CURSO DE PROGRAMACIÓN FULL STACK

SUBPROGRAMAS CON PSEINT

PROCEDIMIENTOS

EGG

GUÍA DE SUBPROGRAMAS

En la guía anterior sobre los subprogramas hicimos hincapié en los distintos tipos de subprogramas, pero solo vimos funciones. Ahora vamos a ver otro tipo de subprograma llamado procedimiento.

PROCEDIMIENTOS

Aunque las funciones son herramientas de programación muy útiles para la resolución de problemas, con frecuencia se requieren subprogramas que no retornen ninguna información o se encarguen de imprimir información. En estas situaciones la función no es apropiada y se necesita disponer del otro tipo de subprograma: el procedimiento.
Un procedimiento es un subprograma que ejecuta un proceso específico. Ningún valor está asociado con el nombre del procedimiento; por consiguiente, no puede ocurrir en una expresión. Un procedimiento se llama escribiendo su nombre. Cuando se invoca el procedimiento, los pasos que lo definen se ejecutan y a continuación se devuelve el control al programa que le llamó.

SINTAXIS

```
SubProceso Nombre (parámetros)

      <acciones>

FinSubProceso
```

Los parámetros tienen el mismo significado que en las funciones.

Invocación a un Subprograma

Un procedimiento puede ser llamado de la siguiente forma:

```
nombre(argumentos)
```

- nombre_procedimiento*: procedimiento que se va a llamar.*
- argumentos*: constantes, variables, expresiones.*

Ámbito: Variables Locales y Globales

Las variables utilizadas en los programas principales y subprogramas se clasifican en dos tipos: *variables locales y variables globales*.

Una ***variable local*** es aquella que está declarada y definida dentro de un subprograma, en el sentido de que está dentro de ese subprograma, es distinta de las variables con el mismo nombre declaradas en cualquier parte del programa principal y son variables a las que el algoritmo principal no puede acceder de manera directa.
El significado de una variable se confina al procedimiento en el que está declarada. Cuando otro subprograma utiliza el mismo nombre se refiere a una posición diferente en memoria. Se dice que tales variables son locales al subprograma en el que están declaradas.

Una *variable global* es aquella que está declarada en el programa o algoritmo principal, del que dependen todos los subprogramas y a las que pueden acceder los subprogramas, a través del paso de argumento. La parte del programa/algoritmo en que una variable se define se conoce como ámbito o alcance (scope, en inglés).

El uso de variables locales tiene muchas ventajas. En particular, hace a los subprogramas independientes, siendo solo la comunicación entre el programa principal y los subprogramas a través de la lista de parámetros.

Una variable local a un subprograma no tiene ningún significado en otros subprogramas. Si un subprograma asigna un valor a una de sus variables locales, este valor no es accesible a otros programas, es decir, no pueden utilizar este valor. A veces, también es necesario que una variable tenga el mismo nombre en diferentes subprogramas. Por el contrario, las variables globales tienen la ventaja de compartir información de diferentes subprogramas sin la necesidad de ser pasados como argumento

Comunicación con Subprogramas: Paso de Argumentos

Cuando un programa llama a un subprograma, la información se comunica a través de la lista de parámetros y se establece una correspondencia automática entre los parámetros y los argumentos. Los parámetros son "sustituidos" o "utilizados" en lugar de los argumentos.

La declaración del subprograma se hace con:

```
Subproceso nombre (PA1, PA2, ...,PAn)
      <acciones>
FinSubproceso
```

y la llamada al subprograma con:

```
nombre (AR1, ARG2,…, ARGn)
```

donde PA1, PA2, …, PAn son los parámetros y ARG1, ARG2, …, ARGn son los argumentos.

Cuando nosotros decidimos los parámetros que va a necesitar nuestro subprograma, también podemos decidir cual va a ser el comportamiento de los argumentos en nuestro subprograma cuando lo invoquemos y se los pasemos por paréntesis. Esto va a afectar directamente a los argumentos y no al resultado final del subprograma.
Para esto existen dos tipos más empleados para realizar el paso de argumentos, el paso por valor y el paso por referencia.

Paso por Valor

Los argumentos se tratan como variables locales y los valores de dichos argumentos se proporcionan copiando los valores de los argumentos originales. Los parámetros (locales a la función o procedimiento) reciben como valores iniciales una copia de los valores de los argumentos y con ello se ejecutan las acciones descritas en el subprograma.
Aunque el paso por valor es sencillo, tiene una limitación acusada: no existe ninguna otra conexión con los parámetros, y entonces los cambios que se produzcan dentro del subprograma no producen cambios en los argumentos originales y, por consiguiente, no se pueden poner argumentos como valores de retorno. El argumento actual no puede modificarse por el subprograma.

En PSeInt todas las variables que pasemos como argumentos pasan por defecto "Por Valor" sino se especifica lo contrario explícitamente.

Paso por Referencia

En numerosas ocasiones se requiere que ciertos argumentos sirvan como argumentos de salida, es decir, se devuelvan los resultados al programa que llama. Este método se denomina paso por referencia o también de llamada por dirección o variable. El programa que llama pasa al subprograma la dirección del argumento actual (que está en el programa que llama). Una referencia al correspondiente argumento se trata como una referencia a la posición de memoria, cuya dirección se ha pasado. Entonces una variable pasada como argumento real es compartida, es decir, se puede modificar directamente por el subprograma.

La característica de este método se debe a su simplicidad y su analogía directa con la idea de que las variables tienen una posición de memoria asignada desde la cual se pueden obtener o actualizar sus valores. El área de almacenamiento (direcciones de memoria) se utiliza para pasar información de entrada y/o salida; en ambas direcciones.
En este método los argumentos son de entrada/salida y los argumentos se denominan argumentos variables.