程编程

3.1 Java的顺序结构

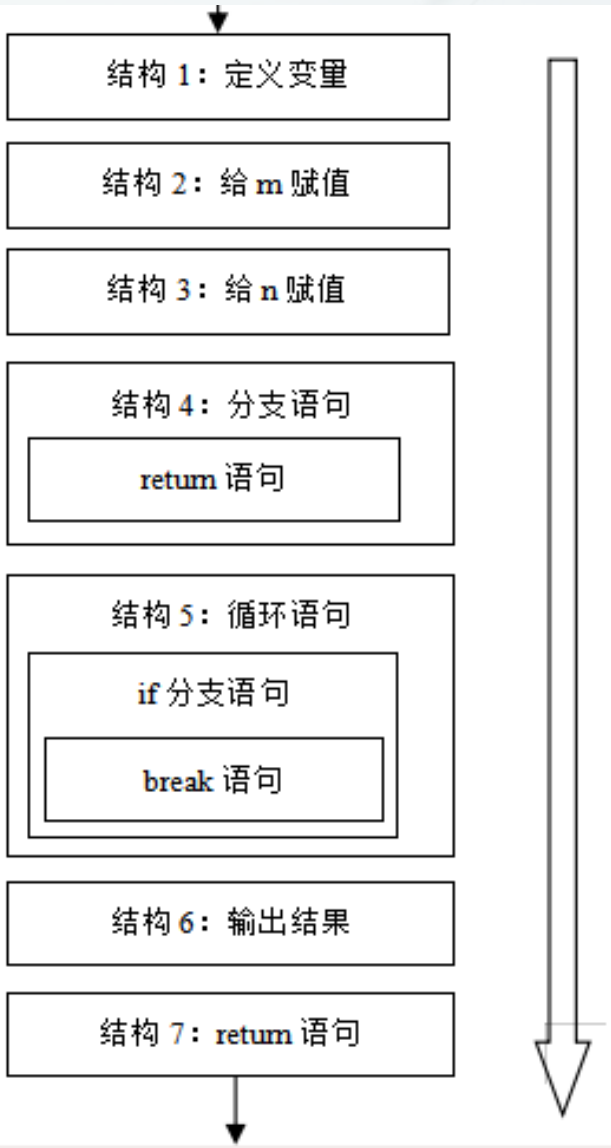
面向过程编程中，我们采用结构化的方式进行编程，每个结构具有一个入口和一个出口，按照解决问题的步骤一个一个结构顺序地执行，直到程序结束。这些结构内部包括各种基本语句，比如定义变量、变量赋值、分支、循环、输入/输出等，或者包括这些语句的嵌套。

相对于分支结构和循环结构，顺序结构主要是指定义变量、变量赋值、表达式运算、输入/输出等语句。这里用一个C语言的程序示例来说明。该程序的功能是求两个正整数的最大公约数。过程描述：从键盘输入2个整数，判断是否为正数，如果否，则结束函数，返回0；如果都为正数，则使用循环结构来求两个数的最大公约数，最后输出结果。

程序示例 3-1　从键盘输入两个整数，求这两个数的最大公约数(C 语言程序)。

程序段(CommonDivisor.c)

```
int main(){
    int m,n,i;                          1
    scanf("%d",&m);                     2
    scanf("%d",&n);                     3
    if(m<=0 || n<=0)                    4
        return 0;
    for(i=m;i>=1;i--)                   5
        if(m%i==0 && n%i==0)
            break;
    printf("result:%d\n",i);            6
    return 0;                           7
}
```



★ 与C语言的比较：

(1) C语言的变量定义都要在程序开始处，而Java可以在程序需要的地方定义变量。

(2) C语言的输入/输出使用scanf()和printf()，而Java的输入使用Scanner类对象及相应的成员函数进行，输出使用System()。

(3) C语言的main函数使用return语句来结束函数运行，而Java使用“System.exit(0);”语句来终止程序。

程序示例 3-2　从键盘输入两个整数，求这两个数的最大公约数(Java 程序)。

程序段(CommonDivisor.java)

```
import java.util.*;
public class CommonDivisor {
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int m = sc.nextInt();
        int n = sc.nextInt();
        if(m<=0 || n<=0)
            System.exit(0);                          如果 m 或 n 小于 0，退出程序
        int i;
        for(i=m;i>=1;i--)                            使用循环寻找最大公约数
            if(m%i==0 && n%i==0)
                break;
        System.out.println("result:" + i);
    }
}
```

