

Lab. Computación I

BUENAS PRÁCTICAS DE PROGRAMACIÓN

¿Qué son?

- ▶ Las buenas prácticas de programación son un conjunto formal o informal de reglas, pudiendo ser opcionales u obligatorias, que se adoptan con el fin general de mejorar la calidad del software.

Permiten

- ▶ Facilitar el proceso de desarrollo para el programador.
- ▶ Aumentar o mejorar la legibilidad y mantenibilidad del código fuente, lo que ayuda a otros desarrolladores—o a versiones futuras del autor inicial— a comprender el software.
- ▶ Evitar cierta clase de errores comunes, por ejemplo equivocarse por 1 en los límites de una iteración
- ▶ Entre otras más....

En general cuando desarrollamos debemos:

- ▶ Priorizar la legibilidad
- ▶ Colocar comentarios
- ▶ Testear tu código
- ▶ Simplificar al máximo
- ▶ Realizar control de versiones
- ▶ No reproducir fragmentos idénticos de código

En particular en C...

Puntos a tener en cuenta respecto la estructura del programa

- Utilice indentación para todas las operaciones que se realizan dentro de una estructura de control, tal como un if, for, while o do-while.

```
while (n != 0) {  
    d = n%10;  
    n = n/10;  
    if(d > 5) {  
        printf("%d", d);  
    }  
}
```




```
while (n != 0) {  
    d = n%10;  
    n = n/10;  
    if(d > 5) {  
        printf("%d", d);  
    }  
}
```




Resalta la estructura lógica del código y simplifica su lectura

- Utilice llaves en todas las estructuras de control—incluso si tienen sólo 1 instrucción en su interior.

```
for(i = 0; i < n; i += 1)
    if(i % k == 0)
        suma += arreglo[i];
```



```
for(i = 0; i < n; i += 1) {
    if(i % k == 0) {
        suma += arreglo[i];
    }
}
```



Esto le ayudará a evitar problemas si posteriormente desea agregar más instrucciones—típicamente printf— bajo dicha estructura de control

- Utilice consistentemente un estilo de apertura y cierre de llaves:

```
if (a > 10) {
    printf("lol");
}
```

```
if (a > 10)
{
    printf("lol");
}
```

```
if (a > 10)
{
    printf("lol");
}
```

- Utilice sólo una instrucción por línea.

```
while (n != 0) {  
    d = n%10; n = n/10;  
    if(d > 5) {  
        printf("%d", d); suma += d;  
    }  
}
```



```
while (n != 0) {  
    d = n%10;  
    n = n/10;  
    if(d > 5) {  
        printf("%d", d);  
        suma += d;  
    }  
}
```



- Utilice un espacio después de las comas

```
int x,y,suma,digitos,factorial;  
if (x < pow(y,suma)) {  
    printf("%d %d %d",x,y,z);  
}
```



```
int x, y, suma, digitos, factorial;  
if (x < pow(y, suma)) {  
    printf("%d %d %d", x, y, z);  
}
```



Entre otros....

Sobre las variables

- Utilice nombres significativos para sus variables.

```
int x, f, z
scanf("%d", &z);
f = z;
while(f != 0) {
    x += f % 10;
    x = x / 10;
}
```



```
int n, temp, sumaDigitos;
scanf("%d", &n);
temp = z;
while(temp != 0) {
    sumaDigitos += temp % 10;
    temp = temp / 10;
}
```



La idea es que la lectura del código sea “auto-explicativa”, lo que facilita su posterior modificación y comprensión

- Escoja una opción entre camelCase y snake_case para los nombres de variables que tienen más de una palabra.
- Todas las variables deben ser inicializadas con algún valor.

Sobre los comentarios

- ▶ Cada programa debe comenzar con un comentario que describa su propósito, los supuestos realizados.
- ▶ Cada bloque de código debe ser precedido por un comentario que explica la función de ese bloque, su propósito.
- ▶ Los comentarios deben ser consistentes con el código. Es muy probable que el programa sea incorrecto si el código y los comentarios no coinciden.
- ▶ Actualizar los comentarios cada vez que cambia el código fuente.
- ▶ Los comentarios deben ser correctos gramaticalmente y ortográficamente.
- ▶ Escriba comentarios para todo lo que no sea completamente obvio al leer el código fuente
- ▶ Ponga un espacio después de la apertura del comentario, y antes del cierre del mismo.

```
/*Este comentario es mas dificil de leer*/  
/* Este comentario es mas facil de leer */
```


Tipos de comentarios

► Única línea

```
/* Este comentario es mas facil de leer */
```

► Múltiple línea

```
/*  
 * Este es un comentario,  
 * con multiples lineas.  
 */
```

► Bloque

```
/*  
 * La primera linea viene despues de una linea vacia.  
 *  
 * Los otros parrafos van separados por una linea vacia  
 * y tambien puede haber una linea vacia al final (opcionalmente).  
 *  
 */
```