

República Bolivariana de Venezuela  
Ministerio de Educación Superior  
Instituto Universitario de Tecnología Valencia  
Departamento de Informática



### GUIA TEORICA N° 3 LENGUAJE DE PROGRAMACION I APUNTES: SUBPROGRAMAS.

Profesor : DELY M. GIL A.

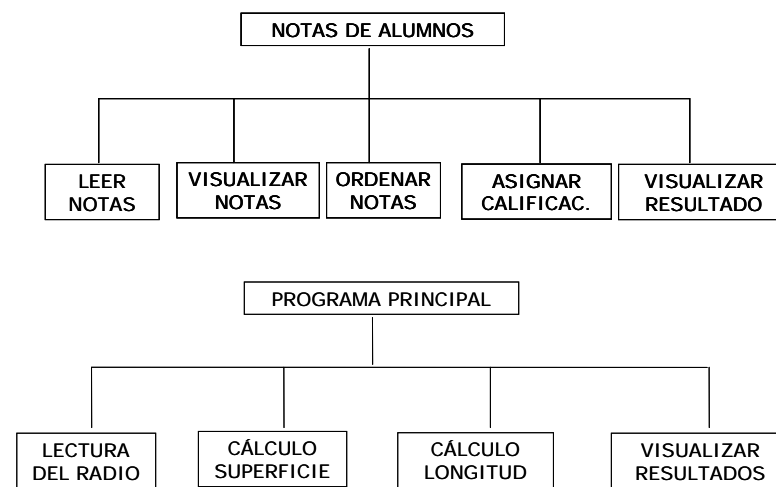
VALENCIA, 2004

#### SUBPROGRAMAS

- La programación modular consiste en dividir un programa en subprogramas llamados subrutinas o módulos, evitando la creación de enormes programas que resulten imposible de manejar.
- Cuando un grupo de pasos se repite en varios lugares del programa, es útil convertir ese conjunto de instrucciones en un subprograma, para evitar una inútil repetición de pasos. En turbo pascal estos subprogramas son denominados :
  - Procedimientos y
  - Funciones.

Ing Del y Gil

#### DESCOMPOSICIÓN MODULAR



Ing Del y Gil

## SUBPROGRAMAS

- La división de un programa extenso en varios subprogramas agiliza el trabajo de programación y facilita la búsqueda de errores al momento de depurar el programa. Los subprogramas declarados en un programa pueden ser llamados desde cualquier parte de un programa las veces que sea necesario.
- Cuando se realiza la llamada a un subprograma, el control del programa se traslada hasta el subprograma, lo ejecuta y regresa a la siguiente instrucción desde el lugar donde fue llamado.

Ing Del y Gil

## DECLARACIÓN Y LLAMADA DE UN SUBPROGRAMA

- La declaración de los subprogramas se hacen antes del cuerpo principal (en la sección de declaraciones), pero las llamadas se hacen en el cuerpo del programa o en otros subprogramas.

<pre> Program ejemplo; Uses     Crt; Var     i,j :byte; {declaración de subprogramas}         </pre>	<pre> Procedure NombrePro; Begin     ..... End; Begin {Cuerpo principal}     ...     NombreProc; End.         </pre>
--	--

Ing Del y Gil

## FLUJO DE CONTROL EN LA LLAMADA DE UN SUBPROGRAMA

<pre> Program &lt;Nombre&gt;; ... Begin     sentencia 1;     sentencia 2;     Llamada Sub Programa     sentencia 4;     sentencia 5; End.         </pre>	<pre> Subprograma&lt;nombre&gt;; ... Begin     sentencia 1;     sentencia 2;     sentencia 3;     sentencia 4; End.         </pre>
--	--

Ing Del y Gil

## PROCEDIMIENTOS

Un procedimiento es un subprograma que realiza una tarea específica. Puede recibir cero o más valores del programa que es llamado y devolver cero o más valores a dicho programa llamador.

```

Program Raices;
...
Begin
    Read(A,B,C)           {lectura de datos}

    Ecu(A,B,C,X1,X2); {llamada al
                        Procedimiento}
    Write(X1,'y',X2,' son las raices'); {Salida}
    Readkey;
        
```

Ing Del y Gil

### DECLARACIÓN - PROCEDIMIENTO

- ❖ Sintaxis Formato 1  
(sin Parámetros)  
Procedure NombreProc;  
    {declaración de variables locales}  
Begin  
    {Cuerpo del Procedimiento}  
End.
- ❖ Llamada Formato 1  
    NombreProc;

