GUÍAS BÁSICAS PARA LA ESPECIFICACIÓN DE ALGORITMOS EN SEUDOCÓDIGO

Es la forma natural en la que los profesionales de la computación especifican soluciones algorítmicas independientes de cualquier lenguaje.

El proceso de resolución de problemas que tiene como objetivo final la construcción de un programa de computación, consta de tres etapas bien definidas:

1. ***Descripción de la solución en lenguaje natural***

Se escribe en forma abstracta y de "alto nivel". En esta etapa, el algoritmo opera sobre un modelo de la realidad.

2. ***Especificación del algoritmo en un seudocódigo formal detallado.***

Se comienza por indicar claramente las precondiciones y postcondiciones (tratando al algoritmo como una "caja negra"). Luego se escribe el seudocódigo en sí mismo (basado en las precondiciones y orientado a satisfacer las postcondiciones), detallando más precisamente el modelo o eventualmente eligiendo alguna estructura de datos en particular. Las precondiciones y postcondiciones son fundamentales para:

- completar la comprensión del algoritmo,
- identificar y entender las condiciones de borde,
- diseñar pruebas para verificar la corrección de los resultados esperados.

Las **precondiciones** indican el *estado en que* ***debe*** *encontrarse* el objeto sobre el cual opera el algoritmo *antes* de que éste sea ejecutado, para que el algoritmo funciones correctamente. Se entiende por ***estado*** la combinación de valores concretos de los atributos o variables internas del objeto.

Las **postcondiciones** indican el nuevo ***estado*** en que el objeto quedará una vez que el algoritmo termine de ejecutarse correctamente.

3. ***Escritura del programa***

Desarrollo del programa propiamente dicho, en algún lenguaje de programación, haciendo uso de las estructuras de datos que es posible definir en el mismo.

Si bien este es siempre el proceso natural que sigue cualquier diseñador de software, el nivel de detalle con el que se escriba cada una de las fases dependerá tanto de la experiencia del programador como de la complejidad de solución concebida.

Además, se trata de un proceso gradual de refinamiento, en el cual la descripción inicial se va enriqueciendo en detalles hasta terminar en la segunda etapa como una descripción formal y detallada del algoritmo solución, que debe poder ser interpretada por cualquier programador y ser implementada prácticamente en cualquier entorno de desarrollo.

No existe un formato general de seudocódigo, y puede encontrarse una gran cantidad de variantes en la literatura técnica. Para la especificación de algoritmos en la asignatura Programación II, se han establecido una serie de reglas que se detallarán y se ilustrarán con un ejemplo a continuación.

Ejemplo de uso

Ilustraremos con un problema sencillo, cómo hallar el promedio de las notas obtenidas por los alumnos de Programación II en un parcial.

Paso 1: Lenguaje natural sobre un modelo conocido

Opción 1:

Se necesita tener la lista de notas (el modelo "lista", un concepto conocido o manejado por todos).

```
Se recorren todos los elementos de la lista (notas), sumando
cada nota a una variable.

Luego se divide esa suma por la cantidad de notas (cantidad de
elemento de la lista),
```
y ese es el promedio buscado.

Opción 2 (un poco más formal)**:**

Se necesita tener la lista de notas (el modelo "lista", un concepto conocido o manejado por todos).

```
Para Cada elemento de la lista:
     Sumar la nota a una variable
Fin Para Cada
Dividir esa suma por la cantidad
```
de notas

Paso 2: Seudocódigo formalizado.

Decidimos implementar la lista sobre un vector o arreglo ("array"). Sea el vector notas, siendo notas[i] una nota cualquiera del arreglo.

PRECONDICIONES:

- El arreglo "notas" de tamaño adecuado y no vacío contiene todas las notas, la primera en la posición 0 y las otras en las siguientes posiciones en forma consecutiva.
- La variable "cantidadDeNotas" es la cantidad de notas que hay en la lista.

