Anexo A. Estilo de tabulación

Contenido

Introducción

Un buen programa, además de **efectivo** y **rápido** debe estar **bien tabulado**, lo cual:

- Aumenta la **claridad** del programa, lo cual facilita su lectura y comprensión.

- Permite **depurar** el programa de forma más sencilla

- Permite **reutilizar** el código tanto por el creador como por otros programadores.

- Para tabular solamente hay que seguir una serie de **reglas básicas** sistemáticamente.

Reglas básicas de tabulación

Bloques y sub-bloques

- Al **inicio** de un bloque, introducimos un **nuevo tabulador**.
- Al **final** de un bloque volvemos al **tabulador anterior**.

Además:

- La llave de **apertura** se coloca a la **derecha** de la cabecera de la estructura.
- La llave de **cierre** se coloca **debajo** del inicio de dicha cabecera.

Reglas básicas de tabulación

Bloques y sub-bloques. Ejemplo:

```
//librerias y constantes
#include <stdio.h>
#include <stdlib.h>
//Hasta ahora estamos en el bloque global, luego todo va sin tabular
int main(void){
        //main define un nuevo bloque, por lo que todo su cuerpo se tabula a la derecha
        int a;
        for(a=1;a<10;a++){
                //la estructura for define un nuevo bloque
                printf("Hola\n");
                if(a<5){
                        //lo mismo sucede con el bloque del condicional
                        printf("mensaje");
                }
                //al cerrar el bloque del condicional disminuimos una
                //tabulación
        }
        //lo mismo sucede con el fin del bloque del for
        return(0);
}
```

Reglas básicas de tabulación

Estructuras de decisión

Teniendo en cuenta la regla anterior, el condicional simple queda:

```
if([condicion]){
        //bloque del SI, tabulado una unidad más
}else{
        //bloque del SINO, tabulado una unidad más
}
```

Ejemplo:

```
if(a>0){
    printf("Positivo\n");
}else{
    printf("Negativo o cero\n");
    if(a==0) {
        printf("Definitivamente cero\n");
    } else{
        printf("Definitivamente negativo\n");
    }
}
```

Reglas básicas de tabulación

Estructuras de decisión (múltiple)

El switch se tabula como si los casos fuesen bloques. Ejemplo:

```
switch(i){
case 0:
    //Instrucciones del caso 0
    printf("caso 0\n");
    break;
case 1:
    //Instrucciones del caso 1
    brintf("caso 1\n");
    break;
case 2:
    //Instrucciones del caso 2
    printf("caso 2\n");
    break;
default:
    //Instrucciones del default
    printf("Caso por defecto\n");
    break;
}
```

Reglas básicas de tabulación

Estructuras de repetición

Bloque tabulado, la llave de apertura a la derecha de la cabecera y la de cierre debajo del principio de ésta.

```
while([condicion]){
    //instrucciones del bloque con una
    //profundidad más de tabulación
}
```

Repetición con condición inicial

```
do{
    //instrucciones del bloque con una
    //profundidad más de tabulación
}while([condición]);
```

Repetición con condición final

```
for([inicio];[condicion];[incremento]){
    //instrucciones del bloque con una
    //profundidad más de tabulación
}
```

Repetición con contador

Reglas básicas de tabulación

Funciones

Las funciones se tabulan igual que el resto, teniendo en cuenta que contienen un bloque:

```
int mayor(int a, int b){
    int r;
    if(a>b){
      r=a;
    }else{
      r=b;
    }
    return(r);
}
int main(void){
    int a,b,c;
    a=5;
    b=7;
    c=mayor(a,b);
    return(0);
}
```

Reglas básicas de tabulación

Consideraciones adicionales

- La única forma de tabular es utilizar el **tabulador** [TAB]. No se debe cuadrar el programa nunca con espacios.
- En general, los **comentarios se tabulan** tal y como si fuesen instrucciones del programa.
- Todo lo que no esté dentro de una función no se tabula (**bloque general**)

```
//librerias y constantes
#include <stdio.h>
#define PI 3.14
// Comentarios relacionados con la funcion
int Funcion(void){
    //comentario relacionado con las variables
    int v1,v2;
    //comentario relacionado con el for
    for(v1=5;v1<10;v1++){
            //comentario relacionado con el printf
            printf("Mensaje\n");
    }
    //return también es una instrucción y se tabula como tal
    return(0);
}
```

Algunas excepciones

Bloques de una sola instrucción

Se pueden obviar las llaves y colocar la instrucción a la derecha de estructura:

```
if(a>b){
        printf("A es mayor que B\n");
}else{
        printf("B es mayor o igual que A\n");
}
```

se podría escribir:

```
if(a>b) printf("A es mayor que B\n");
else printf("B es mayor o igual que A\n");
```

Lo mismo se puede aplicar a bucles.

Algunas excepciones

Estructuras de if-else anidados

La profundidad de estas estructuras puede ser excesiva:

```
if(a<5){
    printf("Menor a 5\n");
}else{
    if(a<10){
        printf("Menor a 10\n");
    }else{
        if(a<15){
            printf("Menor a 15\n");
        }else{
            if(a<20){
                printf("Menor a 20\n");
            }else{
                printf("Mayor o igual a 20\n");
            }
        }
    }
}
```

Algunas excepciones

Estructuras de if-else anidados

Eliminando llaves innecesarias se puede escribir de forma más compacta:

```
if(a<5){
    printf("Menor a 5\n");
}else if(a<10){
    printf("Menor a 10\n");
}else if(a<15){
    printf("Menor a 15\n");
}else if(a<20){
    printf("Menor a 20\n");
}else{
    printf("Mayor o igual a 20\n");
}
```