Ing Del y Gil

### DECLARACIÓN – PROCEDIMIENTO

- ❖ NombreProc: identificador válido
- ❖ Lista de parámetros formales o ficticios:  
Sirven para pasar información al procedimiento y/o devolver información del procedimiento al programa que lo invoca.
- ❖ Formato:  
    Lista\_1: Tipo\_1; Lista\_2: Tipo\_2;

Procedure Prueba (a,b:integer; var c,d: real;  
    var e: char);

Llamada →

Ing Del y Gil

### DECLARACIÓN – PROCEDIMIENTO

- ❖ Sintaxis Formato 2  
(con Parámetros)  
Procedure NombreProc (Lista de parámetros  
    formales);  
    {declaración de variables locales}  
Begin  
    {Cuerpo del Procedimiento}  
End;
- ❖ Llamada Formato 2  
    NombreProc(Lista de Parámetros  
actuales);

Ing Del y Gil

### EJEMPLO DE PROCEDURE

- ❖ Supóngase que se desea buscar en un archivo el registro del empleado que se va a modificar y eliminar. Como esta búsqueda se requiere en el procedimiento MODIFICAR y ELIMINAR, se realizará un subprograma común llamado BUSQUEDA. Como BUSQUEDA es llamado en otros procedimientos, él debe estar declarado antes que éstos.
- ❖ Si no se crea la rutina BUSQUEDA, entonces las instrucciones de búsqueda del empleado tendría que ser incluidas en MODIFICAR y ELIMINAR aumentando el número de instrucciones y complejidad del programa.

Ing Del y Gil

```

Program Nomina;
  (*Declaracion*)

Function
Busqueda(..): Integer;
  Begin
    ...
  End;

Procedure Modificar (..);
  Begin
    (*Se hace la
    llamada a
    Busqueda*)
  End;

Procedure Eliminar (..);
  Begin
    (*Se hace la
    llamada
    a Busqueda*)
  End;

  (*****P.P****)
  Begin
    Read(opc)
    Case opc of
      'M' : Modificar(...);
      'E' : Eliminar(...);
    End;...
  End.
  
```

Ing Del y Gil

```

Program Recuadro;

Procedure Estrellas (N: byte); { N asteriscos}
Var
  i: byte;
  Begin
    For i:= 1 to N do
      write('*')
    End;

    (****Programa Principal****)
  Begin
    Estrellas(5);      { llamada al Proced.}
    Write('Mensaje');
    Estrellas(10)
  End.
  
```

Ing Del y Gil

```

Program Recuadro;

Procedure Estrellas; {declaracion del Proced.}
{Visualizar 9 asteriscos}
  Begin
    write('*****')
  End;

  (****Programa Principal****)
  Begin
    Estrellas;      { llamada al Proced.}
    Write('Mensaje');
    Estrellas      { llamada al Proced.}
  End.
  
```

Ing Del y Gil

**EJEMPLO DE PROCEDURE**

```

Procedure Geometria(Longitud, Anchura: real;
  Var Area, Perimetro: real);

  Begin
    Area      := Longitud * Anchura;
    Perimetro := 2 * (Longitud + Anchura)
  End.
  
```

Este procedimiento recibe como información de entrada Longitud y Anchura, y devuelve como salida Area y Perimetro.

Ing Del y Gil

## VARIABLES

### ◆ Locales

Son variables que están declaradas dentro de un subprograma y se dice que es local al subprograma. Una variable local sólo está disponible durante el procedimiento del subprograma. Su valor se pierde una vez que el subprograma se ejecuta.

### ◆ Globales

Son variables que están declaradas en el programa principal.

Ing Del y Gil

Program Variables;

Var { V. Globales → A y B }

A,B :integer;

Procedure Intercambio (var X,Y : integer);

Var

Aux: integer;

{V. Local}

Begin

Aux := X;

X := Y;

Y := Aux

End;

Begin {P.P.}

A := 1;

B := 5;

write('Valores antes: ',A,' ',B)

Intercambio (A,B);

write('Valores después: ',A,  
' ', B)

End.

Ing Del y Gil

## PARÁMETROS

Los parámetros son identificadores que sirven de enlace entre el cuerpo principal y los módulos que integran el programa, permitiendo el traspaso de información entre ambos.

Tipos:

### ◆ Parámetros Actuales

Intercambio (A,B);

### ◆ Parámetros Formales

Valor : Procedure Estrellas  
(N:byte);

Referencia:

Procedure Intercambio  
(var X,Y : integer);

Ing Del y Gil

## PARÁMETROS ACTUALES

Son los que se encuentran en los llamados de los subprogramas en el cuerpo principal del programa.

Estos parámetros son "variables globales" al programa y el uso de ellos es de entregar y/o recibir información.

