TEORIA 5

TEMAS de la CLASE

1. Concepto de Modularización
2. Procedimientos y Funciones

Etapas de resolución de un problema por computadora

Problema del Mundo Real

Análisis

Complejos
Extensos
Modificables

Modelo

Diseño

Solución Modularizada

Implementación

Verificación

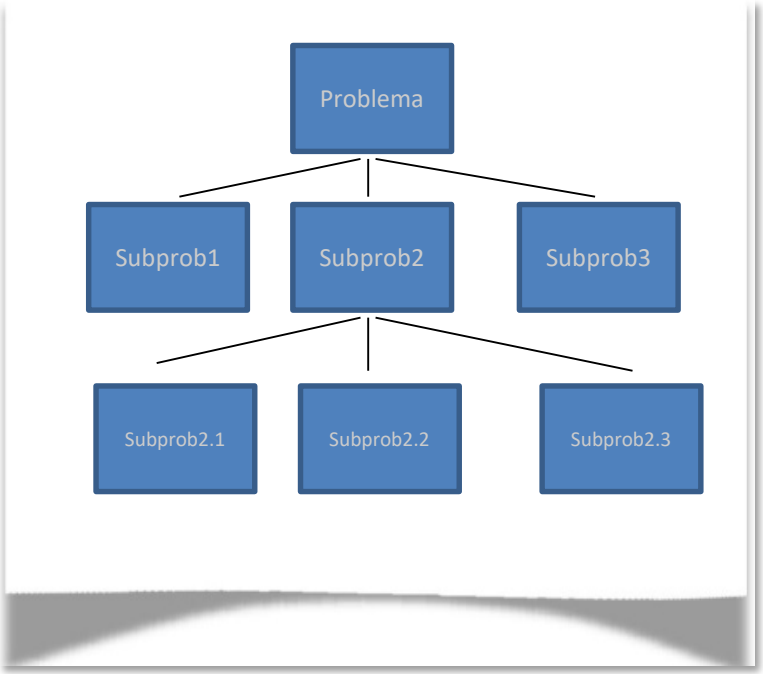
Metodología de diseño Top Down

Principio de "Divide y vencerás"

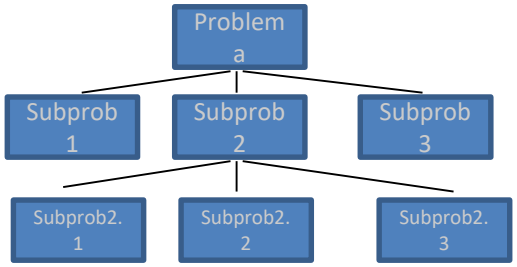
Descomponer el problema en partes (subproblemas) mas simples

Al descomponer un problema se debe tener en cuenta:

- Que cada subproblema resuelva una parte "bien" simple.
- Que cada subproblema pueda resolverse independientemente
- Que las soluciones a los subproblemas deben combinarse para resolver el problema original



¿Cuándo se detiene la descomposición del problema?



Permite distribuir el trabajo

Favorece el mantenimiento correctivo

Ventajas de la descomposición del problema

Facilita la reutilización dentro del mismo problema o en otro similar

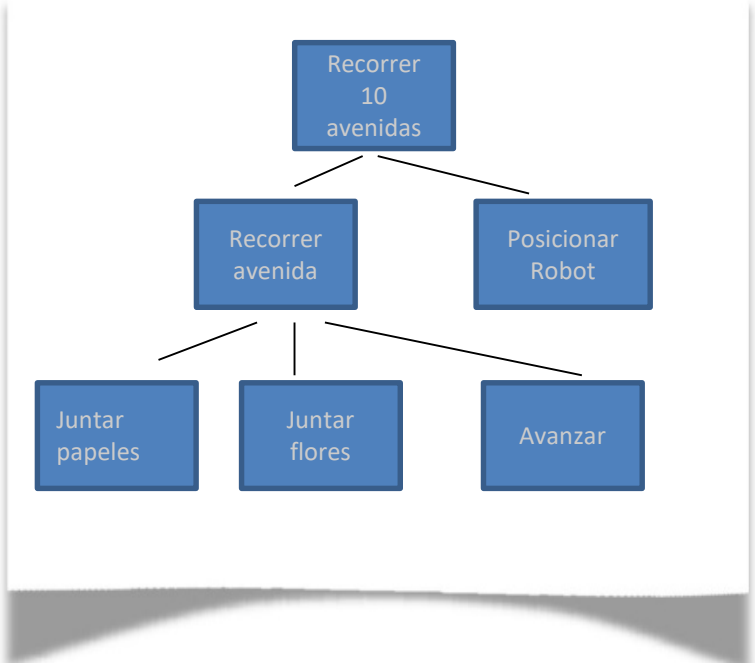
Facilita el crecimiento de los sistemas

Aumenta la legibilidad

Retomando el concepto de Modularización

La tarea de Modularizar implica dividir un problema en partes. Se busca que cada parte realice una tarea simple y pueda resolverse de manera independiente a las otras tareas.

