Universidad Andrés Bello®
Conectar · Innovar · Liderar

FUNDAMENTOS DE PROGRAMACIÓN EN JAVA

Funciones

En programación, una función es un conjunto de líneas de código que realizan una tarea específica y puede retornar un valor. Una función puede ser invocada o llamada desde otras partes del programa tantas veces como se desee. Las funciones pueden tomar parámetros que modifiquen su funcionamiento. Las funciones son utilizadas para descomponer grandes problemas en tareas simples y para implementar operaciones que son comúnmente utilizadas durante un programa y de esta manera reducir la cantidad de código. Cuando una función es invocada se le pasa el control a la misma, una vez que esta finalizó con su tarea el control es devuelto al punto desde el cual la función fue llamada.

La sintaxis de una función es la siguiente:

```
SubAlgoritmo variable_retorno <- Nombre (Argumentos)
    //Instrucciones
FinSubAlgoritmo

Algoritmo ejemplo
    //Instrucciones y llamada a SubAlgoritmo
FinAlgoritmo
```

Los componentes indicados en la sintaxis anterior se explican de la siguiente manera:

- **variable_retorno:** si el subproceso o función, calcula y devuelve un valor, se debe colocar el nombre de la variable que se utilizará para almacenar ese valor; si el subproceso o función no devuelve nada se puede eliminar la variable con su flecha de asignación.
- **Nombre:** es el nombre que recibirá el subproceso, función o subalgoritmo.
- **Argumentos:** los argumentos son variables que requiere la función, utilizando comas para separarlos; si el subproceso o función no requiere argumentos puede dejarse en blanco, y de manera opcional omitir los paréntesis.

Por ejemplo, si debemos calcular el promedio de un alumno, podemos crear una función llamada Promedio, la cual estará encargada de recibir las 5 calificaciones del semestre, sumarlas y dividirlas para conseguir el promedio, el cual será retornado por la función.

Ejemplo 1: Creación de una función que permita obtener el promedio de 5 calificaciones

```
SubProceso promedio <- CalcularPromedio
     Definir n1, n2, n3, n4, n5 Como Entero;
     Definir promedio Como Real;
     Escribir "Ingrese nota 1: ";
     Leer n1;
     Escribir "Ingrese nota 2: ";
     Leer n2;
     Escribir "Ingrese nota 3: ";
     Leer n3;
     Escribir "Ingrese nota 4: ";
     Leer n4;
     Escribir "Ingrese nota 5: ";
     Leer n5;
     promedio<-((n1+n2+n3+n4+n5)/5);
FinSubProceso

Proceso Ejemplo1
     Definir prom Como Real;
     Definir x1, x2, x3, x4, x5 Como Entero;
     prom<-CalcularPromedio;
     Escribir "El promedio obtenido es: ", prom;
FinProceso
```

El ejemplo anterior corresponde a un subproceso que no recibe parámetros, siendo necesario solicitar el ingreso de cada nota de forma manual. Una alternativa a este modelo es entregar todas las notas como parte del subproceso. El ejemplo siguiente da cuenta de este caso.

Ejemplo 2: Creación de una función que permita obtener el promedio de 5 calificaciones, con recepción de parámetros

```
Funcion prom <- CalcularPromedio ( num1, num2, num3, num4,
num5 )
     Definir prom Como Real;
     prom<-((num1+num2+num3+num4+num5)/5);
FinFuncion

Proceso Ejemplo2
     Definir promedio Como Real;
```

```
     Definir x1, x2, x3, x4, x5 Como Entero;

     Escribir "Calculo de promedio";
     Escribir "Ingrese nota 1: ";
     Leer x1;
     Escribir "Ingrese nota 2: ";
     Leer x2;
     Escribir "Ingrese nota 3: ";
     Leer x3;
     Escribir "Ingrese nota 4: ";
     Leer x4;
     Escribir "Ingrese nota 5: ";
     Leer x5;

     promedio<-CalcularPromedio(x1,x2,x3,x4,x5);
     Escribir "El promedio obtenido es: ", promedio;
FinProceso
```

Una tercera opción para definir y acceder a funciones, es a través de sub algoritmos. En el ejemplo siguiente se usará la misma idea de ejemplos anteriores, pero sin entregar un valor de retorno.

