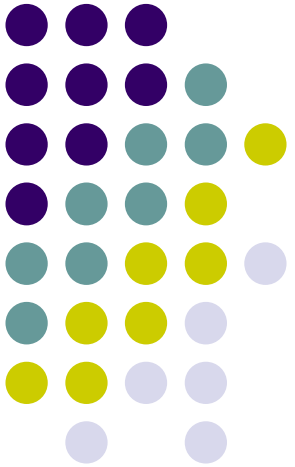
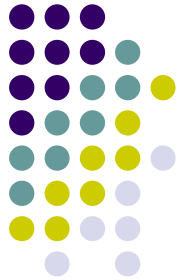
Programación y Computación

Análisis y Diseño de Algoritmos



Introducción



- A la hora de resolver un problema utilizando un computador NO se debe codificar directamente el programa en un lenguaje dado.

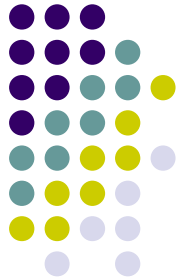
- Si el problema es complejo es muy difícil escribir un programa en un único paso.

Fases para resolver problemas

- Existe una serie de fases para resolver problemas a través de algoritmos que pueden resumirse como:

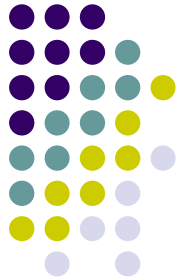
  - **Análisis**
  - **Diseño**
  - **Codificación**
  - **Pruebas**

Análisis



- Se trata de comprender la naturaleza del problema y NO de buscar una forma de resolverlo:
  - ¿Qué datos precisan ser introducidos para obtener la solución.?
  - ¿En que consistirá la solución. (Formulas, etc.)?
  - ¿Qué errores puede presentar.?
  - ¿Qué datos tendrá como salida.?
  - Etc.

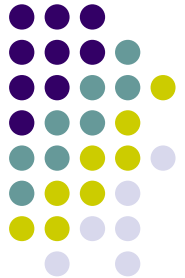
Diseño



- Esta fase busca una forma de resolver el problema, es decir, un algoritmo.

- Existen distintas técnicas para resolver problemas entre ellas están:
  - Diseño Descendente: Top – Down
  - Diseño Ascendente : Bottom - Up

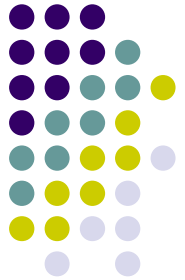
Diseño Descendente: Top – Down



- Se basa en el principio de “Divide y Venceras”.

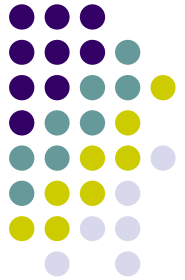
- También conocida como de arriba-abajo, consiste en establecer una serie de niveles de mayor a menor complejidad que den solución al problema.

Diseño Descendente: Top – Down



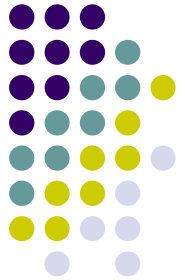
- En el diseño Top-down se formula un resumen del sistema, sin especificar detalles. Cada parte del sistema se refina diseñando con mayor detalle. Cada parte nueva es entonces redefinida, cada vez con mayor detalle, hasta que la especificación completa es lo suficientemente detallada para validar el modelo.

Diseño Descendente: Top – Down



- La utilización de la técnica de diseño Top-Down tiene los siguientes objetivos básicos:
  - Simplificación del problema y de los subprogramas de cada descomposición.
  - Las diferentes partes del problema pueden ser programadas de modo independiente e incluso por diferentes personas.
  - El programa final queda estructurado en forma de bloque o módulos lo que hace mas sencilla su lectura y mantenimiento.

