Capítulo 4 – Técnicas de análisis y diseño de algoritmos

4.1 – Introducción

- La calidad de un programa requiere esencialmente un funcionamiento correcto, una buena documentación y ser eficiente.
- El diseño descendente y un refinamiento sucesivo debería conducir a la obtención de buenos programas.
- Hoy en día la legibilidad es, sin duda, uno de los criterios más sobresalientes a la hora de decidir si un programa es o no bueno.
- Gracias a los avances del Hw, el estilo de programación se ha vuelto más importante que el ahorro de memoria y el tiempo de ejecución.

Capítulo 4 – Técnicas de análisis y diseño de algoritmos

4.2 – Estilo de programación

- El estilo en la construcción de programas, no es una cosa que pueda adquirirse sólo a partir del conocimiento de las reglas de sintaxis de un lenguaje y las técnicas básicas de programación
- No obstante, es posible reunir la inventiva y la ingeniosidad con unas reglas de disciplina y orden en el diseño de programas.
- El estilo de la buena programación está íntimamente unido con la legibilidad de los programas.
- La legibilidad es la clave para la comprensión de un programa, un programa que no se puede comprender no se podrá modificar ni mantener actualizado.

Capítulo 4 – Técnicas de análisis y diseño de algoritmos

4.2 – Estilo de programación

- La calidad de un programa se puede medir con diferentes parámetros, entre ellos, destacaremos:
- El programa debe funcionar
  - Un programa se dice que funciona correctamente cuando proporciona unos resultados correctos para todo el rango posible de datos de entrada y en todos sus estados.
- La documentación
  - Es muy importante que el programa esté documentado. La documentación sirve para ayudar a comprender y utilizar un programa. Puede ser interna (comentarios) o externa (manuales, guías…)

Capítulo 4 – Técnicas de análisis y diseño de algoritmos

4.2 – Estilo de programación

- La eficiencia
  - Antiguamente los programadores gastaban mucho tiempo en ahorrar espacio de memoria y disminución de tiempos de ejecución.
  - Hoy, aunque se debe tener presente siempre el ahorro de memoria y el tiempo de ejecución, los avances de Hw hacen que esta variable no sea tan exigente como en tiempos pasados.
- La corrección
  - El programa debe coincidir exactamente con las especificaciones: proporcionar la respuesta correcta para cualquier rango válido de datos.
- La flexibilidad
  - Los programas deben permitir cambios con ligeros retoques.

Capítulo 4 – Técnicas de análisis y diseño de algoritmos

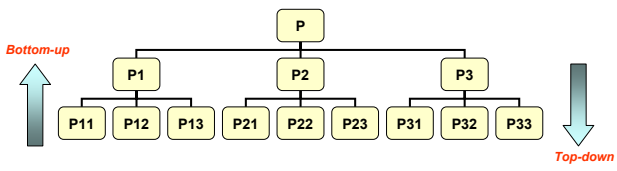
4.2 – Estilo de programación

- La fiabilidad
  - La exactitud y precisión de los resultados debe estar acorde con los tipos y rangos de datos y las características de proceso del computador.
- La presentación
  - El formato de la presentación del código de un programa es decisivo a la hora de mejorar su legibilidad. Hay que poner especial atención a los comentarios, identación (sangría) y nombres de la variables.

Capítulo 4 – Técnicas de análisis y diseño de algoritmos

4.3 – Estructura del programa

- Se recomienda descomponer el programa en módulos. Cada módulo representa una actividad completa y se codifica independientemente.



- El desarrollo se puede hacer mediante el método “Top-down” o “Bottom-up”.

Capítulo 4 – Técnicas de análisis y diseño de algoritmos

4.3 – Estructura del programa

- Top-Down
  - Es una técnica para diseñar que consiste en tomar el problema en forma inicial como una cuestión global y descomponerlo sucesivamente en problemas más pequeños y por lo tanto, de solución más sencilla.
  - La descomposición del problema original (y de las etapas subsecuentes), puede detenerse cuando los problemas resultantes alcanzan un nivel de detalle que el programador o analista pueden implementar fácilmente.

Capítulo 4 – Técnicas de análisis y diseño de algoritmos

4.3 – Estructura del programa

- Bottom-Up
  - Esta técnica consiste en partir de los detalles más precisos del algoritmo completando sucesivamente módulos de mayor complejidad, se recomienda cuando ya se cuenta con experiencia y ya se sabe lo que se va a hacer.
  - Conforme se va alcanzando el desarrollo de módulos más grandes se plantea como objetivo final la resolución global del problema.
  - Este método es el inverso del anterior y es recomendable cuando se tiene un modelo a seguir o se cuenta con amplia experiencia en la resolución de problemas semejantes.

Capítulo 4 – Técnicas de análisis y diseño de algoritmos

4.4 – Complejidad de los algoritmos

```
   void buscar(int c,float a[n]){
     int j;
1:   j=0;                          1 asignación = 1OE
2:   while(a[j]<c)&&(j<n-1){       1 acceso vector + 2 cond + 1 and + 1 dec= 5OE
3:    j=j+1;                       1 incremento + 1 asignacion = 2OE
4:   }
5:   if (a[j]==c)                  1 cond + 1 acceso vector = 2OE
6:     return j;                   1 return= 1OE
7:   else
8:     return 0;                   1 return= 1OE
   }
```

- Mejor Caso

T(n)=1(1) + 2(2) + 2(5) + 1(6 u 8) = 6

- Peor Caso

T(n)=1(1) + n-1(2 y 3) + 5(2) + 2(5) + 1(6 u 8) = 1 + ($\sum_{i=1}^{n-1} 5+2$) + 5 + 2 + 1= 7n + 2

Orden de complejidad O(n)