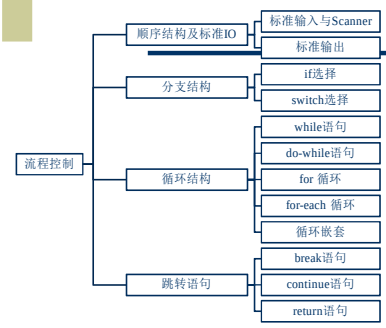
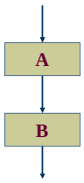
第三章 流程控制

第三章 流程控制

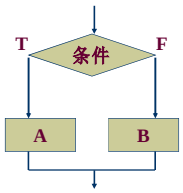


结构化程序设计的基本结构

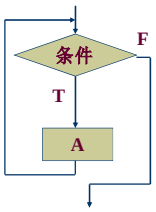
1、顺序结构



2、选择结构



3、循环结构



4、跳转结构

3.1 顺序结构及标准IO

- **顺序结构表示程序的执行顺序是按照语句编写的先后顺序执行的，是一种简单的程序结构。**
- **顺序结构的语句主要有：变量说明语句、赋值语句、方法调用语句等。下面以标准输入输出 I/O为例说明。**

标准I/O

简单的用于输入和输出的控制台就可以了

- 标准输入
  从键盘上读取数据，代表他的对象为 System.in
- 标准输出
  输出到显示器上，代表他的对象为 System.out
- 标准错误输出
  将错误**输出到显示器上，代表他的对象为** System.err

▪**the basic unit of standard I/O is byte**

System.in（了解）

- int read()
  return a byte(0-255) or -1
- int read(byte b[])
  return serveral bytes into b or -1
- int read(byte b[], int off, int len)
  return results into b from off to len or -1

输入

程序需要将字节数据解析成不同类型的数据进行处理，为了简化这部分工作，可以用Scanner对象将字节流转换成基本数据类型、大数类型或字符串类型。创建Scanner对象的形式为：

Scanner类的对象的定义形式：

```
Scanner sc=new Scanner(System.in);
```

sc中封装了系统资源：标准输入设备，用完后需要关闭流，方法为：

```
sc.close();
```

Scanner类中有较多的方法，常用的与基本数据类型和字符串有关的方法见表

表3-1 Scanner类中的常用方法

返回值类型	方法	作用
boolean	hasNext()	判断是否还有另一个标记存在
boolean	hasNextXxx()	检测输入源是否存在下一个“Xxx”表示的数据类型的数据，“Xxx”代表的数据类型可以是boolean、byte、short、int、long、float、double、BigInteger和BigDecimal等，“X”表示首字母为大写，例如haxNextInt()。
boolean	hasNextXxx(int radix)	“Xxx”表示数值类型，radix指定进制基数。
xxx	nextXxx()	读入类型为xxx的数据，可读入的类型与hasNextXxx方法相同，如nextDouble()
String	nextLine()	读入一行字符
String	next()	读入一个字符串

封装输入后的使用

```
import java.util.Scanner;
public class test{
 public static void main(String arg[]){
   Scanner sc=new Scanner(System.in);
   int  in1=sc.nextInt();
   double dou1=sc.nextDouble();
   boolean bool1=sc.nextBoolean();
   String str1=sc.next();
   sc.nextLine();//清除上次输入仍遗留在缓冲中的行结束符
   String str3=sc.nextLine();
   System.out.println(in1+" "+dou1+"  "+bool1+"  "+str1) ;
   System.out.println(str3);
   sc.close();
}}
```

【运行结果】：

```
123 45.67 true scannerTest
str3 test
123 45.67 true scannerTest
str3 test
```

标准输出System.out

基本数据类型可以用PrintStream类中的方法完成输出。类中有三种常用的方法：

- print(parameter)

  打印一行文本，但不换行
- println(parameter)

  打印一行文本并换行

参数：boolean ,char ,double,float ,int,long,String ,char[],object

```
System.out.println("this is a test");
System.out.println("----"+3.5+true+'m'+" end .");
System.out.println();
```

```
this is a test
----3.5truem end .
```

printf(String format, Object... args)

“format”是用于控制后面输出项的字符串，“args”是个数可变的输出数据。

“format”的格式如下：

[普通字符]%[**标志字符**][**输出宽度**][.**小数位数**]**格式控制字符**[普通字符]

其中的“%”和“格式控制字符”必须有，其余的可有可无。“普通字符”按原样输出。

“格式控制字符。

标志字符	作用
-	输出数据左对齐
#	以八或者十六进制输出，数据前会加0或0x
+	强制输出符号+ 或-
‘’	输出正数时前面有一个空格
0	输出数据达不到指定宽度则用0填充
,	千分位（整数部分）
（	如果输出是负数，则给负数加括号但不显示负号

