Reglas de indentación

A diferencia de la sintaxis del lenguaje de programación, que son reglas fijas que obligatoriamente hay que seguir, un *estilo de programación* está constituido por directrices que ayudan a obtener programas más legibles. Es por esto que, si bien no existen estilos de programación absolutamente correctos o incorrectos, es aconsejable la adopción, de una manera sistemática, de un conjunto de normas para la escritura de programas. Por ello, se propone que sigamos las siguientes reglas de indentación:

a) INDENTACIÓN

- Se debe utilizar un único modelo de indentación a lo largo de todo el programa.
- Los bloque de código (por ejemplo, dentro de un ciclo o el cuerpo de una función) deberán ir indentados.
- Si un bloque de código está anidado dentro de otro bloque de código, el bloque más interno deberá ir indentado respecto al externo.
- Para evitar que cuando haya múltiples bloques anidados, unos dentro de otros, el código sobrepase la anchura de la página se recomienda que el número de espacios de la indentación no sea excesivo (3 ó 4).

b) IDENTIFICADORES

Los identificadores o nombres de los elementos de un programa (por ejemplo, los nombres de las variables, constantes, funciones, etc.) deben ser coherentes con el valor que estén almacenando o manejando:

Constantes con nombre

Se escribirán en mayúsculas y si el identificador de la constante está formado por varias palabras, estas se separaran mediante el carácter subrayado. Ejemplos:

ESPACIO, MAX_FILAS

Variables e Identificadores de tipos definidos por el usuario

- Se utilizarán letras minúsculas, comenzando cada palabra que componga el identificador con mayúsculas. Ejemplos:

contador, horasTrabajadas, ctrolGeneral, contarLetras, mostrarResultados,

c) ESTRUCTURA GENERAL DE LAS PARTES DEL PROGRAMA

- Cada una de las partes del programa comenzará en la primera columna, indentando o sangrando las líneas que componen cada una de las partes del programa.
- Se separará cada una de las partes o bloques del programa con una línea en blanco.
- Así mismo, se recomienda dejar un espacio en blanco antes de cada punto y coma (**;**) utilizado para finalizar las sentencias de un programa.

d) COMENTARIOS

Cada programa debe tener un comentario general, que debe aparecer al comienzo del mismo. Se sugiere seguir el siguiente ejemplo:

```
/*====================================================================
| Nombre del programa: nomina.cpp
| Propósito: calcular la nomina de los empleados de la
|          juguetería “Mi alegría”
|
| Revisiones:
| Fecha    Nombre         Revisión
| -------- --------------- ---------------------------------------
| 02-03-98 Juan Pérez      Creado
|
====================================================================*/
```

Los comentarios entre las líneas de código, pueden hacerse de la siguiente forma:

```
/* Esto es un comentario simple */

// Esto es un comentario simple
```

Un comentario también puede ser escrito en varias líneas y en cualquier parte del programa:

```
/* Este es un comentario de varias líneas
de texto, y éste comentario puede tener la estructura
que cada uno desee. Lo que se escriba entre comentarios
será ignorado por el compilador */
```

e) BUENAS PRÁCTICAS

Además de todo lo mencionado anteriormente respecto al formato del código, intentaremos adquirir *buenas costumbres de programación* que hagan nuestros programas más fáciles de depurar, aparte de ser legibles y fáciles de entender. Entre estas buenas prácticas cabe destacar que debe evitarse lo más que se pueda el uso de variables globales, por lo cual es recomendable usar siempre variables locales, así como paso de parámetros.

NOTA: El manejo de variables locales y paso de parámetros lo estudiaremos en la última unidad del curso.

En resumen, las reglas de indentación son las siguientes:

- Siempre se indenta 4 espacios el código encerrado entre corchetes '{' y '}'.
- El primer corchete está en el mismo renglón que la instrucción.
- A partir del último corchete '}' se termina la indentación.

Lo correcto	¡¡¡ INCORRECTO !!!

`void main(int argc, char *argv[]) {`
`    // Código indentado dentro de la función`
`}`
	`void main(int argc, char *argv[])`
	`{`
	`    // Código indentado dentro de la función`
	`}`

`if (condición) {`
`    // una instrucción indentada`
`}`
	`if (condición)`
	`    // encierre esta instrucción sola en su bloque { }`

`if (condición) {`
`    // una instrucción indentada`
`}`
`else {`
`    // primera instrucción`
`    // segunda instrucción bien indentada`
`    // tercera instrucción`
`}`
	`if (condición)`
	`{`
	`    // no use un renglón extra para la instrucción`
	`} else { // no ponga el "else" en el mismo renglón`
	`    // segunda instrucción bien indentada`
	`}`

`while (condición) {`
`    // primera instrucción`
`    // segunda instrucción bien indentada`
`}`
	`while (condición) {`
	`     // Debe indentar 4 espacios, ¡no 5 ni 3!`
	`     // segunda instrucción mal indentada`
	`}`

`for (int i=1; i<=20; ++i) {`
`    // primera instrucción`
`}`
	`for(int i=1;i<=20;++i){//¡No sea necio!`
	`    // ¡Debe usar espacios entre las palabras!`
	`}`

`do {`
`    // primera instrucción`
`} while (condición);`
	`do`
	`{   // no use un renglón extra para la instrucción`
	`    // faltó el ';' al final del "while"`
	`} while (condición)`

`switch (value) {`
`    case 'a': case 'b':`
`        // primera instrucción`
`        // segunda instrucción`
`        break;`
`    case 'c':`
`        // primera instrucción`
`        break;`
`    default:`
`        // primera instrucción`
`        break;`
`}`
	`switch (value) {`
	`    case 'a':`
	`    case 'b':`
	`        // ahorre renglones`
	`        // segunda instrucción bien indentada`
	`        break;`
	`    case 'c': {`
	`            // sobra usar el bloque { }`
	`           break;`
	`        }`
	`    default:`
	`        // primera instrucción`
	`        break;`
	`}`

`list::iterator it = L.begin();`
`for ( ; it != L.end(); ++it ) {`
`    int  a = it->first;   // alineación`
`    long b = it->second;  // y espaciado`
 	
`    c = ( ((long)(a) - b) / 2 ) >> 5;`
`    delta = b - c`
`}`
	`list::iterator it;`
	`for(it=L.begin(), it!=L.end();++it) {`
	`    int a=it->first;      // ¡ Horrible !`
	`    long b=it->second;    // ¡ Todo pegado !`
	`    c=(long)(a)-b/2>>5;  delta = b - c`
	`}`

Fuente de Consulta: http://www.di-mare.com/adolfo/p/indentacion.htm