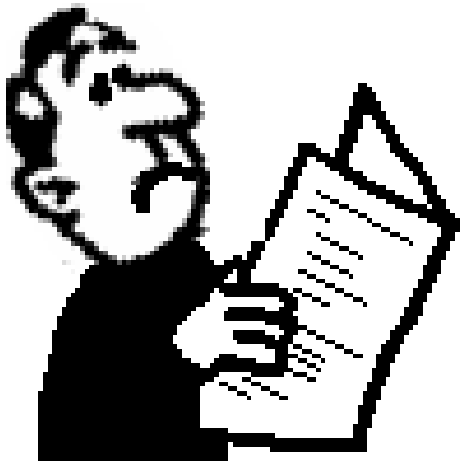
Programando con estilo H



Baltasar García Perez-Schofield
http://jbgarcia.webs.uvigo.es/

Introducción (I): La crisis del Software

- Se emplea más tiempo en mantener un programa que en crearlo por primera vez.
- El mantenimiento implica corrección de errores y ampliación de funcionalidad.
- El mantenimiento implica, por tanto, un profundo estudio del código fuente.
- El software que no se mantiene es aquel que no se usa.

Introducción (II): Manteniendo el software

"Always code as if the guy who ends up maintaining your code your code will be a violent psycopath that knows where you live."

John Woods
(autor de "*The quote executive*")

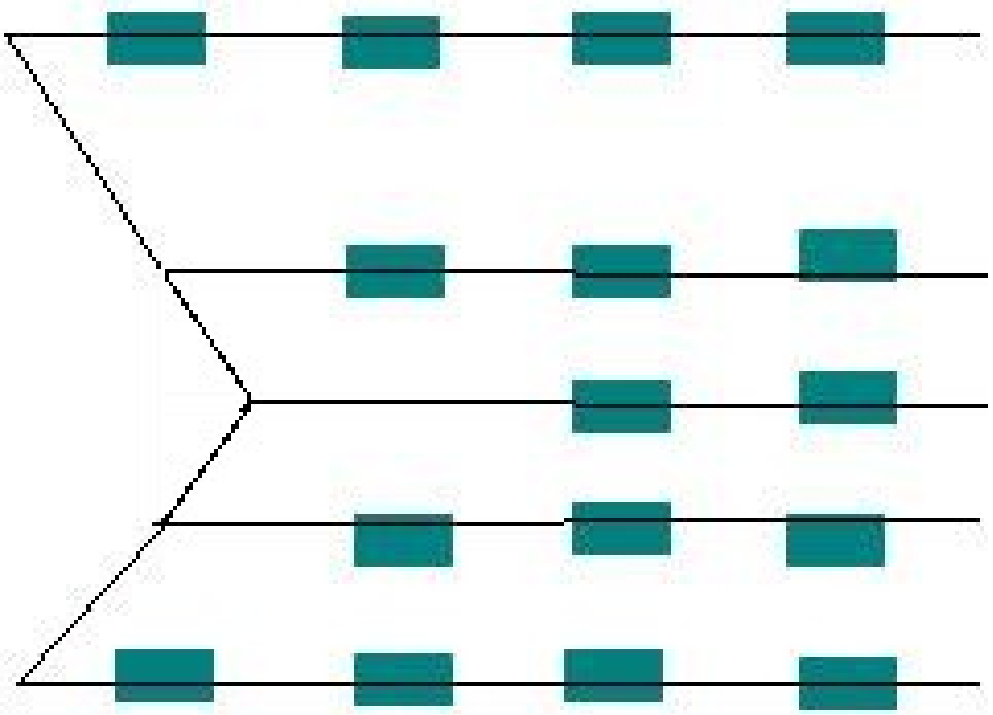
Introducción (II): Manteniendo el software

- Es realmente difícil que un solo programador mantenga una aplicación determinada a lo largo de toda su vida. Y aunque lo haga, si el proyecto es suficientemente grande, necesitará documentación.
- Otros programadores deben ser capaces de leer el código
- Otros programadores deben poder aprender leyendo código.

