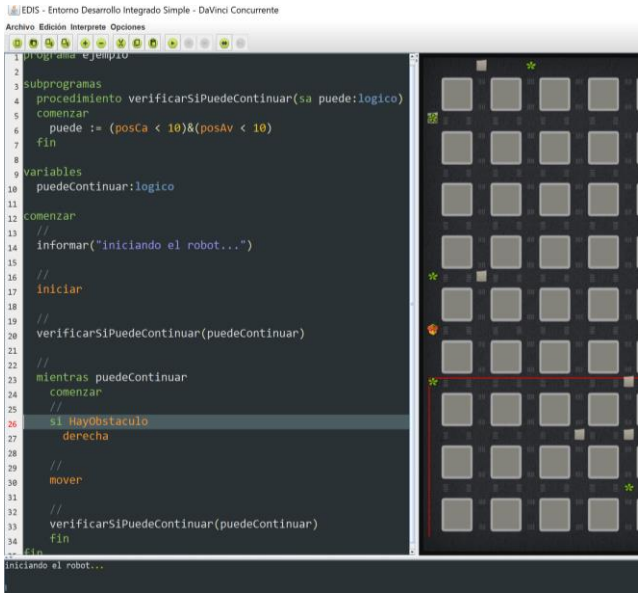
Funciones y Procedimientos

Materia: Algorítmica y programación I

Unidad: Programación Modular

Docente: Daniel Eugenio, Aguil Mallea.



Contexto

- Expresión de problemas y algoritmos
  - Etapas en la resolución de problemas
    - Análisis
    - **Diseño de la solución**
    - Especificación del algoritmo
    - Escritura del programa
    - Verificación
  - Subprogramas
    - Procedimiento
    - Parámetros (en – sa – es)
    - Alcance

Introducción – Diseño

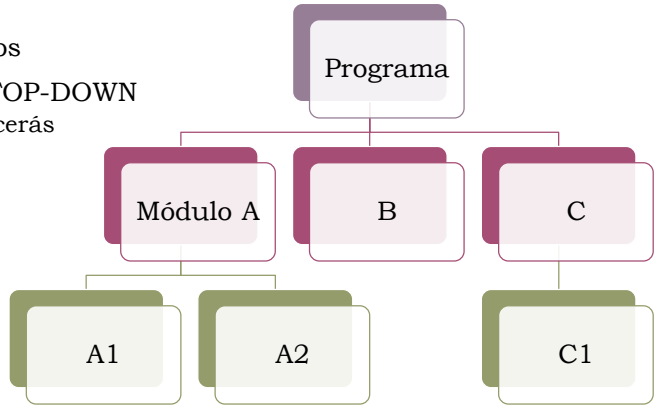
- Diseño de una solución
  - Definir una **estructura de sistema de software** que resuelva el problema.
  - Definir una estructura de sistema hardware.

- **Sistema de software**
  - División del problema (modularización del programa)
    - Descomponer el problema en partes más simples
    - Establecer la comunicación entre las partes

Modularización

- Descomponer la solución en módulos
- **Técnica de diseño descendente** TOP-DOWN
  - Basada en el paradigma divide y vencerás
  - Tareas específicas.

- **La combinación de todas las tareas >> solución**



Modularización

- Realizar un diseño, utilizando el método, nos facilitará:
  - Resolver **problemas complejos** (resolvemos problemas más simples)
  - La **lectura** del sistema en general
  - El mantenimiento
  - Las **modificaciones**
    - La detección y depuración de **errores**
    - La **reutilización** de tareas
  - Trabajar en forma **colaborativa**

- **Establecer adecuadamente la comunicación** entre los módulos no llevará a la resolución del problema.

Módulos o SubProgramas

- Son piezas, fragmentos o **trozos de código** (instrucciones + datos) que realizan una tarea específica.

