# Ejemplos y Consejos Sobre C Para Estudiantes Del E.E.T. #3

acarlstein.com/

Posted by Alejandro G. Carlstein Ramos Mejia on October 31, 2014 October 31, 2014 About Programming / ANSI/POSIX C

**Notificación:** Estos ejemplos y consejos son proveídos a usted(es) con el objetivo de educarlo(s). El uso de esta información esta bajo la responsabilidad de usted(es). Yo no tomo responsabilidad en como usted(es) utilicen esta información.

Yo fui estudiante de la Escuela de Enseñanza Técnica #3 de San Isidro y recientemente se me dio la oportunidad de poder ayudar algunos estudiantes con sus proyectos relacionados con electrónica y programación. Este articulo fue escrito para la clase del día Martes 4 de Noviembre de 2014.

Estos consejos y ejemplos no tienen un orden especifico. Si encuentro mas consejos o ejemplos los agregare a este articulo.
Tambien a medida que vea el nivel de programacion de los estudiantes iré agregando mas información acorde a sus conocimientos.

Estos consejos y ejemplos están escritos en C.

1. Todos los archivos de código tendrían que empezar con una introducción:

```
/**
 * Titulo del Programa o Código
 * Nombre de Clase Cursando
 * Autores: Navoleti Chagorta, Pepone Huecone
 * Creado: Marzo 31, 2014
 * Modificado: Abril 02, 2014
 * Descripción: Este código hace a Jarvis hablar por el parlante.
 */
```

   Esta introducción es una forma legal de proteger el código como también permite a otro programador tener una idea general del código que vera a continuación.

2. Elijan el estilo de escritura para las frases o palabras compuestas usadas en funciones y variables.
   Un estilo utiliza todas las letras en minúscula separadas por un “_”:

```
int color_red = color_to_integer("#FF0000");
```

   Otro estilo se llama CamelCase donde las frases o palabras se conectan juntas y la primera letra de cada palabra es mayuscula:

```
int colorRed = colorToInteger("#FF0000");
```

   Elijan un solo estilo y utilícenlo a lo largo del programa.

3. Las contantes, valores que nunca cambian, van en frases y palabras todas en muscula separadas por “_”:

```
#define COLOR_RED "#FF0000"
const int COLOR_BLUE = 0xFF;
```

4. Los nombres de las variables tienen que describir que valor se esta guardando. Ejemplo:

```
const char* MENSAJE_ERROR_NO_MEMORIA = "[X] Error: Out of Memory\n";

int cociente = 7;
int divisor = 5;
int resto = cociente % divisor;
```

5. Intenten crear pequeñas funciones que sean cortas, que realicen una sola cosa, y que el nombre describa lo mejor posible la funcionalidad de la función.

```
void intercambioDeValores(int *primerValor, int *segundoValor){
    int temporario;
    temporario = *primerValor;
    *primerValor = *segundoValor;
    *segundoValor = temporario;
}
```

   Esta función realiza un intercambio de los valores guardados en dos variables y no retorna nada.
   las variables “a” y “b” son “punteros” y se explicaran mas adelante como trabajan.
   Al hacer las funciones cortas y enfocadas a un solo trabajo, uno puede re-usarlas, fácilmente repararlas, y reducir la cantidad de comentarios

6. La función “debug” funciona como “fprint”, con la única diferencia que si el primer parámetro no es 1 entonces no imprime.
   Dependiendo del compilador, este codico capaz que no funcione
   Ustedes pueden mejorar la funcion “debug” para que les de otros mensaje por ejemplo.
   Eso se los dejo a ustedes para que lo modifiquen como mas les convenga.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>

// 0 = hide, 1 = show
#define TOGGLE_DEBUG 1

int main(){
 debug(TOGGLE_DEBUG, "function: %s", "main()");
 debug(TOGGLE_DEBUG, "El numero es %d", 5);
 return 0;
}

void debug(int doShow, char *fmt, ...){
    if (doShow == 1){
        va_list argp;
        fprintf(stdout, "[DBG] ");
        va_start(argp, fmt);
        vfprintf(stdout, fmt, argp);
        va_end(argp);
        fprintf(stdout, "\n");
    }
}
```

Esto tendria que imprimir:
[DBG] function main()
[DBG] El numero es 5

7. Para cargar la libreria de matematicas utilizen esta linea de codigo (si estan usando gcc como compilador):

