El diagrama de flujo debe ilustrar gráficamente los pasos o procesos que se deben

seguir para alcanzar la solución de un problema. Los símbolos presentados, colo

cados en los lugares adecuados, permiten crear una estructura gráfica flexible

que ilustra los pasos a seguir para alcanzar un resultado específico. El diagrama

de flujo facilita entonces la escritura del programa en un lenguaje de programa

ción, C en este caso. A continuación se presenta el conjunto de reglas para la

construcción de diagramas de flujo:

1. Todo diagrama de flujo debe tener un inicio y un fin.

2. Las líneas utilizadas para indicar la dirección del flujo del diagrama deben

ser rectas: verticales u horizontales.

3. Todas las líneas utilizadas para indicar la dirección del flujo del diagrama

deben estar conectadas. La conexión puede ser a un símbolo que exprese

lectura, proceso, decisión, impresión, conexión o fin del diagrama.

4. El diagrama de flujo debe construirse de arriba hacia abajo (top-down) y de

izquierda a derecha (right to left).

5. La notación utilizada en el diagrama de flujo debe ser independiente del

lenguaje de programación. La solución presentada se puede escribir poste

riormente en diferentes lenguajes de programación.

6. Al realizar una tarea compleja, es conveniente poner comentarios que

expresen o ayuden a entender lo que hayamos hecho.

7. Si la construcción del diagrama de flujo requiriera más de una hoja,

debemos utilizar los conectores adecuados y enumerar las páginas

correspondientes.

8. No puede llegar más de una línea a un símbolo determinado.

Tipos de datos:

Los datos que procesa una computadora se clasifican en simples y estructura

dos. La principal característica de los tipos de datos simples es que ocupan sólo

una casilla de memoria. Dentro de este grupo de datos se encuentran principal

mente los enteros, los reales y los caracteres.

Operadores Lógicas:

Operador

lógico Operación Ejemplos

Resultados

Negación

x = (!(7 > 15)); /* (!0) 1 1 */ x = 1

y = (!0); y = 1

Conjunción

```
x = (35 > 20) && (20 <= 23); /* 1 && 1 */ x = 1
y = 0 && 1; y = 0
```

Disyunción

```
x = (35 > 20) (20 <= 18); /* 1 0 */ x = 1
y = 0 1;
y
```