Introducción (II): Manteniendo el software

Researcher	Result
IBM *Corbi, 1989*	**More than half** of the effort in accomplishing a task for a programmer goes toward understanding the system
Bell Labs *Davison, 1992*	New project members spend **60%-80%** of their time understanding code. This number drops to 20% as the developers gain experience
National Research Council in Canada *Singer et al, 2006*	Developers spending **over 25%** of their time either searching for or looking at code
Microsoft *D. LaToza, 2007*	**Equal amount** of time is spent understanding code as is spent on other tasks such as designing, unit testing, and writing
Microsoft *Peter Hallam, 2006*	**More than 70%** of a developer's time is spent understanding code, as it is a preliminary requisite
Microsoft *Cherubini, 2007*	**95%** of developers agreed that understanding existing code is a significant part of their job. **Over 65%** indicated that they spend time understanding existing code at least once a day (with **over 25%** indicating that they do it multiple times a day)

Agrupación básica recursiva de cualquier programa



Reglas generales de estilo en el código fuente

Normas generales de estilo en el código

- Nombres de identificadores
- Comentarios
- Disposición de la secuencialidad del programa
- Expresiones
- Disposición de los elementos de control (apertura-cierre de funciones, apertura-cierre de bloques)
- Disposición de los controles de flujo del programa (if, while).

Nombres de identificadores

- Tan cortos como sea posible, pero ... tan informativos como sea posible.
- Técnica PascalCase (C#, Pascal):
  - *NombreDeIdentificadorLargo*
- Técnica camelCase (Camel, Java):
  - *nombreDeIdentificadorLargo*
- Técnica guiones a go-gó (C++, Python):
  - *nombre_de_identificador_largo*

Nombres de identificadores

- Por ejemplo:
- x, n, i -> bucles o argumentos de funciones simples.
- numCaracteres
  - numeroCaracteres?
  - numCars?
- nombreUsuario
  - nomDeUsuario?
  - nomUsr?

Comentarios

- Un comentario:
  - Debe introducirse sólo si es útil
  - Debe explicar, no complicar
  - No debe insultar la inteligencia del lector

```
int areaRectangulo = lado1 * lado2;    // cálculo del area
x = Math.PI * r * r;          // cálculo del área del círculo
                              // PI es 3.1415927
```

Comentarios

- Un comentario:
  - Debe introducirse sólo si es útil
  - Debe explicar, no complicar
  - No debe insultar la inteligencia del lector
  - ¿Código autodocumentado? *“There are 2 hard problems in computer science: cache invalidation, naming things, and off-by-1 errors.”* Martin Fowler.

```
function twoPlaces( num ) {
    //Walker E. Pluribus Unum Richardson
    //suuuuuuuuuupppperrrrr geeeeeeeeennniuuuuusssss
    if ( num > 10 ) num = '0' + num;
    return num;
}
```

Comentarios

- Dos tipos básicos de comentarios:
  - "Encima"
    - Los más recomendables (explican un bloque)
  - "A la derecha"
    - Siempre cortos
    - Cuidado con los márgenes (no pasar de ¿80?).

```
// Cálculos previos al rendering
areaRectangulo = lado1 * lado2          // en cms
```

Secuencialidad

- Dispónganse las instrucciones de un programa en párrafos
- Cada párrafo puede llevar un comentario "arriba"
- Nunca una función debe tener más texto que una página impresa, a no ser que se trate de acciones repetitivas.
- Trátese de seguir el esquema: inicialización, proceso (normalmente, un bucle), finalización
- Cortar las líneas antes de la columna 80

Secuencialidad del programa

- Estructura básica:

```
public bool Contains(T x) {
    bool toret = false;

    foreach(T v in list) {
        if ( v == x ) {
            toret = true;
            break;
        }
    }

    return it;
}
```

Secuencialidad del programa

```
// Colocar la ventana
this.Left   = 0;
this.Height = 0;
this.ShowAll();

// Activar los botones
this.BtOk.Enabled = true;
this.BtCancel.Enabled = true;
```

Secuencialidad del programa

```
int    i;
int    x;
int    y;
string edad;
double area;
int a, b, c;          // NO
```

Expresiones Matemáticas

- c = a * b + c
  - ... era (a * b) + c
  - ... era a * (b + c)
- *cars = lineas * caracteresPorLinea + espaciosMargen*
  - ... ¡era a *(b + c)!
- Sin embargo, el código ejecutable generado es exactamente el mismo con paréntesis que sin ellos:
  - ((a * b) + c)
  - a * b + c

