Preparador Informática

www.preparadorinformatica.com

TEMA 26 INFORMÁTICA / TEMA 28 S.A.I.

PROGRAMACIÓN MODULAR.
DISEÑO DE FUNCIONES.
RECURSIVIDAD. LIBRERÍAS.

TEMA 26 INF/ TEMA 28 SAI: PROGRAMACIÓN MODULAR. DISEÑO DE FUNCIONES. RECURSIVIDAD. LIBRERÍAS.

1. INTRODUCCIÓN

El reto de la informática es encontrar soluciones a problemas diarios de diferente tipo y complejidad. Las soluciones planteadas consisten en realizar una serie de tareas o instrucciones que conforman algoritmos, en un orden especificado y utilizando los recursos disponibles. Hoy se cuenta con ordenadores que realizan esta labor, pero se requiere que los algoritmos que ejecutan se traduzcan a un lenguaje especial, lo que se conoce como programación.

En sus inicios la programación clásica tenía como objetivo principal que el programa funcionase, no siendo el mantenimiento un aspecto importante. Esto generaba extensos programas desarrollados en bloque, sin orden ni estructura (código "espagueti"), lo que dificultaba enormemente su comprensión y mantenimiento.

La necesidad de reducir los costes asociados a las tareas de mantenimiento de las aplicaciones dio lugar a la aparición de nuevos paradigmas de programación que facilitasen las tareas de mantenimiento. Con este propósito, surgieron dos paradigmas de programación:

- **Programación estructurada:** técnica constructiva de programas basada en un conjunto de estructuras específicas que facilitan la modificación, reutilización y lectura del código.
- **Programación modular** consiste en la división de un problema complejo en varios problemas más sencillos.

Ambos paradigmas no son opuestos, sino que se complementan. De este modo, la programación modular tiende a dividir un programa en partes más pequeñas, llamadas módulos, y la programación estructurada se encarga de desarrollar estructuradamente cada una de estas partes, facilitando así su legibilidad y mantenimiento.

El presente tema está dedicado al estudio de la técnica de programación modular, detallando el diseño de funciones y la importancia de la recursividad y las librerías.

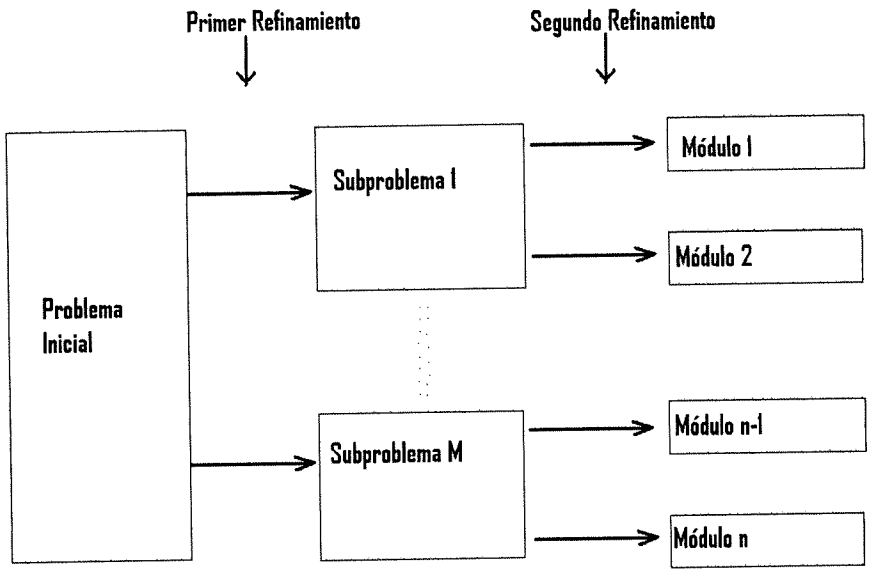
2. PROGRAMACIÓN MODULAR

La programación modular consiste en dividir un programa en unidades de menor tamaño en donde cada una de estas partes realiza una tarea explicita y única. Cada uno de estas partes recibe el nombre de módulo.

Las principales características de la programación modular son:

- **Facilita el trabajo en equipo,** ya que cada programador puede desarrollar un módulo diferente, y posteriormente enlazarlos.
- **Facilita la depuración**, ya que es más sencillo detectar y localizar errores en un único módulo que en el programa completo.
- **Facilita la reutilización**, ya que un módulo lo podemos usar en diferentes programas directamente.
- **Facilita el mantenimiento**, ya que es más sencillo modificar un único modulo que el programa completo.
- **Facilita la creación de librerías**, ya que los módulos que consideremos de uso común para nuestros programas podemos incluirlos en librerías, para posteriormente usarlos en futuros desarrollos.