POSTCONDICIONES:

- La variable "suma" tendrá la suma de todas las notas que estén en la lista
- La variable "i" (variable de control del bucle) tendrá el valor "cantidadDeNotas".
- El cociente de la suma sobre la cantidad de notas es el promedio buscado – devuelto por el algoritmo -.

ALGORITMO EN SEUDOCÓDIGO:

```
obtenerPromedio(notas)
Comienzo
      suma ß 0
      i ß 0
      mientras i < cantidadDeNotas hacer
            suma ß suma + notas[i]
            i ß i + 1
      fin mientras
      devolver (suma/cantidadDeNotas)
Fin
```

Este algoritmo es claro y preciso, puede verificarse perfectamente su corrección, y está libre de detalles de los lenguajes de programación que no aportan a la esencia del problema.

El seudocódigo debe escribirse lo más claro posible, de forma que pueda ser rápidamente comprendido y pasar al Paso 3.

Paso 3: Codificación con algún lenguaje de programación.

El algoritmo escrito es fácil y rápidamente traducible a cualquier lenguaje de programación.

Estándares básicos para la escritura de sentencias de seudocódigo

Si bien la indentación ayuda a la comprensión, debe definirse claramente el ámbito de aplicación de cada sentencia especificando claramente su fin.

Sentencias de control de flujo:

Como **sentencia de selección** se usará la clásica, y en el caso de anidaciones escribir el fin de cada una de ellas de forma de contribuir a la claridad del algoritmo:

```
SI <condición1> ENTONCES
      <bloque de sentencias>
      SI <condición2> ENTONCES
            <bloque de sentencias>
      FINSI
SINO
      <bloque de sentencias>
FIN SI
```

Como **sentencias de repetición** se preferirá el tipo:

```
MIENTRAS <condición> HACER
      <bloque de sentencias>
FIN MIENTRAS
```

Esta sentencia tiene como particularidad que la comprobación de la condición (simple o compuesta) se realiza al ***inicio*** del bloque y sobre una variable de control de bucle ("*vcb*") que debe inicializarse antes (para mayor claridad inmediatamente antes) de la prueba de la condición, y que debe ser actualizada como última sentencia del bloque, para que en algún momento haga falsa la condición:

```
inicializar vcb (por ejemplo i ß 0)
MIENTRAS <condición de la vcb sea verdadera> HACER
      <bloque de sentencias>
      actualizar vcb (por ejemplo i ß i + 1)
FIN MIENTRAS
```

La actualización de la *vcb* en cualquier otro lugar del bloque debe manejarse con mucha cautela.

También puede usarse la sentencia que hace la comprobación de la condición al final del bloque:

```
inicializar vcb
HACER
      <bloque de sentencias>
      actualizar vcb
MIENTRAS <condición de la vcb sea verdadera>
```

En general ***no se aconseja*** el uso del tipo de sentencia **"for"** que ofrecen la mayoría de los lenguajes y en la se compacta en una sola línea la inicialización de la *vcb*, la condición de terminación y la actualización de la *vcb*. La semántica de la sentencia puede variar levemente de un lenguaje a otro lo que ocasionaría dificultad de comunicación con otros analistas.

A diferencia de un lenguaje de programación, el objetivo del seudocódigo ***no es*** *compactar y simplificar código fuente*, sino todo lo contrario: ***especificar claramente soluciones algorítmicas, independientes del lenguaje de programación***.

Variables

- Usar nombres significativos. Se desaconseja usar nombres cortos no ilustrativos, excepto tal vez para las variables de control de bucles.
- Usar notación "camello", por ejemplo: unAlumno, unCliente, posInicial, cantidadElementos,
- Indicación de los tipos de datos: Pueden omitirse en caso de ser predefinidos, obvios, triviales o siempre que no aporten al conocimiento o concepto que se desea expresar

Operadores

Aritméticos / lógicos básicos: +, - *. /, AND, OR, XOR, NOT….

Comparación: <, >, <=, >=, <>, =

Asignación: ß