```
gcc yourfile.c -lm
```

8. Comentarios tendrían que ser usados solamente cuando el código podría ser confuso para otra gente:

```
#define M_PI 3.14159265358979323846264338327950288
#define CIRCULO_EN_RADIOS M_PI / 180
#define TIERRA_DIAMETRO_EN_KM 6367.0

// Obtener distancia en kilometros entre coordenadas usando la formula de
Harversine
double obtenerDistanciaEnKM(double latitudOrigen
                                        , double longitudOrigen
                                        , double latitudDestino
                                        , double longitudDestino){

  double latitudOrigenEnRadios = latitudOrigen * CIRCULO_EN_RADIOS;
  double longitudOrigenEnRadios = longitudOrigen * CIRCULO_EN_RADIOS;
  double latitudDestinoEnRadios = latitudDestino * CIRCULO_EN_RADIOS;
  double longitudDestinoEnRadios = longitudDestino * CIRCULO_EN_RADIOS;

  double differenciaLatitud = latitudDestinoEnRadios - latitudOrigenEnRadios;
  double differenciaLongitud = longitudDestinoEnRadios - longitudOrigenEnRadios;

  double differenciaEntrePuntos = pow(sin(differenciaLatitud / 2.0), 2)
                                                          + cos(latitudDestinoEnRadios)
                                                          * cos(latitudOrigenEnRadios)
                                                          * pow(sin(differenciaLongitud
/ 2.0), 2);

  double distanciaAngular = atan2(sqrt(differenciaEntrePuntos), sqrt(1 -
differenciaEntrePuntos));

  return distanciaAngular * TIERRA_DIAMETRO_EN_KM;
}
```

9. Aca hay un ejemplo de una funcion macro la cual se puede applicar con differentes tipos de variables como int, double, float
   Acuerdense que el pre-procesador solo hace reemplazos, asi que funciones macros con muchos codigos de linea es no productivo.

   ```
   #define max(a, b) \
       ({ typeof (a) _a = (a); \
           typeof (b) _b = (b); \
           _a > _b ? _a : _b; })
   ```

   “\” indica al pre-procesador que el código sigue en la próxima linea
   “typeof” hace que el pre-procesador remplace “typeof” por el tipo de la variable usada, por ejemplo: “typeof” pasa a ser “int” si “a” es del tipo int.
   “_a > _b ? _a : _b;” es otra forma de escribir un if statement. Si _a es major que “_b”, elibe a “_a” sino a “_b”

10. C no tiene valores booleanos como TRUE, FALSE, YES o NO.
    Pero eso no quiere decir que no podemos simularlos.
    Para simularlos usaremos el “enum”

```
typedef enum {
  FALSE,
  TRUE
} bool;
```

El valor FALSE sera 0 y el valor TRUE sera 1

11. La mejor forma de mantener valores relacionados juntos es crear una nueva tipo de variable.
    Utilizaremos estructuras para eso:

```
struct estructuraDeCoordenadas2D{
  int x;
  int y;
} structCoordenadas2D;

void main(){
  structCoordenadas2D puntoOrigen2D;
  puntoOrigen2D.x = 12;
  puntoOrigen2D.y = 25;
  mostrar(puntoOrigen2D);
}

void mostrar(structCoordenadas2D punto2D){
  printf("X: %d, Y: %d \n", punto2D.x, punto2D.y);
}
```

12. Esta tabla de caracteres constante pude ser de ayuda:

| | |
|---|---|
| \\ | \ |
| \? | ? |
| \” | “ |
| \’ | ‘ |
| \n | Nueva linea |
| \b | Backspace |
| \f | Limpia la pantalla o saltea a la proxima pagina si se usa en impresoras |
| \a | Usar Alarma del parlante |
| \r | Carriage return |
| \t | Tab Horizontal |
| \v | Tab Vertical |
| \ooo | Numero octal “\o44” |

| | |
|---|---|
| \xhh | Numero hexagecimal “\x0F” o “\0x0F” |