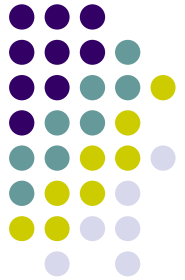
Diseño Ascendente: Bottom – Up



- El diseño ascendente se refiere a la identificación de aquellos procesos que necesitan computarizarse a medida que aparecen para satisfacer el problema inmediato.

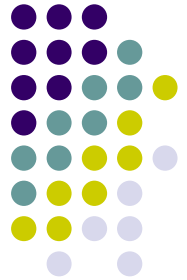
- Cuando la programación se realiza internamente y haciendo un enfoque ascendente, es difícil llegar a integrar los subsistemas al grado tal de que el desempeño global, sea fluido.

Diseño Ascendente: Bottom – Up



- Los problemas de integración entre los subsistemas son sumamente costosos y muchos de ellos no se solucionan hasta que la programación alcanza la fecha limite para la integración total del sistema. En esta fecha, ya se cuenta con muy poco tiempo, presupuesto o paciencia de los usuarios, como para corregir aquellas delicadas interfaces, que en un principio, se ignoran..

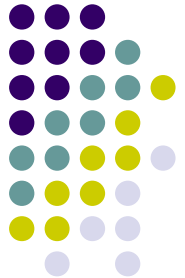
Diseño Descendente o Ascendente:



- La diferencia entre estas dos técnicas de programación radica en el resultado que presentan frente a un problema dado.

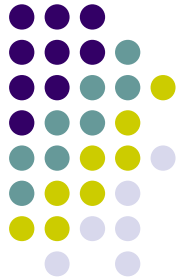
- Imagine una empresa, la cual se compone de varios departamentos (contabilidad, marketing, etc.), en cada uno de ellos se fueron presentando problemas los que se solucionaron basándose en un enfoque ascendente (Bottom Up): creando programas que satisfacían sólo el problema que se presentaba.

Diseño Descendente o Ascendente:



- Como no hubo un previo análisis, diseño de una solución a nivel global en todos sus departamentos, centralización de información, que son características propias de un diseño Descendente (Top Down) y características fundamentales de los sistemas; la empresa no pudo satisfacer su necesidad a nivel global.
- La creación de algoritmos se basa sobre la técnica descendente, la cual brinda el diseño ideal para la solución de un problema.

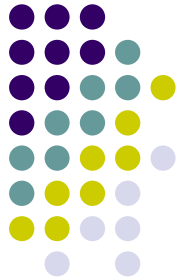
Utilizando Diseño Descendente: Top – Down



- En primer lugar se plantea el problema empleando términos del mismo problema (Nivel de Abstracción I)

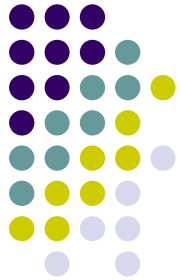
- En segundo lugar, se descompone en varios subproblemas expresados también en términos del problema y tratando de hacerlos lo mas independientes entre sí como sea posible.

Utilizando Diseño Descendente: Top – Down



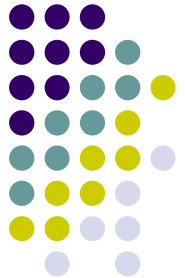
- Este paso se repite para cada subproblema tantas veces como sea necesario hasta llegar a una descripción del problema que emplee instrucciones sencillas que puedan ser transformadas a código en un lenguaje de programación.

Codificación



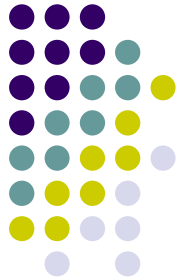
- Es la fase de programación o implementación propiamente tal. Aquí se implementa el código fuente, haciendo uso de prototipos así como pruebas y ensayos para corregir errores.
- Dependiendo del lenguaje de programación y su versión se crean las librerías y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.

Pruebas



- Los elementos, ya programados, se ensamblan para componer el sistema y se comprueba que funciona correctamente antes de ser puesto en explotación.