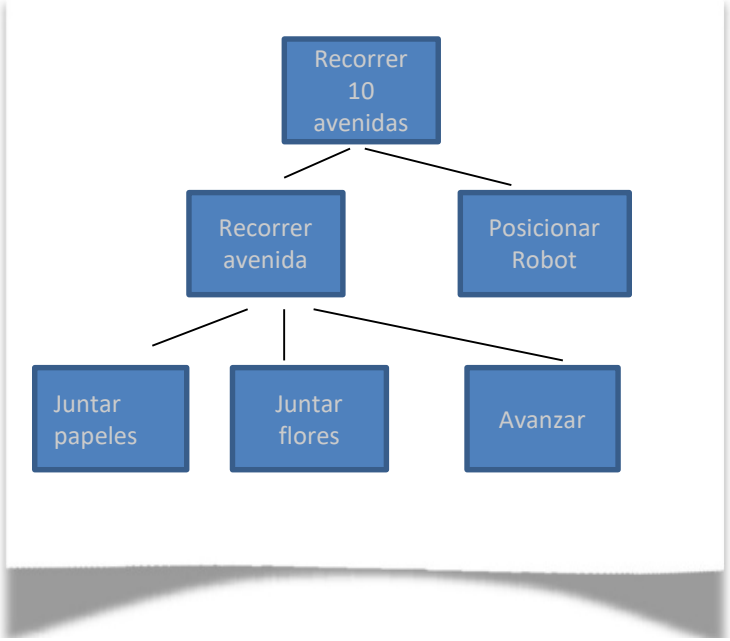
Recorrer 10 avenidas de la ciudad juntando flores y papeles



Módulo

Resuelve un subproblema particular

- ✓Define el conjunto de acciones
- ✓Define los datos necesarios

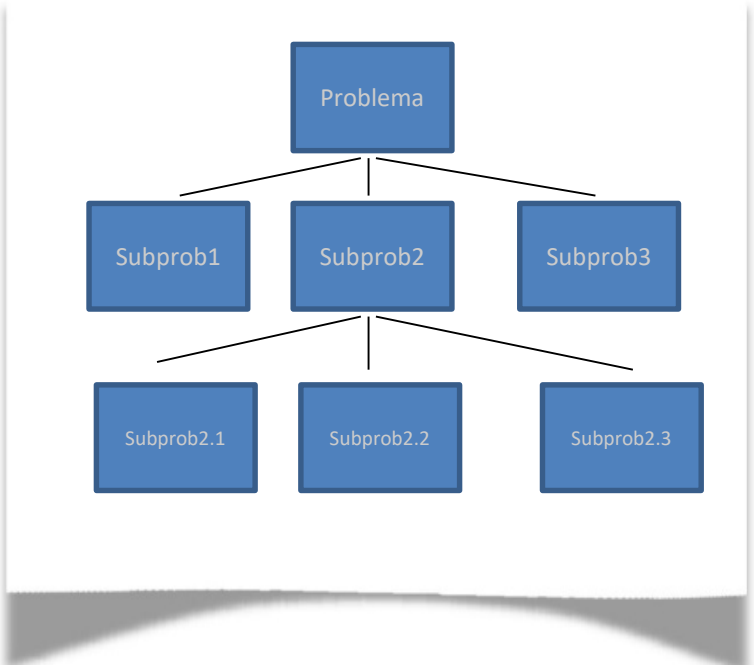


Módulo

- ¿Cómo lo hace? (implementación)
- ¿Qué hace? (objetivo)
- ¿Recibe datos de otros módulos?
- ¿Devuelve resultados?