El diseño modular y descendente resuelve un problema efectuando descomposiciones en otros problemas más sencillos a través de distintos niveles de refinamiento.



La utilización de la programación modular tiene los siguientes objetivos básicos:

- Simplificación del problema.
- Las diferentes partes del problema pueden ser programadas de modo independiente.
- El programa queda estructurado en forma de bloque lo que hace más sencilla su lectura y mantenimiento

La utilidad principal de la programación modular, es la de conseguir que los programas sean más legibles y, por tanto, más fáciles de depurar y modificar. De este modo, un programador que no haya realizado un código podrá fácilmente: entender su diseño, corregir posibles errores y ampliar el diseño original.

Con la utilización de esta técnica de diseño, surgen los conceptos de:

- Programa principal y subprogramas (internos o externos)
- Identificadores globales y locales.

2.1. PROGRAMA PRINCIPAL Y SUBPROGRAMAS

El problema principal se soluciona por el correspondiente programa o algoritmo principal y la solución de los subproblemas mediante los subprogramas, conocidos como funciones.

La estructura de un subprograma es básicamente la de un programa, y su función es resolver de modo independiente una parte del problema.

Cuando el subprograma figura físicamente en el mismo archivo fuente que el principal, se le denomina **subprograma interno**. En cambio, si el subprograma figura separado del programa principal en otro archivo fuente, se le denomina **subprograma externo**.

2.2. ÁMBITO DE UN IDENTIFICADOR

Los identificadores u objetos (variables, constantes, etc.) se pueden clasificar según su ámbito. Se conoce como ámbito de un identificador a la parte del programa desde donde este puede ser identificado porque se conoce su existencia.

De este modo, se distinguen:

- **Identificadores locales**: son los definidos dentro de un subprograma y su ámbito es dicho subprograma, no siendo accesible fuera de él.
- **Identificadores globales**: son los definidos en el programa principal, por lo que su ámbito se extiende a todo el programa.

3. DISEÑO DE FUNCIONES

Una función es un subprograma que toma cero, uno o más valores de entrada (parámetros) y devuelve un resultado asociado al nombre de la función.

De forma general, se puede decir que existen dos tipos de funciones: las funciones propias del lenguaje de programación y las funciones definidas por el usuario. Las funciones definidas por el usuario necesitan ser declaradas para posteriormente poder ser llamadas (invocadas) tantas veces como se desee.

3.1 DECLARACIÓN

La declaración de una función es el primer paso que hay que llevar a cabo antes de poder utilizar una función. Consta de 2 partes bien diferenciadas:

- **Cabecera**: Aquí se indica un conjunto de posibles modificadores, el tipo de valor retornado, el nombre de la función y una lista de parámetros indicando el tipo asociado. Los parámetros de la declaración de la función se denominan parámetros formales, y son nombres de variables que solo se utilizan dentro del cuerpo de la función.
- **Cuerpo** de la función: son las instrucciones a realizar por la función.

Ejemplo de declaración de una función en Java:

```
// Función que calcula el factorial de un nº
public static int factorial(int n){
      int fact=1;
      for(int i=1;i<=n;i++)
            fact=fact*i;
      return fact;
}
```

3.2 INVOCACIÓN

La llamada a una función se realiza con el nombre de la función y entre paréntesis los valores que son la lista de parámetros actuales, identificándose uno a uno y en el mismo orden con los parámetros formales. Es decir, los parámetros actuales son los datos que se le pasan a una función cuando se invoca. Una función puede ser llamada de la siguiente forma:

```
nombre_funcion (lista parámetros actuales)
```


3.3. PASO DE PARÁMETROS

Existe una correspondencia automática entre los parámetros formales y actuales cada vez que se llama a una función y/o procedimiento. Los parámetros actuales sustituirán y serán utilizados en lugar de los parámetros formales.

Existen 2 métodos para establecer el paso entre parámetros:

- **Paso por valor**: Los parámetros formales reciben una copia de los valores de los parámetros actuales. De modo que los cambios que se realicen dentro del subprograma no alterarán el valor que tenga desde el programa principal.

- **Paso por referencia**: en este método se envía al subprograma la dirección de memoria del parámetro actual y, por tanto, es una variable compartida pudiendo ser modificada directamente por el subprograma.

4. RECURSIVIDAD

La recursividad, consistente en el uso de funciones recursivas, es una herramienta de programación potente que en muchos casos puede producir algoritmos más pequeños.

Una función recursiva es una función que expresa la solución en términos de una llamada a sí mismo. De modo que la llamada a sí mismo se conoce como llamada recursiva.

4.1. ESTRUCTURA

La recursividad se compone de 2 elementos:

1. **Caso base**: es una instancia que se puede resolver sin recursión. Se debe tener siempre al menos un caso base para salir de la recursividad.
2. **Llamada recursiva**: La llamada recursiva debe progresar hacia un caso base.

