

Funciones

Una función es un bloque de código que realiza una determinada tarea. Las funciones son los elementos básicos de la programación estructurada. En el lenguaje C las funciones se declaran, se definen y se llaman. En el lenguaje Java las funciones solamente se definen y se llaman.

Definición y llamada de una función

El nombre de las funciones comienza por letra minúscula y deben sugerir acciones (mover, calcular, etc.). La definición de una función tiene el siguiente formato:

```
tipo nombreFuncion(tipo parm1, tipo parm2, tipo parm3){
    //...sentencias
    return resultado;
}
```

Entre las llaves de apertura y cierre se coloca la definición de la función. *tipo* indica el tipo de dato que puede ser predefinido **int**, **double**, etc, o definido por el usuario, una clase cualquiera.

Para llamar a un función miembro o método se escribe

```
tipo retorno=nombreFuncion(arg1, arg2, arg3);
```

Cuando se llama a la función, los argumentos *arg1*, *arg2*, *arg3* se copian en los parámetros *parm1*, *parm2*, *parm3* y se ejecutan las sentencias dentro de la función. La función finaliza cuando se llega al final de su bloque de definición o cuando encuentra una sentencia **return**.

Cuando se llama a la función, el valor devuelto mediante la sentencia **return** se asigna a la variable *retorno*.

Cuando una función no devuelve nada se dice de tipo **void**. Para llamar a la función, se escribe

```
nombreFuncion(arg1, arg2, arg3);
```

Una función suele finalizar cuando llega al final del bloque de su definición

```
void funcion(...){
    //sentencias...
}
```

Una función puede finalizar antes del llegar al final de su definición

```
void funcion(...){
    //sentencias...
    if(condicion) return;
```

```
//sentencias..  
}
```

Una función puede devolver un valor (un tipo de dato primitivo o un objeto).

```
double funcion(...){  
    double suma=0.0;  
    //sentencias...  
    return suma;  
}
```

Cualquier variable declarada dentro de la función tiene una vida temporal, existiendo en memoria, mientras la función esté activa. Se trata de variables locales a la función. Por ejemplo:

```
void nombreFuncion(int parm){  
    //...  
    int i=5;  
    //...  
}
```

La variable *parm*, existe desde el comienzo hasta el final de la función. La variable local *i*, existe desde el punto de su declaración hasta el final del bloque de la función.

El propósito de este capítulo es definir un conjunto de funciones dentro de una clase que describe una aplicación y llamarlas en el cuerpo de la función principal *main*. Veamos un ejemplo sencillo.

```
public class Funciones {  
  
    public static void main(String[] args) {  
        //llamada a las funciones  
        double s=sumar(3.2, 5.7);  
        System.out.println("la suma es "+s);  
        verMensaje("hola");  
    }  
    //definición de las funciones  
    static double sumar(double a, double b){  
        double resultado=a+b;  
        return resultado;  
    }  
    static void verMensaje(String texto){  
        System.out.println(texto);  
    }  
}
```

En este ejemplo, se han definido dos funciones, la primera *sumar* toma dos números *a* y *b* y devuelve su suma

```
double sumar(double a, double b){  
    double resultado=a+b;  
    return resultado;  
}
```

a y *b* son los parámetros, ambos de tipo **double**, *resultado* es una variable temporal, las tres desaparecen al finalizar el bloque de la definición de la función. Esta función se puede definir de la siguiente forma equivalente

```
double sumar(double a, double b){  
    return (a+b);  
}
```

La llamada a la función se realiza en el cuerpo de la función *main*.

```
double s=sumar(3.2, 5.7);
```

El parámetro *a* toma el valor 3.2, el parámetro *b* toma el valor 5.7, la función devuelve la suma 8.9 que se guarda en la variable local *s*. A continuación imprimimos el contenido de esta variable

```
System.out.println("la suma es "+s);
```

La definición de la función *verMensaje* es

```
void verMensaje(String texto){  
    System.out.println(texto);  
}
```

texto es un objeto de la clase predefinida *String* (en el siguiente capítulo hablaremos de clases y objetos), la función no devuelve nada es de tipo **void**.

Su llamada es

```
verMensaje("hola");
```

El objeto *texto* de la clase *String*, se inicializa con "hola" y se imprime

Como vemos una o varias funciones se pueden llamar desde otra función, en este caso *main*

```
public static void main(String[] args) {  
    //llamada a las funciones  
    double s=sumar(3.2, 5.7);  
    verMensaje("hola");  
}
```

Las funciones, *main*, *sumar* y *verMensaje* se dice que son funciones miembros de la clase *Funciones*. En Java no hay funciones que se puedan definir fuera de una clase. El prefijo **static** se explicará en las próximas páginas.

La función *main* no hay que llamarla explícitamente como *sumar* o *verMensaje*, se llama automáticamente cuando se corre una aplicación.

En los próximos apartados, veremos ejemplos de funciones, cómo se definen y cómo se llaman.

El valor absoluto de un número.

- Nombre de la función: *absoluto*
- Parámetros: un entero
- Devuelve: un número entero positivo.

El valor absoluto de un número x es x si el número es positivo y $-x$ si el número es negativo. Una función *absoluto* que calcula el valor absoluto de un número recibirá el dato del número, y devolverá el valor absoluto del mismo. Supongamos que dicho dato es entero, la definición de la función será:

```
int absoluto(int x){
    int abs;
    if(x>0) abs=x;
    else abs=-x;
    return abs;
}
```

Podemos simplificar el código sin necesidad de declarar una variable temporal *abs*.

```
int absoluto(int x){
    if(x>0) return x;
    else return -x;
}
```

O bien,

```
int absoluto(int x){
    if(x>0) return x;
    return -x;
}
```

Una función similar nos hallará el valor absoluto de un dato de tipo **double**.

```
double absoluto(double x){
    if(x<0) return -x;
    return x;
}
```

En el lenguaje Java dos funciones pueden tener el mismo nombre absoluto, pero deberán tener distinto número de parámetros o de distinto tipo. Por ejemplo, la primera función tiene un parámetro de tipo **int**, y la segunda de tipo **double**.