Ejemplo 3: Creación de una función que permita obtener el promedio de 5 calificaciones, con recepción de parámetros

```
SubAlgoritmo CalcularPromedio (num1, num2, num3, num4, num5)
     Definir prom Como Real;
     prom<-((num1+num2+num3+num4+num5)/5);
     Escribir "El promedio calculado es: ", prom;
FinSubAlgoritmo

Proceso Ejemplo3
     Definir x1, x2, x3, x4, x5 Como Entero;
     Escribir "Calculo de promedio";
     Escribir "Ingrese nota 1: ";
     Leer x1;
     Escribir "Ingrese nota 2: ";
     Leer x2;
     Escribir "Ingrese nota 3: ";
     Leer x3;
     Escribir "Ingrese nota 4: ";
     Leer x4;
     Escribir "Ingrese nota 5: ";
     Leer x5;

     CalcularPromedio(x1,x2,x3,x4,x5);
 FinProceso
```

Los subprocesos sin parámetros se llaman desde el proceso principal simplemente por su nombre sin más argumentos; se pueden abrir y cerrarparéntesis, pero esto es opcional. En cambio, si el subproceso contiene parámetros, estos si deben especificarse.

Ejemplo 4: Desarrolle un algoritmo en pseudocódigo que retorne todos los números primos entre el 1 y el 30.

```
SubAlgoritmo resultado <- Primo ( num )
    Definir cantidadDivisores, cont Como Entero;
    Definir resultado Como Logico;
    cantidadDivisores <- 0;
    Para cont <- 1 Hasta num Hacer
        Si num % cont = 0 Entonces
            cantidadDivisores <- cantidadDivisores + 1;
        FinSi
    FinPara
    Si cantidadDivisores <= 2 Entonces
        resultado <- verdadero;
    Sino
        resultado <- falso;
    FinSi
FinSubAlgoritmo

Algoritmo PrimosDel1Al30
    Definir n Como Entero;
    Para n <- 1 Hasta 30 Hacer
        Si Primo(n) Entonces
            Escribir n;
        FinSi
    FinPara
FinAlgoritmo
```

Recursividad

Las funciones recursivas son aquellas que se llaman a sí mismas durante su propia ejecución. Ellas funcionan de forma similar a las iteraciones, pero es necesario de planificar el momento en que dejan de llamarse a sí mismas o se tendrá una función recursiva infinita.

Estas funciones se estilan utilizar para dividir una tarea en sub-tareas más simples de forma que sea más fácil abordar el problema y solucionarlo.

El siguiente ejemplo da cuenta de una función recursiva para el cálculo del factorial de un número entero.

Ejemplo 5: Desarrolle una función recursiva que calcule el factorial de un número.

```
SubProceso fact <- Factorial ( N )
     Definir fact Como Entero;
     Si N = 0 O N = 1 Entonces
          fact <- 1;
     SiNo
          fact <- N * Factorial(N-1);
     FinSi
FinSubProceso

Proceso Ejemplo5
     Definir num Como Entero;
     Escribir "Ingrese un número: ";
     Leer num;
     Escribir "El factorial de ",num," es ",Factorial(num);
FinProceso
```

El ejemplo anterior corresponde a una función recursiva con retorno. También es posible usar este mismo concepto sin necesidad de entregar un valor como resultado. Esto se refleja en el ejemplo siguiente.