Program Variables;

Var { V. Globales → A y B }

A,B :integer;

Procedure Intercambio (var X,Y : integer);

Begin

...

End;

Begin {P.P.}

Intercambio (A,B);

End.

Ing Del y Gil

### PARÁMETROS FORMALES O FICTICIOS

Se encuentran en la cabecera de los procedimientos. Ellos sirven para contener los valores de los parámetros actuales cuando se invoca al procedimiento.

```

Program Variables;
Var           { V. Globales → A y B}
  A,B :integer;
Procedure Intercambio (var X,Y : integer);
Begin
  Intercambio (A,B);
End.
  
```

Ing Del y Gi I

### PARÁMETROS FORMALES TIPO REFERENCIA

Cuando se pasan parámetros por Referencia del programa principal a un procedimiento, los cambios que se realicen a una variable introducida como parámetro, modifican el contenido de dicha variable. En este caso específico los parámetros deben estar precedidos por la palabra reservada "VAR", lo cual indica al compilador que los valores enviados como parámetro pueden "CAMBIAR".

```

Procedure Geometria(Longitud, Anchura: real;
  Var Area,Perimetro: real);
  
```

Ing Del y Gi I

### PARÁMETROS FORMALES TIPO VALOR

Cuando se pasan parámetros por valor en memoria sucede lo siguiente: se obtiene una copia temporal de la variable usada y dentro de la función o procedimiento se trabaja con la copia obtenida. De esta forma la variable introducida como parámetro, no será afectada en su valor inicial al terminar el proceso, sin importar las diferentes operaciones que se realicen con dicha copia.

```

Procedure Geometria(Longitud, Anchura: real;
  Var Area,Perimetro: real);
  
```

Ing Del y Gi I

### CORRESPONDENCIA DE PARÁMETROS

Los parámetros actuales en la invocación del procedimiento deben coincidir en número, orden y tipo con los parámetros ficticios de la declaración del procedimiento.

Nota: Los nombres de los parámetros actuales y formales pueden ser los mismos, aunque se recomienda sean diferentes a efectos de legibilidad del programa

Ing Del y Gi I

```

Program, Correspondencia;
Var
    X,Y,A,P      : real;

Procedure Geometria(Longitud,Anchura:real,
                    var Area,Perimetro: real)
{ Procedimiento que calcula el área y perímetro}
Begin
    Area      := Longitud * Anchura;
    Perimetro := 2 * (Longitud + Anchura)
End;
{ Programa Principal}
Begin
    Write('Ingresar Longitud: '); Read(x);
    Write('Ingresar Anchura: ');  Read(y)
    Geometria(x,y,A,P);
    { llamada al procedimiento}
    Writeln('El Área es      : ', A:0:2);
    Write('El Perímetro es : ', P:0:2);
End.

```

```

Program, Correspondencia;
Var
    X,Y,A,P      : real;

Procedure Geometria(Longitud,Anchura:real,
                    var Area,Perimetro:
real);
Begin {P.P}
...
    Geometria(4, 7, A, P);
...
End.

```

Ing Del y Gil

## FUNCIONES DE TURBO PASCAL

### ⊕ Funciones matemáticas (Def. Y Ejemplo)

Abs:

Exp:

Ln:

Sqr:

Sqrt

Frac:

Int:

Trunc:

Round:

### ⊕ Funciones trigonométricas

(Sin, Cos, Arctan)

Sin:

Cos:

Arctan:

### ⊕ Funciones ordinales

(Ord, Chr, Pred, Succ)

Ord:

Chr:

Pred :

Succ:

### ⊕ Funciones especiales.

(Pi, Odd)

### ⊕ Incremento y decremento (Inc,Dec)

### ⊕ Números Aleatorios

(Random,Randomize)

Rahdom:

Randomize:

Diferencia→

Ing Del y Gil

### FUNCIONES DEFINIDAS POR EL USUARIO

☐ Es un subprograma que devuelve un único valor o resultado al programa o subprograma que lo llama.

Las funciones y los procedimientos son definidos de forma parecida, excepto que una función retorna un valor y el tipo de ese resultado es específico.

```
Begin
  Read(A,B)           {lectura de datos}
  z := Suma (A,B);    {llamada a una Función}
  write('La suma es: '; Z:3:2) {Salida}
  Readkey;
End.
```

Ing Del y Gil

### DECLARACIÓN DE UNA FUNCIÓN

☐ Sintaxis Formato 2

(con Parámetros)

```
Function NombreFunc (pf1,pf2,...,pfn): tipo;
  {declaración de variables locales}
Begin
  {Cuerpo de la función}
  Nombrefun := valor de la función;
End;
```

