UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE MATEMÁTICA
INTRODUCCIÓN A LA COMPUTACIÓN

TEMA 4: Subprogramas

Los subprogramas son rutinas, procedimientos o conjuntos de instrucciones que realizan una labor específica. Los subprogramas o subrutinas nacieron de la necesidad de no repetir innecesariamente un trabajo ya hecho. Pueden invocarse desde el cuerpo del programa principal cuantas veces se desee. Constituyen el núcleo de lo que se denomina programación estructurada, y permiten la descomposición de un problema complejo en subproblemas más sencillos abordables de forma independiente

En principio, por motivos de reusabilidad (usar varias veces) y de claridad se pueden definir subprogramas, también denominados procedimientos o subrutinas. Un subprograma es en sí un programa completo, con su parte de declaraciones y sus instrucciones. Con esta técnica se solucionan problemas complejos al dividirlos en subprogramas y luego dividirlos estos en otros mas simples, hasta que estos sean más fáciles de resolver. Esta técnica se llama "divide y vencerás". El problema principal denominado controlador o conductor (drive) y la solución de los subproblemas conocidos como procedimientos (subrutinas) o funciones.

Los subprogramas tienen la propiedad de poder ser utilizados dentro de otros programas o subprogramas. Para ello:

- el subprograma debe incluirse en la parte de declaraciones del programa o subprograma que lo quiere usar;
- para usar el subprograma se realiza una invocación o llamada al mismo dentro de las instrucciones del programa o subprograma que lo quiere usar.

Procedimientos o Acciones Nominadas

Conjunto de acciones que tiene un nombre que lo identifica. Puede hacerse referencia a este nombre para ejecutar dicho conjunto de acciones. Las acciones facilitan la programación modular. Las acciones se utilizan como mecanismo de abstracción. Una vez ejecutadas todas las acciones, se retorna al punto en el cual se hizo la invocación a la acción nominada.

Notación a utilizar:

Procedure <Nombre>(<Lista de parámetros formales>])

{<Acciones>}

EndProcedure

Invocación de la acción:

<Nombre>([<Lista de parámetros actuales>]);

Ejemplo:

```
Procedure Saludar()
String nombre;
Write("Escribe tu nombre: ");
read(nombre);
Write("Hola " + nombre);
EndProcedure

Procedure Despedida()
Write("Hasta luego");
EndProcedure

Procedure Main()
Saludar();
Despedida();
EndProcedure
```

Ejemplo: Imprimir tablas de multiplicar

```
Integer t,x;

Procedure ImprimirTabla()
Integer i, x;
For i = 1 to 11 do
x = i * t;
Write(x);
EndFor
EndProcedure

Procedure Main()
Read(x);
t = x;
ImprimirTabla();
EndProcedure
```

En este ejemplo, antes de invocar a la acción ImprimirTabla, se copia el valor de x en t. Las variables t, x son globales y pueden ser accesadas desde cualquier punto del programa. Las variables i, x de ImprimirTabla únicamente son visibles desde la acción ImprimirTabla. La única vía que tienen Main e ImprimirTabla de comunicarse es a través de la variable t, ya que la variable x global no es visible dentro de ImprimirTabla, porque se redefine localmente.

Otra forma de lograr que las acciones se comuniquen es a través del pase de parámetros. Este mecanismo permite que una acción invoque a otra y le envíe los valores necesarios para su ejecución. Por ejemplo, un número indicando cual tabla de multiplicar desea imprimirse. De esta forma, el algoritmo anterior se modificaría:

```
Integer x;

Procedure ImprimirTabla(Integer t)
Integer i, x;
For i = 1 to 11 do
x = i * t;
Write(x);
EndFor
EndProcedure

Procedure Main()
Read(x);
ImprimirTabla(x);
EndProcedure
```

Tipos de pase de parámetro

- Por valor: cuando se invoca al procedimiento, el valor del parámetro actual es copiado en el parámetro formal. Cualquier modificación que se haga sobre el parámetro formal se está haciendo sobre una copia, por lo que el valor original se mantiene intacto.

- Por referencia: cuando se invoca al procedimiento, el parámetro formal recibe una referencia del parámetro actual. Esto significa que el parámetro formal contiene la dirección en memoria del parámetro actual. De esta forma, cada cambio que se haga sobre el parámetro formal, en realidad se está haciendo directamente sobre el parámetro actual.

Para indicar que un parámetro formal es por referencia se usará la notación:

```
Procedure xyz(Ref <tipo> <nombre de variable>)
...
EndProcedure
```

Si en la declaración del parámetro no aparece la palabra Ref, entonces el parámetro es pasado por valor.

Por ejemplo:

```
Integer x;

Procedure A(Ref Integer y)
y = y + 1;
EndProcedure

Procedure B(Integer y)
y = y + 1;
EndProcedure

Procedure Main()
x = 0;
B(x);
Write(x);
A(x);
Write(x);
EndProcedure
```

Otro ejemplo:

```
Integer w,z;

Procedure A(Integer x, Ref Integer y)
x = x + 1;
y = y + 2;
w = w + 3;
Write(x); Write(y); Write(w); Write(z);
B(x,y);
Write(x, y, w, z);
EndProcedure

Procedure B(Ref Integer r, Ref Integer s)
Integer z;
r = r + 4;
s = s + 5;
z = w + z;
Write(r); Write(s); Write(w); Write(z);
EndProcedure
```

```
Procedure Main
Integer u,p;
w = 1;
z = 1;
u = 0;
p = 2;
Write(w); Write(u); Write(p); Write(z);
B(p,u);
Write(w); Write(z); Write(p); Write(u);
A(u,p);
Write(w); Write(z); Write(p); Write(u);
EndProcedure
```

Los parámetros por referencia se utilizan cuando se necesita devolver algún valor del procedimiento llamado al procedimiento llamador. Por ejemplo, un procedimiento que dados dos números encuentre el mayor entre ellos:

```
Procedure Max(Integer A, Integer B, Ref Integer C)
if A >B then
C = A;
else
C = B;
EndProcedure

Procedure Main()
Integer R, A, B;
Read(A);
Read(B);
Max(A, B, R);
Write(R);
EndProcedure
```

Cuando el parámetro formal es por referencia, el parámetro actual debe ser una variable, no puede ser ni una expresión, ni una constante, ya que estos no tienen dirección de memoria.

Funciones

Una función es un tipo procedimiento que siempre devuelve un valor de algún tipo. Las funciones toman uno o más valores llamados argumentos y producen un valor llamado resultado. Para declarar funciones se utilizará la notación:

Function <Nombre>([<Lista de par_ametros>]):<Tipo>
...
return <valor>;
EndFunction

<Tipo> indica el tipo de datos de retorna la función.

La línea return <valor> indica cuál es el valor que debe retornarse. Cuando se llega a esta línea, la invocación de la función concluye, y el control de programa se retorna al procedimiento llamador.

Ejemplo:
Function Max(Integer A, Integer B) : Integer
if A >B then
return A;
else
return B;
EndProcedure
Procedure Main()
Integer A, B;
Read(A);
Read(B);
Write(Max(A, B));
EndProcedure

Otra forma de implementar Max es:

Function Max(Integer A, Integer B) : Integer
if A > B then
return A;
endIf
return B;
EndProcedure

Licenciado Wilson Manyoma, Caracas, Diciembre 2012