Ejemplo 6: Desarrolle una función recursiva que simule una cuenta regresiva.

```
Funcion cuentaRegresiva (contador)
     Si contador > 0 Entonces
          Escribir contador;
          cuentaRegresiva(contador-1);
     SiNo
          Escribir "Lanzamiento!";
     FinSi
FinFuncion

Proceso Ejemplo6
     cuentaRegresiva(10);
FinProceso
```

La creación de funciones es esencial para programar en Java, puesto que estas hacen que el **código sea más simple y manejable**. Cuando se construye una función, ese fragmento de instrucciones podrá ser reutilizado las veces que sean necesarias a lo largo de todo el programa.

Java cuenta con una serie de funciones internas. También puedes hacer tus propias bibliotecas de funciones o hacer funciones para usos puntuales dentro de un programa en particular. Siendo posible **crear una función en Java,** para hacerlo de manera correcta, tienes que saber si esta va a devolver, o no, algún dato, puesto que de ello dependerá su estructura.

Cómo crear funciones que no tienen valores de retorno

Si quieres crear una función que no devuelve nada al programa principal y que tampoco hace uso de parámetros, es decir, no necesita datos para ejecutarse, la sintaxis es la siguiente:

static void nombre(){

bloque de código

}

Revisa el siguiente ejemplo para comprenderlo mejor:

```
class Main {

  //Dentro de la clase ppal. creamos una función que no retorna ningún valor
  static void frase() {
    System.out.println("Escribo desde la función 'frase'.");
  }

  public static void main(String[] args) {

    //usamos la función que hemos generado.
    frase();
  }
}
```

También podemos crear una función que no retorna valores, pero que hace una operación y recibe argumentos. Los argumentos son los datos que recibirá la función para cumplir con la tarea para la cual ha sido programada. La sintaxis sería:

static void nombre(parámetros){

bloque de código

}

El siguiente es un ejemplo sencillo:

```
class Main {

  //Creamos una función con dos parámetros de tipo entero que sumará estos
  static void sumar(int num1, int num2) {

    //se crea una variable interna que almacenará la suma de esos valores
    int suma = num1 + num2;
    System.out.println("La suma es: " + suma);
  }

  public static void main(String[] args) {

    //se invoca a la función 'sumar' y se le pasan argumentos para que se ejecute
    sumar(7, 20);
  }
}
```

Cómo crear funciones que devuelven un valor

- Para construir una función que retorne su contenido deberás usar la palabra reservada ***return*** en el cuerpo de la función. Estas pueden recibir parámetros o no, dependiendo de la necesidad del código. Siempre se debe indicar el tipo de dato que va a retornar. La sintaxis es así:

static tipo de dato que retorna nombre de función(){

bloque de código

return ;

}

Con el siguiente ejemplo, verás todo más claramente:

```
class Main {

   //Creamos una función con dos parámetros de tipo
   //entero que sumará estos y devolverá el resultado
   static int sumar(int num1, int num2) {

      //se crea una variable interna para retornar
      //el resultado
      int suma = num1 + num2;
      return suma;
   }

   public static void main(String[] args) {

      //se crea una variable local que recibirá
      //el contenido de la función
      int resultado;
      resultado = sumar(78, 2);
      System.out.println("La suma es: " + resultado);
   }
}
```

Es importante acotar que cuando se construyen funciones que van a ser invocadas desde la función ***main***, estas deben tener la palabra ***static*** como modificador de acceso, para que el programa se ejecute correctamente.

Anexo: Referencias

1.- Funciones / Subprocesos en PSeInt

Referencia: https://victomanolo.wordpress.com/funciones-subprocesos-en-pseint/#:~:text=Las%20funciones%20tambi%C3%A9n%20llamadas%20Subproceso,que%20se%20pueden%20invocar%20(ejecutar)

2.- Creación de funciones y procedimientos

Referencia: https://www.aprendeaprogramar.com/cursos/verApartado.php?id=2012

3.- Funciones recursivas

Referencia:
https://plataforma.josedomingo.org/pledin/cursos/programacion/curso/u32/

https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Java/Funciones