格式控制字符

控制字符	描述
B或b	以字符串形式显示的布尔类型,b表示小写，B表示大写的，如果这个位置上的参数输入为空显示false，如果不是boolean或Boolean也不为空，则显示true
H或h	数据以十六进制字符串显示
S或s	字符串，输出字符串
C或c	字符，输出字符
d	整型，输出十进制整型数
o	整型，输出八进制整型数
X或x	整型，输出十六进制整型数
E或e	浮点型数，数据以指数形式输出
f	浮点型数，数据以十进制浮点型输出
G或g	浮点型数，根据数据的大小和精度，自动选择合适的格式输出
A或a	浮点型数，十六进制P指数法输出
n	换行符
T或t	输出Date或Time的字符串形式
%	百分符

```
public class Test{
 public static void main(String arg[]){
int a=123,b=456;
  System.out.printf("1:a>b=%b,a<b=%B%n",a>b,a<b);//注意换行
  System.out.printf("2:%-8d,%8d\n",a,b);
  double c=123.0;
  System.out.printf("0x1.9p6=%-8.2f\n",0x1.9p6);
  System.out.printf("0x1.9p6=%-5a\n",100.0);
  System.out.printf("the result is %.3f",c);}
}
```

```
1:a>b=false,a<b=TRUE
2:123     ,     456
0x1.9p6=100.00
0x1.9p6=0x1.9p6
the result is 123.000
```

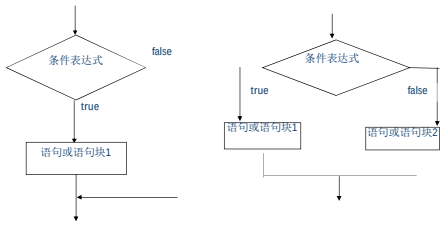
- **分支语句**
  1. if-else语句
  2. switch语句
- **循环语句**
  1. while 语句
  2. do-while 语句
  3. for 语句
  4. for each
- **跳转语句**
  1. break语句
  2. continue语句
  3. return 语句

15

3.2 选择语句

IF 选择

1、if和if-else语句



例

【例3.2】闰年出现的规律是：四年一闰；百年不闰，四百年再闰。根据输入的年份x判断是否为闰年，也就是当年份满足下列两个条件之一，则结果为闰年。

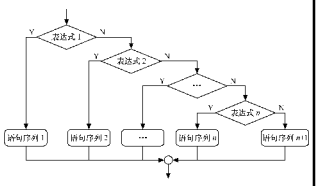
- x可以被4整除，但不能被100整除
- x可以被400整除

```
import java.util.Scanner;
public class leapYear{
  public static void main(String arg[]){
    Scanner sc=new Scanner(System.in);
    System.out.println("请输入年份");
    int year=sc.nextInt();
    c.close();
    if ((year%4==0)&&(year%100!=0)|| year %400==0)
         System.out.printf("%d是闰年", year);
    else
         System.out.printf("%d是平年",year);
}}
```

2．if语句的嵌套

如果if或else的子句还包含if或if-else语句，形成多分支结构。
多分支的结构很多，如：

```
if（布尔表达式1）
 语句1;
 else if（布尔表达式2）
     语句2;
     …
     else if （布尔表达式n）
      语句n;
          else 语句n+1;
```



求三个数中的最值

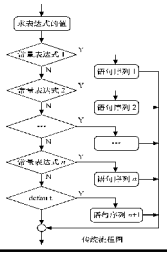
```
import java.util.Scanner;
public class maxNumber{
  public static void main(String arg[]){
     double a,b,c,max ;
     System.out.println("please input three numbers ");
     Scanner sc=new Scanner(System.in);
     a=sc.nextDouble();
     b=sc.nextDouble();
     c=sc.nextDouble();
     sc.close();
     if (a>b)
          if (a>c)
               max=a;
          else
               max=c;
     else
           if(b>c)
               max=b;
           else
               max=c;
        System.out.println("the max number is :"+max);}}
```

用关系运算代替：
max=a>c?a:c;
max=b>c?b:c。

switch 语句

当判断条件较多时，用嵌套的if/else结构显得有些笨拙，这时可以用switch语句实现多重选择判断。

```
switch （表达式）
 { case 常量1: 语句块1; [break;]
   case 常量2: 语句块2; [break;]
   …
   case 常量n: 语句块n ; [break;]
   [default : 默认处理语句]
 }
```