Como resultado de la etapa de Diseño se tiene:

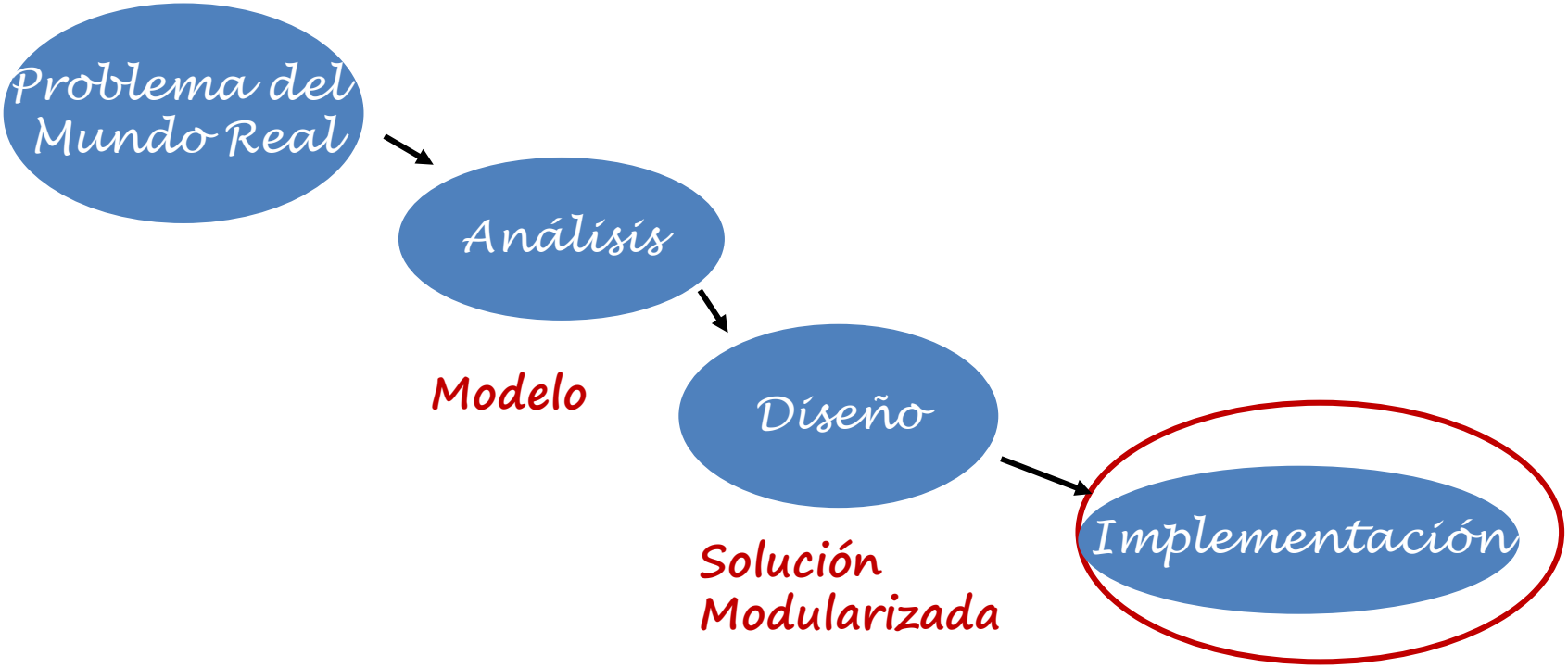
- Cuales son los módulos



- Cual es el objetivo de cada uno
- Cuales son los datos propios
- Cuales son los datos compartidos con Otros módulos.
- Cuales es el conjunto de acciones para alcanzar ese objetivo

Esta etapa no depende del lenguaje de programación que se use…

Avanzamos a la etapa de Implementación



- Se debe elegir el lenguaje de programación para escribir los algoritmos de cada módulo y la declaración de sus datos

- Los lenguajes de programación ofrecen diversas opciones para implementar la modularización.

Definición del módulo ¿Qué hace el módulo cuando se ejecuta?

- Encabezamiento (Interface)
  - Tipo de módulo
  - Identificación
  - Datos de comunicación
- Declaración de tipos
- Declaración de variables
- Sección de instrucciones ejecutables

Invocación del módulo ¿Cómo se hace cuando se quiere usar el módulo?

- Se debe conocer de qué manera se invoca o se llama al módulo para que ejecute sus acciones

La invocación puede hacerse mas de una vez

¿Qué ocurre con el flujo de control del programa?

¿Qué tipos de módulos ofrece Pascal?

PROCEDIMIENTOS (PROCEDURE)

FUNCIONES (FUNCTION)

Tienen características comunes, pero ciertas particularidades determinan cual es el mas adecuado para implementar un módulo particular

- ¿El módulo devuelve datos?
  - ¿Cuántos datos devuelve?
    - ¿De qué tipo son los datos que devuelve?
      - ¿Qué tipo de acciones ejecuta el módulo?

¿PROCEDURE?

¿FUNCTION?

¿Cuáles son los aspectos que los diferencian?

- Encabezamiento del módulo
- Invocación
- Lugar donde retorna el flujo de control una vez ejecutado el módulo

PROCEDURE

Conjunto de instrucciones que realiza una tarea específica y como resultado puede retornar 0, 1 o más valores.