Expresiones Matemáticas

- ¿Donde cortar una línea con una expresión larga?
  - Antes de una subexpresión
  - Antes de un operador
  - Antes de un paréntesis

```
int x = ( a * b + c ) / ( c * d * d )
        + ( a / ( b * c ) )
        + ( ( Math.PI * b ) + d )
;
```

Otras expresiones

- Por ejemplo:

```
while( *ptrDestino++ = *ptrOrigen++ );
```

```
return x % 2 == 0 ? x / 2 : 3 * x + 1;
```

Otras expresiones(II)

- Añadiendo unos paréntesis:

```
while(*( ptrDestino++ ) = *( ptrOrigen++ ));
```

```
return ( x % 2 == 0 ) ?
            ( x / 2 )
          : ( 3 * x + 1 );
```

Otras expresiones(III)

- Versión real legible:

```
while( *ptrOrigen != 0 ) {
  *ptrDestino = *ptrOrigen;
  ++ptrOrigen;
  ++ptrDestino;
}
```

Otras expresiones(IV)

Versión real legible:

```
if ( x % 2 == 0 ) {
    return x / 2;
} else {
    return ( x * 3 ) + 1
}
```

Bloques

- La forma más recomendable de colocar los bloques es marcar el inicio y el fin de un bloque en líneas separadas para métodos, empezando en la misma línea para el resto.

```
if ( divisor != 0 ) {
  resultado = dividendo / divisor;
  Console.WriteLine( resultado );
}
```

Bloques de una sola línea

- La línea tiende a ser ilegible. Dificulta el mantenimiento.
- Recuérdese que los bloques de una sola línea sin marcas de bloque son una posibilidad, no una obligación.
- Lenguajes de programación recientes como **Google Go** ya no los permiten. No se deben usar nunca.

```
if ( divisor != 0 ) {
  Console.Write( dividendo/divisor );
}
```

Estructuras de flujo y repetición

- Disposición de las condiciones en un *if()* o en un *while()*.
  - Una *sub*condición por línea, comenzando por el juntor lógico. Si es necesario, una condición puede llevar un comentario "a la derecha".
  - Si existen varias subexpresiones condicionales, se pueden indentar respecto a la expresión principal.

Estructuras de decisión

```
if  ( !fichEntrada.eof()
 && ( bytesLeidos < 0
    || bytesLeidos > 1000 )
 && caracter != EOF )
{
   caracter = fichEntrada.read();
}
```

Reglas semánticas de escritura de código

¿Qué significa lo que escribes?
¡Dame una pista!

Nombres de variables según uso

- Devolución de un valor:

```
public class Punto {
      // ...
     public override string ToString()
     {
      string toret = "";

      toret += "(" + x + ", " + y + ")";

      return toret;
     }
}
```

Nombres de funciones significativos

- Los nombres de métodos deben sugerir qué hacen:

```
int getEdad(Persona);
bool esPalindromo(const string &);
```

- Evítesen nombres como los siguientes:

```
int procesa(StreamReader reader);
string pasaAuxiliar(string str);
int[] leeYduplica(StreamReader reader);
```

Ejemplos reales

El operador ?

- Función, en el tres en raya, para el valor estático del estado actual (?):

```
// Calcula el valor estático del estado actual
private int valorEstado()
{
   return HayGanador( COMPUTADORA ) ? COMPUTADORA_GANA :
          HayGanador( HUMANO )       ? HUMANO_GANA :
          TableroLLeno()             ? EMPATE : INCIERTO;
}
```

El operador ?

```
// Devuelve el estado del juego: vencedor, empate o incierto
private EstadoJuego GetEstado(){
    int toret = EstadoJuego.INCIERTO;

    if ( esGanador( HUMANO ) ) {
        toret = EstadoJuego.HUMANO_GANA;
    }
    else
    if ( esGanador( COMPUTADORA ) ) {
        toret = EstadoJuego.COMPUTADORA_GANA;
    }
    else
    if ( estaTableroLleno() ) {
       toret = EstadoJuego.EMPATE;
    }
    return toret;
}
```