程序结果：

General Output

```
--------------------Configuration:
35
14
result:7

Process completed.
```

3.2 Java的分支结构

分支结构主要有单分支、双分支和多分支三种结构，程序运行到该结构时根据分支条件来判断走哪条“路”：

(1) 单分支：分支条件成立，执行分支语句，否则不执行。

(2) 双分支：分支条件成立，执行第一条分支语句，否则执行第二条分支语句。

(3) 多分支：从第一个分支条件开始自上向下判断分支条件，哪个分支条件成立就执行哪条分支语句，然后退出整个多分支结构。

3.2.1 if语句

if语句的三种结构形式如下：

(1) 单分支结构：

```
if(分支条件)
        语句/语句块;
```

(2) 双分支结构：

```
if(分支条件)
        语句/语句块 1;
else
        语句/语句块 2;
```

(3) 多分支结构：

```
if(条件 1)
        语句/语句块 1;
else if(条件 2)
        语句/语句块 2;
else if(条件 3)
        语句/语句块 3;
…
else if(条件 n-1)
        语句/语句块 n-1;
else
        语句/语句块 n;
```

下面以双分支的if-else结构为例进行说明，图3-1是双分支结构的流程图，程序从a进入到该结构碰到分支条件，当条件为真时执行S2语句(或语句块)，条件为假时执行S1语句(或语句块)，不管执行哪条语句，都要从b出口退出该分支结构。简单来说就是条件为真走S2这条“路”，条件为假走S1这条“路”，S1和S2这两条路，只能选择一条。

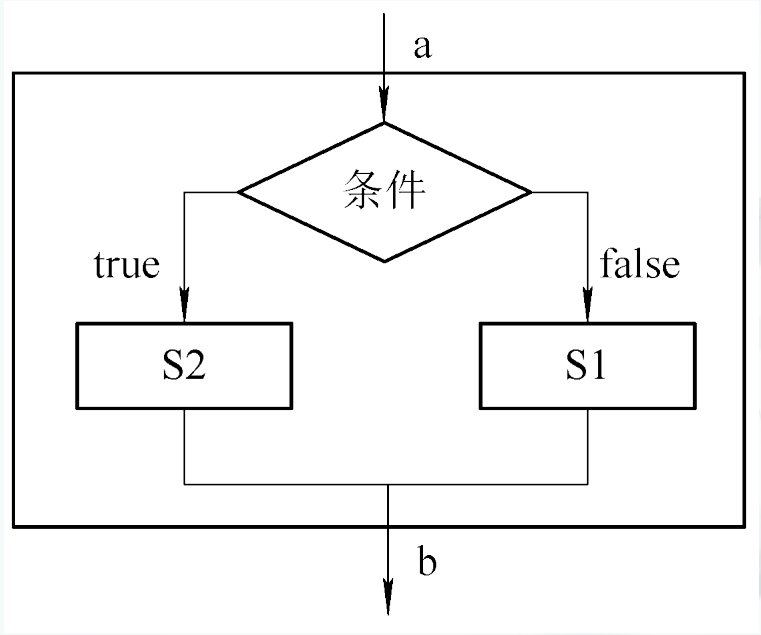


图3-1 选择结构

分支条件说明：

(1) 条件可以由关系表达式、逻辑表达式或布尔逻辑变量等构成。

(2) 关系表达式是由==、!=、<、>、>=、<=等这些关系运算符连接起来的运算式。

(3) 逻辑表达式是由逻辑非(！)、逻辑与(&&)、逻辑或(||)三个逻辑运算符连接起来的运算式。

各运算符的优先级别为：

！>算术运算符>关系运算符>&&>||>赋值运算符

(4) && 和 || 运算符的结合性是从左到右。

(表达式1)&&(表达式2)若表达式1为假，则表达式2不会被运行

(表达式1)||(表达式2)若表达式1为真，则表达式2不会被运行

(5) 0<x<10在数学中表示x大于0且小于10，但是在程序中表达这个条件时应该写为

x>0 && x<10

程序示例 3-3　从键盘输入一个整数，根据整数的正负输出对应字符串。

程序段(IfDemo1.java)

```
import java.util.*;
public class IfDemo1 {
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int m = sc.nextInt();
        if(m > 0){                                          分支条件
            System.out.println("m 是正数");                  分支语句1
        }else{
            System.out.println("m 是负数");                  分支语句2
        }
    }
}
```