¿Cómo se define el módulo?

```
Procedure nombre (lista de parametros);

Type
    ……

Var
    ……

begin
      ……..
      ……..
end;
```

Encabezamiento

Declaración de tipos internos del módulo (opcional)

Declaración de variables internas del módulo(opcional)

Sección de instrucciones

PROCEDURE

Conjunto de instrucciones que realiza una tarea específica y como resultado puede retornar 0, 1 o más valores.

¿Cómo se invoca el módulo?

```
Program uno;
      ……..

procedure Calculo (Parámetros Formales);
Type
      ….
Var
      ……
Begin

……
…..

End;


Begin
   ……
    Calculo (parámetros actuales);
   ……
End.
```

PROCEDURE

Conjunto de instrucciones que realiza una tarea específica y como resultado puede retornar 0, 1 o más valores.

Luego de ejecutado el módulo ¿Qué instrucción se ejecuta?

¿Qué ocurre con el flujo de control del programa?

Luego de ejecutado el módulo, el flujo de control retorna a la instrucción siguiente a la invocación del módulo

```
Program uno;
      ……

   procedure Calculo (Parámetros Formales);
   Type
         ….
   Var
        ……
   Begin

   ……
   …..

   End;

Begin
   ……
   Calculo (parámetros actuales);
   ……
End.
```

Esquema general de un programa que utiliza módulos

```
program uno;

Const
{Declaración de constantes del programa}

Type
{Declaración de tipos definidos}

Var
{variables}

Procedure Calculo (parámetros formales);
……
Procedure Otro (parámetros formales);
...

Var
{variables}

Begin {Sección del programa principal}

……
End.
```

Zona de Declaración de constantes

Zona de Declaración de tipos

Zona de Declaración de variables accedidas desde la sección del programa y los módulos

Zona de Declaración de los módulos

Zona de Declaración de variables accedidas solo desde la sección del programa

Zona de Instrucciones ejecutables

Analicemos el alcance de Variables...

```
Program dos;
Var
 a, b: integer;

procedure calculo(parámetros formales);
  var
      x: integer;
  Begin
      x:= 9; a:= 100;
      write (x);
  End;

 Var
  h: char;
Begin
  a:= 80;
  b:= a * 2;
  h:= 'A';
  calculo(parámetros actuales);
End.
```

Variables globales

Parámetros

Variables Locales del módulo

Variable Local del programa

¿Dónde se pueden utilizar a y b?

¿Dónde se puede utilizar x?

¿Dónde se puede utilizar h?

¿Qué pasa si dentro de calculo se declara b: integer?

¿Qué pasa si dentro de calculo se declara b: char?

Los parámetros formales solo pueden utilizarse en el módulo dónde están declarados

Variables globales, locales y parámetros

- **Variable global:** su declaración se hace en la sección de declaración del programa principal, es decir fuera de todos los módulos del programa y podrá ser usada en el programa y en todos los módulos del mismo. Por lo tanto, podrían ser utilizadas para la comunicación entre el programa y los módulos.

- **Variable local al módulo:** su declaración se hace en un módulo particular y sólo podrá ser usada por ese módulo. Si este módulo contiene a su vez otros módulos, entonces esa variable podría ser también usada por todos los módulos interiores, si está declarada previo a ellos.

- **Parámetros:** son los datos que se utilizarán para la comunicación entre el programa y los módulos, de una manera explícita.

- **Variable local al programa:** su declaración se hace antes de la sección de instrucciones ejecutables del programa y después de la declaración de los módulos del programa. Su uso se limita a la sección de instrucciones ejecutables.

```
program alcance;
var a, b: integer;

  procedure uno(parámetros formales);
  var c: char;
    procedure dos (parámetros formales);
    var d: real;
    begin
      b := a * 2 + 4; d:= 2,5;
      writeln(a, b, c, d);
      uno (parámetros actuales);
    end;
  var h: integer;
  begin
   b := a * 2;
   dos (parámetros actuales);
   c:= 'A';
   writeln(a, b, c, h);
   tres (parámetros actuales);
  end;

  Procedure tres (parámetros formales);
  begin
    write ('Tres'); write (a, b, h);
    uno (parámetros actuales);
    dos (parámetros actuales);
  end;
```

```
Var x, y: integer;
begin
  x:= 5; y:= 20 mod 10;
  uno (parámetros actuales);
  dos (parámetros actuales);
  tres (parámetros actuales);
  writeln(x, y);
end.
```

Teniendo en cuenta el alcance de las variables y la visibilidad de los módulos, analizar:

¿Qué invocaciones son válidas?
¿Qué asignaciones son válidas?