

Tema 7

Funciones Y Procedimientos

Concepto de subprograma

Un subprograma es una parte de un programa, que se desarrolla por separado y se utiliza invocándolo mediante un nombre simbólico.

Funciones

Cuando se diseña y desarrolla un programa aparecen con frecuencia operaciones significativas que dan como resultado un valor simple y único en función de ciertos parámetros.

- **Definición de funciones:** El primer paso en el manejo de una función es declarar su interfaz. Esta declaración incluye su nombre, los argumentos que necesita con el correspondiente tipo para cada uno de ellos, y el tipo de resultado que proporciona.

```
PROCEDURE Nombre( argumento:tipo;
... ) : TipoResultado
```

Se pueden declarar conjuntamente varios argumentos del mismo tipo escribiendo sus nombres, seguidos, separados por comas (,). El valor de retorno se devuelve con RETURN:

```
RETURN expresión;
```

- **Uso de funciones:** Para usar una función en los cálculos de un programa, se invoca está escribiendo su nombre y a continuación, entre paréntesis, los valores concretos de los argumentos, separados por comas.
- **Funciones predefinidas:**

Función	Uso
ABS(X)	Valor absoluto de un número
CAP(C)	Carácter convertido a mayúscula
CHR(X)	Carácter de la tabla de caracteres en la posición X
FLOAT(X)	X convertido a valor REAL
HIGH(V)	Índice superior de un vector abierto V
MAX(T)	Valor máximo del tipo T
MIN(T)	Valor mínimo del tipo T
ODD(X)	Devuelve <i>cierto</i> cuando el valor de X es impar
ORD(X)	Posición que ocupa X en la lista de valores de su tipo
SIZE(T)	Tamaño en bytes del tipo de dato T
TRUNC(R)	Valor de R (REAL) truncado a entero
VAL(T, X)	X convertido al tipo T

- **Funciones estándar:** Las funciones definidas en módulos estándar se denominan funciones estándar y pueden ser utilizadas sin necesidad de escribir su definición, pero (a diferencia de las funciones predefinidas) hay que indicar expresamente que se van a utilizar dichas funciones mediante una declaración `IMPORT` del módulo que las contenga.

Procedimientos

Un procedimiento es un subprograma que realiza una determinada acción. A diferencia de las funciones, un procedimiento no tiene como objetivo, en general, devolver un valor obtenido por cálculo.

- **Definición de procedimientos:** La definición de un procedimiento es la siguiente:

```
PROCEDURE Nombre(argumento:tipo;
...);
declaraciones;
BEGIN
sentencias;
END Nombre;
```

- **Uso de procedimientos:** Un procedimiento se invoca escribiendo su nombre y a continuación, si los hay, los valores de los argumentos particulares en esa llamada, separados por comas. Los valores de los argumentos pueden darse, en general, mediante expresiones. Si no hay argumentos se suprimen también los paréntesis, con lo que la llamada a un procedimiento sin argumento se reduce a su nombre.

- **Procedimientos predefinidos:**

Procedimiento	Uso
DEC(X)	Decrementa en 1 el valor de la variable X
DEC(X, N)	Decrementa en N el valor de la variable X
EXCL(S, X)	Excluye el elemento X del conjunto S
HALT	Finaliza la ejecución del programa
INC(X)	Incrementa en 1 el valor de la variable X
INC(X, N)	Incrementa en N el valor de la variable X
INCL(S, X)	Incluye el elemento X en el conjunto S

- **Procedimientos estándar:** En los módulos estándar asociados a cada compilador se disponen de diversos procedimientos estándar que pueden utilizarse sin más que realizar la correspondiente importación.

Paso de argumentos

La manera fundamental de comunicar información entre las sentencias de un subprograma y el programa que lo utiliza es mediante los argumentos. Los argumentos representan valores que se transmiten desde el programa que llama hacia el subprograma.

- **Paso de argumentos por valor:** El modo de paso por valor implica que los elementos usados como argumentos en la llamada al subprograma no pueden ser modificados por la ejecución de las sentencias del subprograma.
- **Paso de argumentos por referencia:** En ciertos casos es deseable que el subprograma pueda modificar las variables que se usen como argumentos. Esto permite producir simultáneamente varios resultados y no sólo uno. Se consigue anteponiendo la palabra clave `VAR` al nombre del argumento formal.

Visibilidad. Estructura de bloques

Los argumentos no son la única forma de comunicación entre las sentencias de un subprograma que lo invoca. Existe también la posibilidad de que desde dentro del subprograma se puedan ver directamente ciertos elementos del programa que lo usa.

Cualquier bloque tiene visibilidad hacia los bloques exteriores, pero nunca hacia los interiores a él mismo.

Problemas de uso

- **Redefinición de elementos:** De acuerdo con las reglas de visibilidad, el bloque más interno tiene acceso a todos los elementos definidos de los bloques más externos. Al dar un nombre ya utilizado a un nuevo elemento en el bloque interno, automáticamente se pierde la posibilidad de acceso al elemento del mismo nombre existente en cualquiera de los bloques más externos.
- **Efectos secundarios:** Cuando un subprograma modifica alguna variable externa, se dice que está produciendo efectos secundarios o laterales. Se entiende por *funcionalidad* la cualidad que poseen aquellos subprogramas que permiten asegurar que siempre que se llamen con los mismos argumentos producirán exactamente los mismos resultados. Esto se logra evitando la utilización de efectos secundarios. Esta cualidad también se denomina *transparencia referencial*.
- **Doble referencia:** Se produce doble referencia cuando un mismo elemento se referencia con dos nombres distintos. Esto puede ocurrir sí:
 1. Cuando un subprograma utiliza una variable externa que también se le pasa como argumento.
 2. Cuando para utilizar un subprograma se pasa la misma variable en dos o más argumentos.