程序结果：

General Output

```
--------------------Configuration:
请输入一个整数给m: 12
m是正数

Process completed.
```

General Output

```
--------------------Configuration:
请输入一个整数给m: -5
m是负数

Process completed.
```

但是当我们输入 0 时，结果 m 是负数。

General Output

```
--------------------Configuration:
请输入一个整数给m: 0
m是负数

Process completed.
```

程序示例 3-4　从键盘输入一个整数，根据整数的值输出正数、负数或 0。

程序段(IfDemo1.java)

```
import java.util.*;
public class IfDemo1{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("请输入一个整数给 m: ");
        int m = sc.nextInt();
        if(m > 0){
            System.out.println("m 是正数");          分支语句 1
        }else if(m < 0){
            System.out.println("m 是负数");          分支语句 2
        }else{
            System.out.println("m 为 0");           分支语句 3
        }
    }
}
```

(3) 不管使用哪种分支结构，程序的逻辑要正确，不要有漏洞。if只能管制一条语句，可以使用花括号将多条语句构成一个{语句块}归属if管制，if和else后面固定使用{ }包围分支语句(不管是一条还是多条)是一个良好的编程习惯。

3.2.2 switch语句

Java与C语言一样可以使用switch的多分支结构，switch语句的一般形式如下：

```
switch(表达式)
{
    case 值1：子句1；break；
    case 值2：子句2；break；
    …
    case 值n：子句n；break；
    default：子句m；
}
```

switch语法说明：

(1) switch语句中的表达式类型只能是byte、short、int、char和枚举等类型，在JDK 1.7后可以有string表达式类型。

(2) case后面的值1、值2、…、值n必须是整型、字符型常量以及字符串，各个case后面的常量值不能相同。

(3) switch语句的主要流程是把表达式的值依次与各个case子句中的值比较，如果值相等，表示匹配成功，找到对应的分支，执行该case后面的子句。

(4) 可以把switch后面的表达式看成选路的依据，case后面的值是分支路径的路标，如果表达式的值与路标相同，表示找到了分支路径，开始执行该路径下的语句。

(5) 一般在每条分支最后都有break语句，作用是执行完一个case分支后，使程序跳出switch语句，不再执行其它语句；如果某个子句后不使用break语句，则继续向下执行后面的语句，直至碰到下一个break或运行到switch的右花括号结束。

(6) 关于default语句，当switch表达式的值与所有case语句中的值都不匹配时，就会找到default，开始执行default分支的语句。

程序示例 3-5　从键盘输入一个百分制成绩，根据分数的值输出该分数所在等级。

程序段(SwitchDemo1.java)

```
System.out.println("请输入您的成绩(0-100)：");
Scanner sc = new Scanner(System.in);
int score = sc.nextInt();
switch (score/10) {                                    选路表达式
    case 10:
     case 9:                                           如果选路表达式的值为 10 或 9，从这里进入
        System.out.println("优秀");
        break;                                         执行完该分支就退出整个 switch 结构
    case 8:
        System.out.println("良好");
        break;
    case 7:
        System.out.println("中等");
        break;
    case 6:
        System.out.println("及格");
        break;
    case 5:     case 4:     case 3:     case 2:   case 1:     case 0:
        System.out.println("不及格");
        break;
     default :                                         如果所有 case 都不匹配，进入该分支
        System.out.println("输入错误！");
```

程序结果：

General Output

```
----------------------
请您的成绩(0-100)：
42
不及格

Process completed.
```

General Output

```
----------------------
请您的成绩(0-100)：
65
及格

Process completed.
```

General Output

```
----------------------
请您的成绩(0-100)：
88
良好

Process completed.
```

General Output

```
----------------------
请您的成绩(0-100)：
75
中等

Process completed.
```

General Output

```
----------------------
请您的成绩：
-30
输入错误！

Process completed.
```

General Output

```
----------------------
请您的成绩(0-100)：
95
优秀

Process completed.
```