La potencia de exponente entero de un número entero

- Nombre de la función: *potencia*
- Parámetros: la base y el exponente, ambos enteros
- Devuelve: el resultado, un número del tipo **long**.

Para hallar la potencia de un número se multiplica tantas veces la base como indica el exponente. Por ejemplo, para hallar la quinta potencia de 3, se escribirá

$$3^5=3*3*3*3*3$$

Podemos realizar este producto en un bucle **for**, de manera que en la variable *resultado* se vaya acumulando el resultado de los sucesivos productos, tal como se recoge en la Tabla, entre paréntesis figura el valor de *resultado* en la iteración previa, el valor inicial es 1.

iteración	valor de <i>resultado</i>
1ª	(1)*3
2ª	(1*3)*3
3ª	(1*3*3)*3
4ª	(1*3*3*3)*3
5ª	(1*3*3*3*3)*3

```
long resultado=1;
for(int i=0; i<5; i++)
    resultado*=3;
```

El siguiente paso es la generalización del proceso a un exponente positivo cualquiera y a una base entera cualesquiera. La variable *resultado* es de tipo **long**, porque al hallar la potencia de un número entero se puede sobrepasar el rango de dichos números.

```
long resultado=1;
for(int i=0; i<exponente; i++)
    resultado*=base;
```

Por último, le pondremos una etiqueta a esta tarea, asociaremos esta rutina al nombre de una función.

```
long potencia(int base, int exponente){
    long resultado=1;
    for(int i=0; i<exponente; i++){
        resultado*=base;
    }
    return resultado;
}
```

El factorial de un número

- Nombre de la función: *factorial*
- Parámetros: un entero **int**
- Devuelve: el resultado, un número positivo del tipo **long**.

Como ejemplos de las sentencias iterativas **for** y **while** ya tuvimos la ocasión de calcular el factorial de un número. Ahora se trata de poner las líneas de código dentro de una función para que pueda ser reutilizada.

```
long resultado=1;
while(numero>0){
    resultado*=numero;
```

```

        numero--;
    }

```

Es mucho más lógico definir una función que calcule el factorial del número *num*,

```

long factorial(int num){
    long resultado=1;
    while(num>0){
        resultado*=num;
        num--;
    }
    return resultado;
}

```

que repetir tres veces el código del cálculo del factorial para hallar el número combinatorio *m* sobre *n*, de acuerdo a la siguiente fórmula.

$$\binom{m}{n} = \frac{m!}{n!(m-n)!}$$

El numerador y el denominador del número combinatorio *m* sobre *n* se obtiene del siguiente modo:

```

long num=factorial(m);
long den=factorial(n)*factorial(m-n);

```

La clase *Matematicas*

Ahora solamente nos queda poner todas las funciones en el interior de una clase que describe una aplicación y le llamaremos *Matematicas*, anteponiéndole a cada una de las funciones la palabra reservada **static**.

```

public class Matematicas {
    static long factorial(int num){
        //...
    }
    static long potencia(int base, int exponente){
        //...
    }
    static double absoluto(double x){
        //...
    }
    static int absoluto(int x){
        //...
    }
    public static void main(String[] args) {
        //llamada a las funciones
    }
}

```

Llamada a las funciones miembro

Para llamar a las funciones miembro se escribe

- Hallar el número combinatorio m sobre n

```
int m=8, n=3;
System.out.print("El número combinatorio "+m+" sobre "+n);
long numerador=factorial(m);
long denominador=factorial(m-n)*factorial(n);
System.out.println(" vale "+numerador+" / "+denominador);
```

- Hallar la potencia de un número entero

```
System.out.print("La potencia 5 elevado a 4 ");
System.out.print("vale "+potencia(5, 4));
```

- Hallar el valor absoluto de un número

```
double y=-2.5;
System.out.print("El valor absoluto de "+y);
System.out.println(" vale "+absoluto(y));
```

```
public class Matematicas {
    static long factorial(int num){
        long resultado=1;
        while(num>0){
            resultado*=num;
            num--;
        }
        return resultado;
    }
    static long potencia(int base, int exponente){
        long resultado=1;
        for(int i=0; i<exponente; i++){
            resultado*=base;
        }
        return resultado;
    }
    static double absoluto(double x){
        if(x<0) return -x;
        return x;
    }
    static int absoluto(int x){
        if(x<0) return -x;
        return x;
    }
    public static void main(String[] args) {
        //llamada a las funciones
        //número combinatorio
        int m=8, n=3;
        System.out.print("El número combinatorio "+m+" sobre "+n);
        long numerador=factorial(m);
        long denominador=factorial(m-n)*factorial(n);
        System.out.println(" vale "+numerador+" / "+denominador);
        System.out.println("*****");

        //Potencia de un número
```

```
        System.out.print("La potencia 5 elevado a 4 ");
        System.out.print("vale "+potencia(5, 4));

//Valor absoluto de un número real
        System.out.println("");
        System.out.println("*****");
        double y=-2.5;
        System.out.print("El valor absoluto de "+y);
        System.out.println(" vale "+absoluto(y));

//Valor absoluto de un número entero
        System.out.println("");
        System.out.println("*****");
        int z=-2;
        System.out.print("El valor absoluto de "+z);
        System.out.println(" vale "+absoluto(z));
    }
}
```