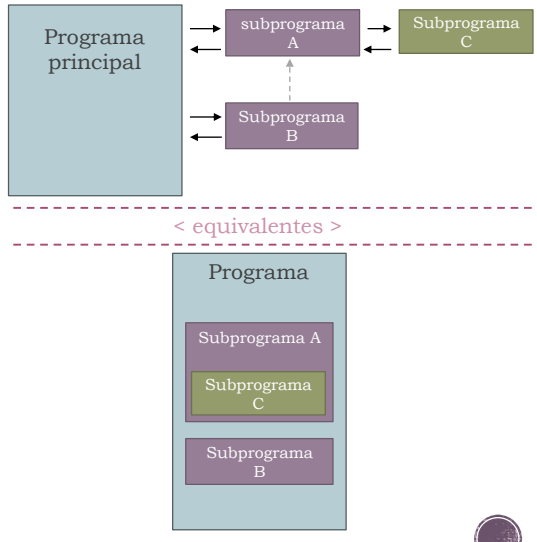
- **Identificado con un nombre y parámetros**, permiten su llamada y la correspondiente comunicación (con el programa o subprograma que los invoca)

- **Independientes**, no pueden tener acceso a otras partes de módulos. Solo pueden realizar invocaciones a otros módulos .

- El **alcance** o visibilidad de un módulo es dependiente del lenguaje

Subprograma

- Lo activa un programa u otro subprograma
- En pascal el alcance de cada uno permite:
  - PP → A
  - PP → B
  - PP X C
  - A → C
  - B → A
  - A X B
  - B X C
- Al programa principal lo podemos considerar como un módulo



Tipos de subprograma

- **Funciones**
  - Retorna **siempre 1 valor** (tipo simple)
  - Tiene asociado un **tipo de dato**
  - Puede formar parte de una expresión
  - Por lo general se calcula el valor de retorno en base a su entrada
- **Procedimientos** o subrutinas
  - Puede retornar **0 o N valores** (pueden ser compuestos)
  - No pueden formar parte de una expresión.
  - Por lo general ejecutan un conjunto de sentencias
- Ambos tipos de subalgoritmos realizan tareas específicas, se identifican con un nombre y aceptan parámetros.

Función – un ejemplo

```
function cantidad(<parámetros formales>) :<tipo_de_dato>;
var …
begin
        …
        cantidad:= <valor>;
end;
var resultado:<tipo_de_dato>;
begin
    resultado := cantidad(<parámetros reales o argumentos>) + ….;
end.
```

forman parte de una expresión

Procedimiento – un ejemplo

```
procedure imprimir(<parámetros formales>);
var …
begin
        …
end;
var resultado:<tipo_de_dato>;
begin
    resultado := cantidad(<argumentos>) + ….;
    imprimir(<argumentos>);
end.
```

NO PUEDEN formar parte de una expresión;
No tienen asociado un tipo de dato

Parámetros

- La serie de datos con la que se **comunican los subprogramas**
  - **Parámetros formales:** son los que aparecen en la declaración del subprograma.
    - Modo de comunicación
    - Nombre
    - Tipo de dato
  - **Parámetros reales** o argumentos: son las expresiones o variables que aparecen en la invocación del subprograma.
    - Dependiendo el modo de comunicación se pueden usar expresiones o no.
  - Pascal solo permite pasaje de parámetros por posición.
    - El tipo y la cantidad se deben corresponder en orden.

Parámetros – modo de comunicación

- El modo de comunicación establece **cómo se realiza el enlace** (transmitidos o devueltos) entre los argumentos y los parámetros formales.
- **Por valor**
  - Se realiza **una copia del argumento al parámetro.** Luego puede utilizarse como una variable.
  - No se altera el valor del argumento.

```
procedure imprimir(texto:String);
begin
    writeLn(texto);
    texto := 'hola mundo';
end;
```

```
var
    mensaje:string;
begin
    mensaje:= 'el universo del discurso…';
    imprimir(mensaje);
    imprimir(mensaje);
end.
```

el universo del discurso…
el universo del discurso…

Parámetros – modo de comunicación

- **Por referencia (variable)**
  - Al parámetro formal se le **asigna la misma dirección de memoria** que tiene el argumento. Luego puede utilizarse como una variable.
  - Se altera el valor del argumento.

```
procedure imprimir(var texto:String);
begin
    writeLn(texto);
    texto := 'hola mundo';
end;
```

```
var
    mensaje:string;
begin
    mensaje:= 'el universo del discurso…';
    imprimir(mensaje);
    imprimir(mensaje);
end.
```

el universo del discurso…
hola mundo

Parámetros – modo de comunicación

- **Por referencia (constante)**
  - Al parámetro formal se le **asigna la misma dirección de memoria** que tiene el argumento.
  - El compilador verifica que sea tratada como solo lectura.

```
procedure imprimir(const texto:String);
begin
    writeLn(texto);
    texto := 'hola mundo';
end;
```