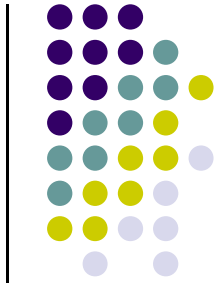
Ejemplo Usando Técnica Top-Down



Problema:

Determinar si un año ingresado por el usuario es bisiesto.

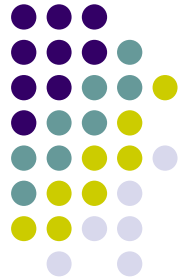
Ejemplo Usando Técnica Top-Down



Análisis:

- Entradas: El usuario debería digitar un año, un año es un numero positivo.
- Proceso: Un año es bisiesto si es múltiplo de 4 pero no de 100, la excepción son los años múltiplos de 400.
- Salida: Hay dos posibilidades “El año es bisiesto” y “El año no es bisiesto”
- Condición de Error: Si el dato introducido no es válido (número negativo o cero) debería indicarse “Dato no válido”.

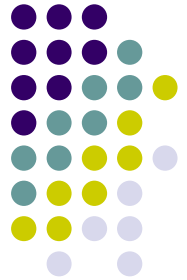
Ejemplo Usando Técnica Top-Down



Diseño Nivel I:

- Determinar si un año indicado por el usuario es un año bisiesto o no

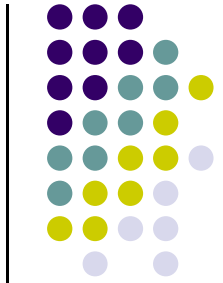
Ejemplo Usando Técnica Top-Down



Diseño Nivel II:

1.- Solicitar un año al usuario
2.- Determinar si el año es bisiesto o no
3.- Indicar al usuario el resultado obtenido

Ejemplo Usando Técnica Top-Down



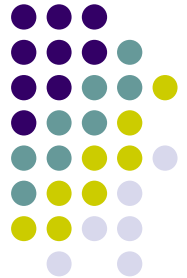
Diseño Nivel III:

1.1 Dar mensaje al usuario solicitando un año
1.2 Leer el año
1.3 Si el año no es válido indicárselo al usuario

2.1 Si el año no es multiplo de 4, no es bisiesto
2.2 Si el año es multiplo de 4, pero no de 100, es bisiesto
2.3 Si el año es múltiplo de 400, es bisiesto

3.1 Si el año es bisiesto dar el mensaje “El año es bisiesto”
3.2 Si el año no es bisiesto dar el mensaje “El año no es bisiesto”

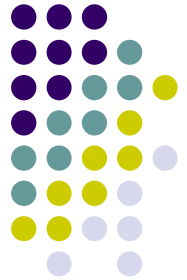
Ejemplo Usando Técnica Top-Down



Diseño Nivel IV:

```
ESCRIBIR 'Por favor, déme un año'
LEER AÑO
SI AÑO <= 0 ENTONCES
    escribir 'El año no es válido'
SINO
 SI el año es múltiplo de 4 ENTONCES
      SI el año es múltiplo de 400 ENTONCES
           BISIESTO ← si
      SINO
          SI el año es múltiplo de 100 ENTONCES
               BISIESTO ← no
          SINO
               BISIESTO ← si
          FINSI
      FINSI
      SINO
          BISIESTO ← no
      FINSI
      SI BISIESTO = si entonces
          escribir 'El año es bisiesto'
      SINO
          escribir 'El año no es bisiesto'
      FINSI
FINSI
```

Ejercicios Usando Técnica Top-Down



- Realice las fases de análisis y diseño para los siguientes problemas:
  - Construya una aplicación que convierta una longitud expresada en centímetros a su equivalente en pulgadas.
  - Construya una aplicación que dado 3 números permita saber si la suma de cualquier pareja de ellos es igual al tercero.
  - Escriba una aplicación que sume e imprima la serie 3,6,9,12,…99.