Ejemplo de función recursiva en Java:

```
// Función recursiva que calcula el factorial de un número
public static int factorialRec(int n){
     if (n==0)
          return 1;                    //Caso base
     else
          return (n*factorialRec(n-1)); //Llamada recursiva
}
```

4.2. CARACTERÍSTICAS

Algunas de las principales características de la recursividad son:

- La recursividad requiere hacer una asignación dinámica de memoria.
- Los datos de las llamadas recursivas se van almacenando en una pila, de forma que se irán retirando de ella en el orden contrario a como se introdujeron.
- Se define la profundidad de recursión, como el número de veces que la función se llama a sí misma antes de iniciar los retornos.
- Es una herramienta muy potente, sobre todo en cálculo, y es idónea para problemas que se definen de forma natural en términos recursivos.
- La solución recursiva suele consumir más recursos que la solución iterativa.

4.3. TIPOS

Existen dos tipos de recursividad:

A. **Recursividad directa**: en este tipo de recursividad una función se invoca a sí misma. La función `factorialRec` vista anteriormente es un ejemplo de recursividad directa.

B. **Recursividad indirecta**: en este tipo de recursividad una función contiene llamadas a otras funciones y así sucesivamente hasta que, finalmente se acaba invocando a su vez a la primera función.

Ejemplo: Mediante estas dos funciones se indica si un número es par o impar.

```
public static boolean esPar(int n){
      if(n==0)
          return true;
      return esImpar(n-1);
}
```

```
public static boolean esImpar(int n){

    if(n==0)

         return false;

    return esPar(n-1);

}
```

5. LIBRERIAS

Las librerías o bibliotecas de funciones son un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que se utilizan por el programador para realizar determinadas operaciones y facilitar así la programación.

Se caracterizan porque no se pueden ejecutar de modo independiente o aislado, sino que la ejecución de su código se realiza previa llamada o inclusión desde un programa.

5.1. ESTRUCTURA

De forma general, la estructura de una librería es la siguiente:

- **Llamada a otras librerías**: Se incluye la llamada a otras librerías que se utilicen dentro de ésta.
- **Declaraciones globales**: se incluye la declaración de variables, constantes y funciones.
- **Definición de funciones**: se incluye el desarrollo de las funciones que contiene la librería.

5.2. CARACTERÍSTICAS

- Las librerías implementan uno de los principios de la programación: la ocultación de la información.
- No tienen un punto concreto de inicio, sino que se puede llamar a cualquiera de sus funciones independientemente del lugar en el que se haya definido dentro de la librería.
- Facilita la reutilización del código.
- Facilitan la portabilidad del software cuando diferentes sistemas utilizan un mismo estándar de codificación.

5.3. TIPOS

Se pueden distinguir dos tipos de librerías:

A. **Librerías estándar**: son las que proporciona el propio lenguaje de programación. Por ejemplo, en Java cuando se descarga el entorno de compilación y ejecución, se obtiene la API de Java. Se trata de un conjunto de bibliotecas que proporcionan clases útiles para nuestros programas. Algunas de las librerías más importantes que ofrece Java son: java.io java.lang, java.util, java.swing, etc.

B. **Librerías de usuario**: Los usuarios también pueden crear sus propias librerías con las funciones que ellos mismos han implementado.

6. RECURSOS Y HERRAMIENTAS DE INTERÉS

Como herramienta educativa de interés relacionada con este tema podemos citar a **Sololearn** que se trata de una aplicación web para iOS y Android. Esta aplicación cuenta con una amplia comunidad de usuarios para el aprendizaje de lenguajes de programación y fundamentos de algoritmia, desde nivel principiante hasta nivel profesional. Fue galardonada por la FbStart (Facebook) como mejor aplicación del año 2017.

- Sitio web oficial: www.sololearn.com

7. CONCLUSIÓN

En el presente tema se ha presentado una visión global de la programación modular, la cual hace los programas más fáciles de escribir, leer y mantener, utilizando para ello el diseño de funciones, consiguiendo además minimizar el tamaño y complejidad de los programas mediante el uso de la recursividad y las librerías.

8. BIBLIOGRAFÍA

- Joyanes Aguilar, L. **Fundamentos de programación. Algoritmos, estructuras de datos y objetos**. Editorial McGraw-Hill
- Alcalde, E. y García, M. **Metodología de la Programación**. Editorial McGraw-Hill
- Prieto, A. Lloris, A. y Torres, J.C. **Introducción a la informática**. Editorial McGraw-Hill
- http://atc.ugr.es/APrieto videoclases Departamento de Arquitectura y Tecnología de los Computadores. Universidad de Granada. Prieto, A.