Se produce un error en tiempo de compilación

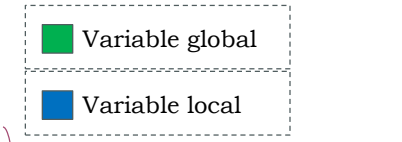
```
var
    mensaje:string;
begin
    mensaje:= 'el universo del discurso…';
    imprimir(mensaje);
    imprimir(mensaje);
end.
```

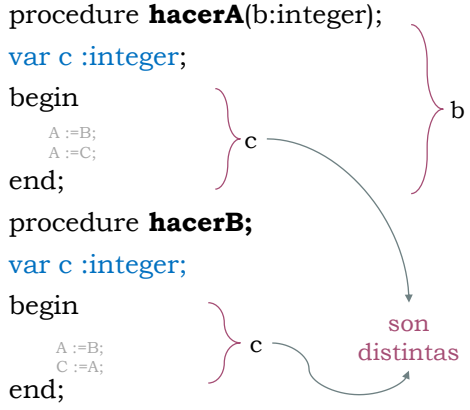
Variables locales y globales

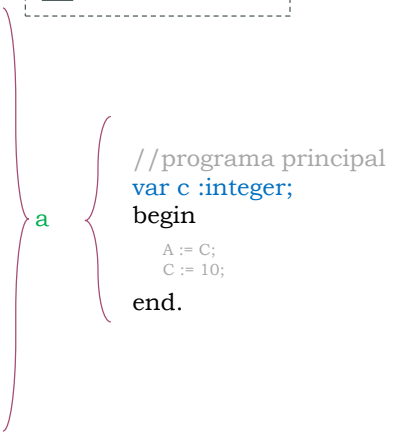
- Una **variable local** es aquella que se encuentra **definida dentro de un módulo** (programa o subprograma).
  - Sólo podrá ser accedida por el módulo.
  - Se puede utilizar el mismo nombre siempre y cuando estén definidas en distintos espacios.
- Una **variable global** es aquella que se encuentra **definida en un programa** y puede ser accedida desde cualquier espacio.
  - Si bien las variables globales son útiles en ciertos escenarios, en esta etapa deberíamos minimizar su uso, ya que pueden producir efectos no deseados!!!

Alcance

```
program holaMundo;
var a :integer;
procedure hacerA(b:integer);
var c :integer;
begin
    A :=B;
    A :=C;
end;
procedure hacerB;
var c :integer;
begin
    A :=B;
    C :=A;
end;
//programa principal
var c :integer;
begin
    A := C;
    C := 10;
end.
```







Algunos ejercicios…

- Escribir una programa, que lea un valor (natural) y luego calcule el factorial del valor leído.
- Escribir una función MOD (X,Y) que calcule el resto de la división de X por Y (X,Y son enteros, al igual que el resto).
- Escribir un procedimiento para intercambiar los valores de dos variables X e Y Enteras

Resumen

- Es necesario la **utilización de métodos** y técnicas como la expuesta, que nos **permitan diseñar soluciones** a problemas complejos.
- **Analizar el problema y dividirlo en subproblemas** sucesivamente hasta contar con tareas simples.
- Programar las **tareas en funciones o procedimientos**, definiendo claramente las pre y post condiciones sobre los datos a utilizar.
- Definir la **comunicar las tareas** a través de sus parámetros.
- Es necesario **documentar** las tareas y los módulos.
- Se debe evitar la utilización de **variables globales**.

Bibliografía

- Algoritmos, Datos y Programas con Aplicaciones en Pascal, Delphi, y Visual Da Vinci
  - De Giusti, Madoz, Bertone, Naiouf, Lanzarini, Gorga, Russo, Chapredonde.
  - Prentice Hall – Año 2001
- Metodología de la Programación
  - Cairó Osvaldo
  - Alfaomega – Año 2005
- Fundamentos de programación. Algoritmos y estructuras de datos
  - Joyanes Aguillar
  - Mc Graw Hill – Año 2003
- Apuntes de cátedra
  - Depetris- Aguil Mallea – Romano
  - Extensión y adaptación apunte UNLP - Año 2013