SUBPROGRAMAS

Consiste en:

La división de Problemas en Subproblemas

Donde cada división:

Es un módulo o un subprograma

El cual realiza:

Una tarea en particular, bajo la dependencia de un algoritmo u otro subprograma que lo utiliza

Lo que implica:

Recibo y entrega de información entre el subprograma y quién lo invoca

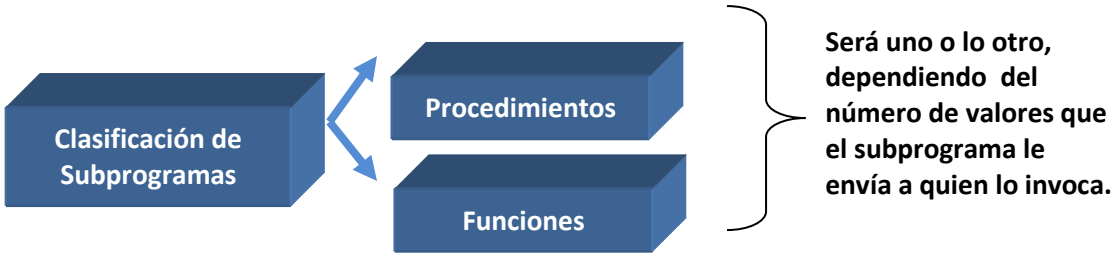
Teniendo en cuenta:

El control de ejecución

La comunicación a través de variables que se denominan parámetros

VENTAJAS DE LOS SUBPROGRAMAS:

- Evita la redundancia de instrucciones.
- Facilidad de distribución del trabajo en equipo.
- Claridad en el código y en la detección de errores en el algoritmo.



PROCEDIMIENTOS

Son subprogramas que *devuelven cero o más valores* a quién lo invoca y retorna el control de la ejecución a la instrucción siguiente desde donde se llaman.

ESTRUCTURA:

Para activar el procedimiento (llamarlo), el formato dentro de quién lo invoque (puede ser el programa principal u otro subprograma), debe ser el siguiente:

```
Inicio
//…
        Nombre_Procedimiento (argumentos)
//…
Fin
```

Los argumentos son la lista de campos (variables o constantes) que usa quién invoca al subprograma. Estos deben ser igual en número y en tipo de dato a los parámetros en el procedimiento como tal, pero con diferente nombre (uso de información de manera local).

El subprograma como tal, es el siguiente:

```
Procedimiento  Nombre_Procedimiento (tipo de dato: parámetros)

        //secuencia de instrucciones

Fin_Procedimiento
```

Nombre_Procedimiento:
Debe ser único y cumplir con las normas de los nombres de las variables.

Parámetros:
Lista de campos separados por coma.

FUNCIONES

Son subprogramas que *devuelve un único valor* a quién lo invoca, a través de una variable en una instrucción de retorno.

ESTRUCTURA:

Para activar la función, el formato dentro de quién lo invoque debe ser el siguiente:

```
Inicio
//…
        variable ← Nombre_Funcion (argumentos de entrega)
//…
Fin
```

La variable debe estar declarada. Dicha activación no tiene argumentos de recibo, debido a que el valor que devuelve la función se le asigna a una variable, que debe ser del mismo tipo de dato que el valor que retorna la función.

El subprograma como tal, es el siguiente:

```
Funcion  Nombre_Funcion  (tipo de dato: parámetros de recibo)

        //secuencia de instrucciones
        Retorne valor

Fin_Funcion
```

Este es el único valor que entrega a quien lo invoca y lo hace en esta instrucción.

Parámetros:
Lista de campos variables separados por coma. En las funciones solo son de recibo, a diferencia de los procedimientos, en los que hay parámetros de recibo y entrega.

EJEMPLO 1:

Hacer un algoritmo que para calcular y mostrar si un número dado es par o no, invoque a un subprograma que se llama Par.

Análisis:

Datos de Entrada: numero
Datos de Salida: Un mensaje que indique si el número es o no par.

Proceso: Normalmente, se haría lo siguiente:

```
Si (num % 2 = 0) ent
        Escriba: "El número es par"
sino
        Escriba: "El número es impar"
Fsi
```

Sin embargo, adicional a lo anterior, se debe tener en cuenta que, al incluir subprogramas, esto debe estar en un procedimiento, dado que no se devuelve ningún valor al programa principal, sino que se muestra un mensaje, como se hará en el algoritmo.

Algoritmo:

```
Inicio
        Entero: numero

        Escriba: "Digite un número para indicarle si es o no par"
        Lea: numero

        Par (numero)   //La flecha hacia arriba, indica que un argumento de entrega
Fin

Procedimiento  Par (entero: num) //La flecha hacia abajo, indica que un parámetro de recibo
        Si (num % 2 = 0) ent
                Escriba: "El número es par"
        sino
                Escriba: "El número es impar"
        Fsi
Fin_Proc
```

Prueba de Escritorio:

```
Inicio
        numero

        Digite un número para indicarle si es o no par
        numero←7

        Par (7)

        Procedimiento  Par (num←7)
```

```
        Si (7 % 2 = 0) ent
        sino
                El número es impar
        Fsi

    Fin_Procedimiento
Fin
```

EJEMPLO 2:

Hacer un algoritmo con subprogramas, que encuentre el mayor valor en un conjunto de tres números reales diferentes.

Análisis:

Datos de Entrada: num1, num2, num3
Datos de Salida: mayor
Proceso: //Se debe utilizar función, dado que se devuelve un valor, el mayor

```
Funcion  Mayor (real: num1, num2, num3)
        //Real: mayor
        Si (num1>num2  ^ num1>num3) ent
                mayor ← num1
        sino
                Si (num2>num3) ent
                        mayor ← num2
                sino
                        mayor ← num3
                Fsi
        Fsi
Retorne mayor
Fin_Funcion
```

Algoritmo:

```
Inicio
        Real: numero1, numero2, numero3, numeromayor

        Escriba: "Digite tres números diferentes"
        Lea: numero1, numero2, numero3
```

```
        Si ( numero1<>numero2 ^ numero2<>numero3 ^ numero1<>numero3) ent

                numeromayor ← Busqueda_Mayor (numero1, numero2, numero3)
                Escriba: "El número mayor es:", numeromayor

        sino
                Escriba: "Hay números iguales"
        Fsi
Fin

Funcion  Busqueda_Mayor (Real: num1, num2, num3)
        Real: mayor

        Si (num1>num2  ^ num1>num3) ent
                mayor ← num1
        sino
                Si (num2>num3) ent
                        mayor ← num2
                sino
                        mayor ← num3
                Fsi
        Fsi

Retorne mayor
Fin_Funcion
```

Prueba de Escritorio:

```
Inicio
        numero1
        numero2
        numero3
        numeromayor

        Digite tres números diferentes
        numero1←8
        numero2←5
        numero3←1
```

```
Si ( 8<>5 ^ 5<>1 ^ 8<>1) ent
        numeromayor ← Busqueda_Mayor (8, 5, 1)

        Funcion Busqueda_Mayor (num1<-8, num2<-5, num3<-1)
                mayor

                Si (8>5  ^ 8>1) ent
                        mayor ← 8
                Fsi
        Retorne 8
        Fin_Funcion

        numeromayor ← 8
        El número mayor es 8
Fsi
Fin
```