☐ Llamada Formato 2

Variable:= NombreFunc(pa<sub>1</sub>,pa<sub>2</sub>,...,pa<sub>n</sub>);

Ing Del y Gil

### DECLARACIÓN DE UNA FUNCIÓN

☐ Sintaxis Formato 1

(sin Parámetros)

```
Function NombreFunc: Tipo_función;
  {declaración de variables locales}
Begin
  {Cuerpo de la Función}
  nombreFun := valor de la función;
End;
```

☐ Llamada Formato 1

Variable:=NombreFunc

Ing Del y Gil

### DECLARACIÓN DE UNA FUNCIÓN

☐ NombreFunc: identificador válido

☐ Lista de parámetros formales o ficticios:

Lista\_1: Tipo\_1; Lista\_2: Tipo\_2;

☐ tipo\_funcion: Tipo de dato que devuelve la función

Function expon(u:real; e:integer) : real;

Begin

...

P := expon(x,y); { p: V.G. → real}

End.

Ing Del y Gil

```

Program Potencia;
Var
    N, ResCubo : integer;

Function CUBO (X:integer) : integer;
    {Proporciona x al cubo}
begin
    CUBO:= x*x*x
End;

Begin {Programa Principal}
    Write("Ingresar el número: ");
    Read(n)
    ResCubo:= CUBO(n);
    Write(N, ' al cubo es: ',ResCubo)
End.

```

Ing Del y Gil

```

Program Suma_N;
Var
    Num ,Total_suma: word;

Function Suma(N:word): word;
{Calcula la suma 1+2+3+...+n}
Var Total,j:word;
Begin
    Total := 0;
    For j:=0 1 to N do
        Total := Total + j;
    Suma := Total
End;

Begin
    Write('Cantidad de números a sumar: '); Read(Num);
    Total_Suma := Suma(Num);
    Write(' La suma de la serie es: ', Total_Suma);
    Readkey;
End.

```

Ing Del y Gil

## EJEMPLOS DE FUNCIONES

```

Program Ejemplos;
Uses CRT;
Var

(**Función que calcula el triple de un número**)
Function Triple (Numero:real):real
    Begin
        Triple:= 3*Numero;
    End;

(**Función que devuelve el signo del número**)
Function Signo(x:real) :integer;
    { devuelve el signo del parámetro X }
    var aux: integer;
    begin
        if x>0 then aux:= 1;
        if x=0 then aux:= 0;
        if x<0 then aux:= -1;
        signo:= aux
    End;

Function Signo(x:real) :integer;
    { devuelve el signo del parámetro X }
    begin
        if x>0 then signo := 1;
        if x=0 then signo := 0;
        if x<0 then signo := -1
    end;

(**Programa Principal*)

```

Ing Del y Gil

❶ Diseñar una función que permita verificar si un número determinado está comprendido en un rango determinado (Límite superior, Límite inferior), en caso afirmativo devolver un valor verdadero y en caso contrario falso.

```
Program Validar_Rangos;
Var
    N : integer;
    X,y : byte;

Function Rango_Validado(LI,LS,N: integer) :
boolean;
{Función que devuelve true si el rango es válido
de lo contrario devuelve Falso}
begin
    if (N>=LI) and (n<= LS) then
        Rango _Validado:= TRUE
    Else
        Rango_Validado := FALSE;
        {Rango_valido := n in [LI..LS] }
End;

{Programa Principal}

Begin
    Write(' Ingresar un número entre 1 y 10: ');
    x:=wherex;y:=whereY;
    Repeat
        gotoxy(x,y);ClrEol;
        gotoxy(x,y);
        Read(n)
    Until(Rango_Validado(1,10,n))
Readkey;
End.
```

❶ Diseñar una función que convierta una letra en minúscula si está en el rango 'A'..'Z'.

```
Program Funciones;
Uses
    Crt;
Var
    Caracter :Char;

(*Funcion que devuelve el car en minuscula*)
Function Minuscula(Car:Char):Char;
Var
    R_Ord : Byte;
    R_Chr : Char;
Begin
    If car in ['A'..'Z'] then
        Begin
            R_Ord := Ord(Car);
            R_Chr := Chr(R_Ord);
            Minuscula := Chr(Ord(car)+32)
        End
    Else
        Minuscula:= Car;
    (** Otra lógica de programación **)
    Minuscula := Car;
    If car in ['A'..'Z'] then
        Minuscula := Chr(Ord(car)+32)
    (***)
    Fin
    End;

Begin
    Write('Ingrese caracter:');
    readln(Caracter);
    Caracter:= Minuscula(Caracter);
    Writeln('Caracter en minúscula es: ',Caracter);
    readkey
End.
```