Switch语句需要注意的问题：

1.表达式的值类型：char, byte, short, int或者其对应的封装类(核心类一章)以及 枚举类型，Java7中，增加了对String 的支持。

2、各个case后面的常量不能有相同的值。

3、语句块可以是多条语句，不必使用大扩号。

4、default子句是可选的。

5、switch语句的每一个case判断，都只负责指明流程分支的入口点，而不负责指明分支的出口点，分支的出口点用break来标明。我们可以利用这一点来用同一段语句处理多个case条件 。

```
public class VowelsAndConsonants{
  public static void main(String[ ] args) {
    for (int i=0; i<5; i++) {
      char c=(char)(Math.random( )*26+'a');
      System.out.print(c+":");
      switch (c){
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u': : System.out.println("vowel");//元音
                  break;
        case   'y':
        case 'w': System.out.println("Sometimes a vowel");
                  break;
        default: System.out.println("consonant"); //辅音
      } ;
} } }
```

Java7中对switch的新增特性：支持String类型

```
import java.util.Scanner;
public class StrSwitch{
 public static void main(String arg[]){
   String result;
   for(String s:arg){
     switch(s){
          case "Mon": result="周一"; break;
          case "Tue": result="周二"; break;
          case "Wed": result="周三"; break;
          case "Thu": result="周四"; break;
          case "Fri": result="周五"; break;
          case "Sat": result="周六"; break;
          case "Sun": result="周天"; break;
          default:    result="未知描述";
        }
     System.out.printf("%s is %s\n",s,result);
 } } }
```

注意：传给switch 的String变量不能为null，同时case中的字符常量也不能为null

因为，编译时是通过求字符串的hashCode，转变成整数，如果这个字符串是空的，没办法调用hashcode函数，会出现异常。

枚举类型

```
enum LightColor {red, yellow, green}

public class EnumDemo{
public static void main(String arg[]) {
 LightColor lc=LightColor.red;
 switch(lc) {
 case red:    System.out.println("亮红灯");
              break;
 case yellow: System.out.println("亮黄灯");
              break;
 case green:  System.out.println("亮绿灯");
              break;
} }}
```

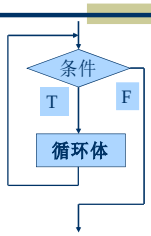
注意：switch 中每个case后面的枚举类型值不能加前缀类型 LightColor. 可由switch表达式确定。否则编译不通过

3.3 循环结构

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

3.3.1 while

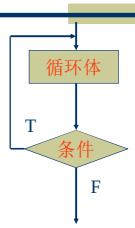
while（表达式）
{
循环体
}
注意:需要一个循环控制变量



3.3.2 do-while语句

do
{循环体 }
while（条件表达式）；

Note:1 至少执行一次
2 需要一个循环控制变量

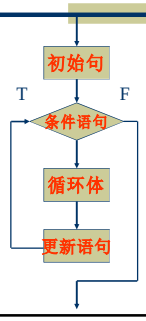



```
猜数字：猜测系统随机生成的数字，直到猜出为止。
import java.util.Scanner;  //加载包
public class Guess {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
int number = (int) (Math.random() * 10);
int guess;
do {
System.out.print("猜数字（0 ~ 9）:");
guess = scanner.nextInt();
} while(guess != number);
System.out.println("数字为："+number);
}}
```

3.3.3、for 语句

1、for（初始句；条件语句；更新语句）
{循环体 }

Note:
- **初始语句和更新语句可不止一个表达式**
  for (i=1,j=10；i<j；i++,j--) { …… }
- **三个表达式都可以为空（无穷循环）**



求Fibonacci数列前40项。

Fibonacci 数列：1，1，2，3，5，8，13，21，34，…

从数列看到：F1=1 (n=1)

F2=1 (n=2)

$Fn=F_{n-1}+F_{n-2}$ (n≥3)

```
public class Fibonacci{
    public static void main(String args[]){
          long f1=1,f2=1;
          for(int i=1;i<=20;i++){
               System.out.printf("%-10d %-10d", f1, f2);
             f1=f1+f2;
               f2=f1+f2;
               if(i%5==0)
               System.out.println();
          }
    }
}
```

Jdk1.5版本对for语句的功能给予扩充

2、 for(声明循环变量：数组或可迭代集合)
{循环体 }

- **for语句翻译成对于循环变量依次取数组的每一个元素的值。**
- **“声明循环变量”，必须是变量声明，不可以使用已经声明过的变量。**

例如：

```
for(String str1:arg)
 System.out.println(str1);
```

```
public class forDemo{

public static void main(String arg[])
{
 int a[]={1,2,3,4};
 char b[]={'a','b','c','d'};
 for(int i=0;i<a.length;i++) //传统方式
  System.out.println(a[i]);

 for(int i:a)   //for-each方法
  System.out.println(i);

 char c;
 for(c:b)  System.out.println(c);//错误方式
}}
```