3.3 循环结构

循环结构示意图如图3-2所示，当满足循环条件时就执行循环语句，执行完循环语句后，再进行条件判断，如果条件还为真，则继续执行循环语句，直到条件为假时退出循环体。

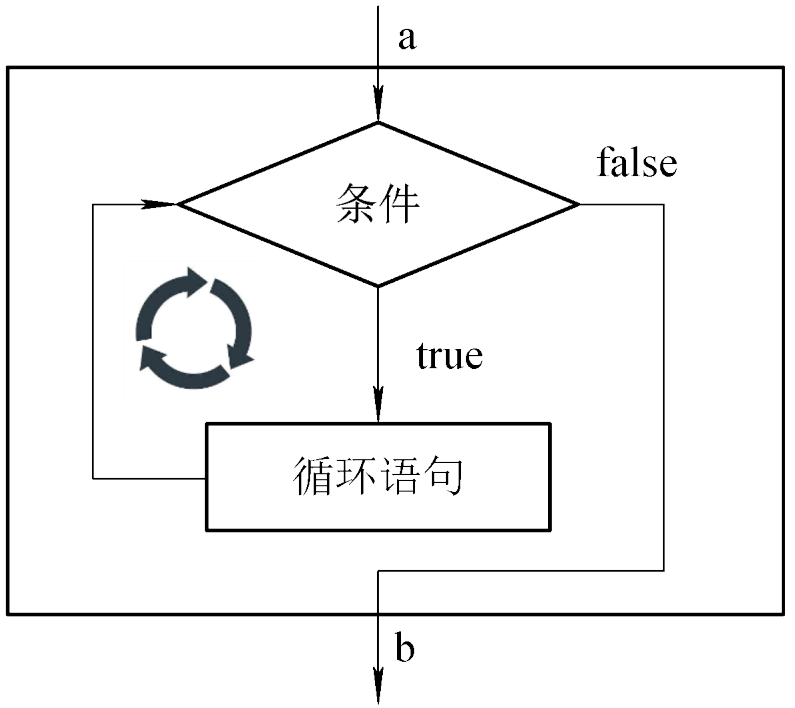


图3-2 循环结构

3.3.1 while循环结构

while循环结构，又称为当型循环：当条件成立时，进入循环体；当条件不成立时，退出循环体。当型结构如图3-2所示，一般结构形式如下：

```
while(循环条件)
    循环体语句;
```

程序示例 3-6　将 i 变量反复加入到 sum 中，完成 1+2+3+…+10 求和。

程序段(WhileDemo1.java)

```
int i = 1;
int sum = 0;
while(i <= 10){                                          循环条件
    sum = sum + i;
    i++;                                                 步进变量变化
    System.out.println("i = " + i);
}
System.out.println("sum = " + sum);
```

程序结果：

```
General Output
--------------------Configuration:
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
i = 11
sum = 55

Process completed.
```

3.3.2 for循环结构

如果清楚知道循环次数，或者循环的步进变化很明确，这时使用for循环更为方便。从for循环的头部就能很直观地读出循环的次数，其结构形式如下：

```
for(表达式 1; 循环条件; 表达式 2)
    循环语句;
```

for循环的结构如图3-3所示。

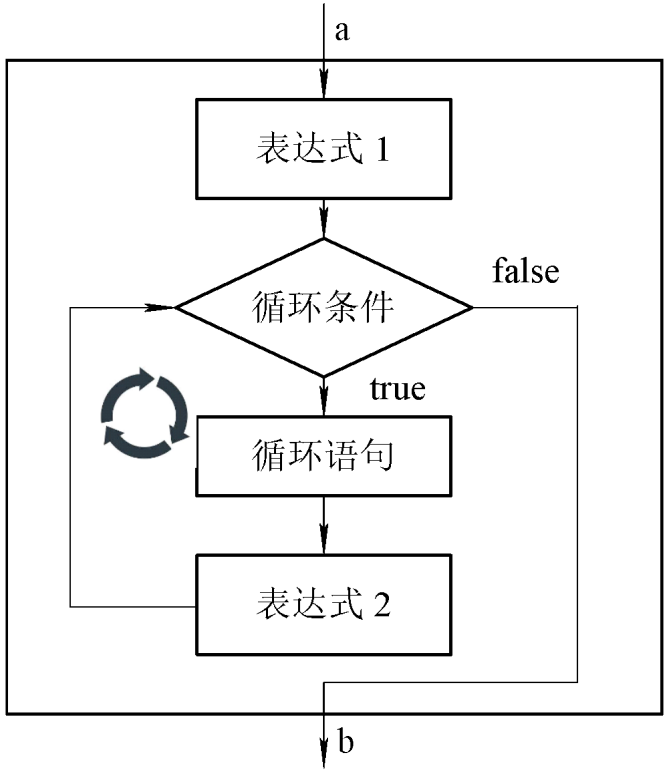


图3-3　for循环结构