3.3.5 循环嵌套

- 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。上述三种循环（while循环，do-while循环和for循环）语句之间可以相互嵌套使用。

【例：3.11】百鸡问题

已知公鸡5元1只，母鸡3元一只，小鸡1元3只，要求用100元刚好买100只鸡，问有多少种采购方案。

分析：设变量I、J、K分别代表公鸡数、母鸡数及小鸡数，则I+J+K=100（只）应是满足的第一个条件。要满足的第二个条件是：5I+3J+K/3=100（元），若用100元全部买公鸡，最多只能买20只，若全部买母鸡最多只能买33只，况且在已确定了购买的公鸡数后，母鸡最多还不能买33只，应扣除相应的公鸡数。

```
public class Loop_Loop3{
    public static void main(String args[]){
          int I,J,K;
          System.out.println(" 公鸡I 母鸡J  小鸡K");
          for(I=0;I<=20;I++) // I为公鸡数
          {
                for(J=0;J<=33;J++) // J为母鸡数
                {
                  K=100-I-J; // K为小鸡数
                  if(5*I+3*J+K/3.0==100)
                  System.out.println(" "+I+" "+J+" "+K);
                }
}}}
```

【运行结果】

```
公鸡I 母鸡J 小鸡K
 0  25  75
 4  18  78
 8  11  81
 12  4  84
```

3.3.4 跳转语句

1 continue

2 break

3 return

跳转语句用于中断控制流程，实现程序执行流程的转移。Java的跳转语句有：

- break、
- continue
- return。

1 break statement

- **强制退出循环，不执行循环中剩余的语句。**
- **退出switch语句。**
- **在循环结构中的使用形式：**

1、不带标号

break;

从最内层的循环体中跳转出来，转去执行该层循环体后面的语句

2、带标号

break 标号；

需要结束嵌套循环中的外层循环，可以使用带标号的break语句。表示，从标号所对应的循环中跳转出来。标号标记方式为：

标号：循环语句块

- 求一个数a的最大真因数。
- 分析：算法以a-1为起点，从大到小进行枚举，直到找到能整除a的数，则用break退出循环。

```
import java.util.Scanner;
public class MaxDiv{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
          sc.close();
        int i=a-1;
           while(i>1){
             if(a%i==0){
                System.out.print(a+"的最大真因数为："+i);
                break; //不带标号的break，用于结束本层循环
              }
             i--;
           }
    if(i==1)System.out.print("没有真约数");
    }
}
```

【例】在二维数组中查找从键盘输入的待查数据，并返回首次查到的位置

```
import java.util.Scanner;
public class LabelBreak {
        public static void main(String arg[]) {
                int[][] intArr={{8,7,4,18,14},{7,6,14,16,15},{4,13,8,2,13}};
                int i=0,j=0;
                System.out.println("please input the checked digital [0 20)");
                Scanner sc=new Scanner(System.in);
                int checkIn=sc.nextInt();
                sc.close();
                outLoop:for(i=0;i<3;i++)
                        for(j=0;j<5;j++)
                           if(intArr[i][j]==checkIn)
                           {
                             System.out.printf("It is in intArr[%d][%d]",i,j);
                              break outLoop;
                                    }
                if(i>=3&&j>=5) System.out.println("not in the array");
        }
   }
```

2、continue 语句

continue语句只能在循环语句中使用。

continue语句的作用是中断当前一次执行的循环体操作，开始下一次迭代。两种形式：

（1）终止当前这一轮的循环，进入当前循环的下一轮

continue

（2）跳过本次循环的剩余语句，跳到有标号标记的外层循环的下一次循环

continue 标号；

```
/* 打印图形：
####
####
####
####
####        */
public class ContDemo{
public static void main(String args[ ]){
int x;
for(x=1;x<21;x++){
System.out.print("#");
if( x % 4 != 0 )
continue;
System.out.println();
}
}
}
```

```
/*利用穷举法，找1~100之间的素数   */
public class PrimeDemo{
   public static void main(String args[]){

loop1:  for(int i=2;i<100;i++){
                for(int j=2;j<=Math.sqrt(i);j++){
                        if((i%j)==0)
                                continue loop1;
                 }
                System.out.print (i+" ");
      }
}}
```

3 return

结束一个方法，并返回一个值（如果有的话）

- return [表达式];

```
public class returnDemo{
   public static int Max(int a, int b){
              return  a>=b?a:b;
  }
   public static void main(String args[]){
     System.out.println("the max value is:"+Max(17%3, 19%3));
  }
}
```

总结:

- 标准IO及Scanner类的基本用法。
- 掌握选择语句if、switch的用法、循环结构while、do_while、for及for-each的用法。
- 掌握跳转语句break、continue、return的用法。