El operador ?

```
// Devuelve el estado del juego: vencedor, empate o incierto
private EstadoJuego getEstado()
{
    if ( esGanador( HUMANO ) ) {
        return EstadoJuego.HUMANO_GANA;
    }
    else
    if ( esGanador( COMPUTADORA ) ) {
        return COMPUTADORA_GANA;
    }
    else
    if ( estaTableroLleno() ) {
        return EMPATE;
    } else {
        return INCIERTO;
    }
}
```

Documentación compulsiva

```
// Utiliza el constructor de copia
Inventory copy = family_videos;
// Utiliza el operador de asignación
// sobrecargado
copy2 = copy;
// Muestra los datos de los videos del
// inventario
family_videos.print_video_info();
// Utiliza la copia para mostrar los datos
// de los videos del inventario
copy.print_video_info();
// Utiliza otra copia para mostrar los datos
// de los videos del inventario
copy2.print_video_info();
```

Documentación compulsiva

```
// Se define la clase complejo
public class Complejo extends Object {
    // La clase tiene las dos variables miembro siguientes
    double preal;
    double pimag;

    // Se define el constructor de la clase
    public Complejo(double partereal, double parteimag) {
        // ...
    }

    // Se define el método para calcular el complejo opuesto
    public void opuesto() {
        // ...
    }
    // ...
}
```

Secuenciación incorrecta

```
public void init() {
   Complejo c1,c2,c3;
   c1=new Complejo(1.0,2.0);
   c2=new Complejo(3.0,4.0);
   c3=new Complejo(0.0,0.0);
   c1.imprime();c2.imprime();
   c3=c2;
   c3.opuesto();c3.imprime();
   c3.conjugado();c3.imprime();
   // ...
}
```

Secuenciación incorrecta

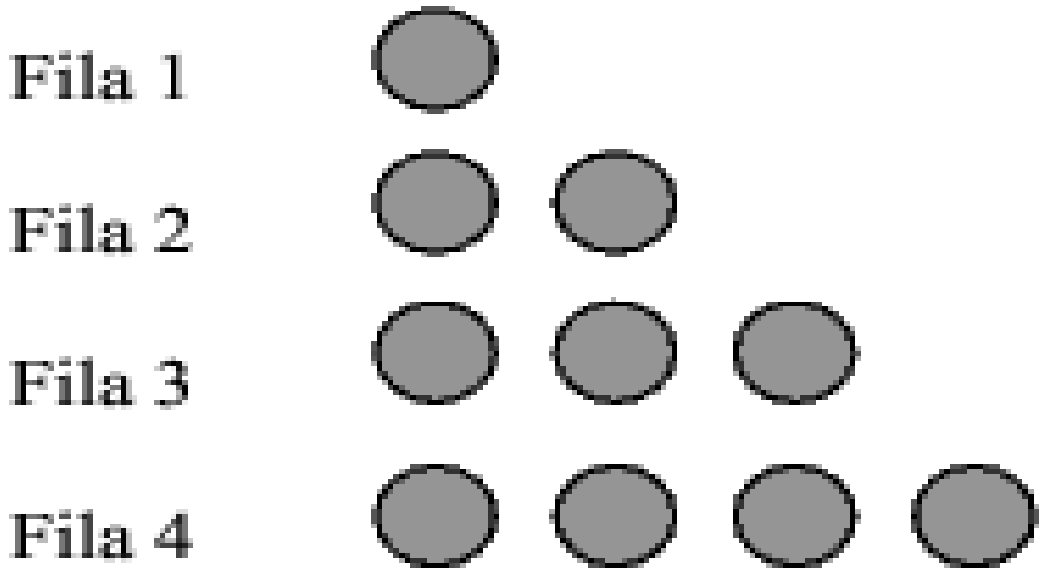
```
public void init() {
    Complejo c1;
    Complejo c2;
    Complejo c3;

    c1 = new Complejo( 1.0, 2.0 );
    c2 = new Complejo( 3.0, 4.0 );
    c3 = new Complejo( 0.0, 0.0 );

    c1.imprime();
    c2.imprime();

    c3 = c2;
    c3.opuesto();
    c3.imprime();
}
```

El juego del Nim



El juego del Nim

- La estrategia del juego se basa en dos métodos, en un programa en Java:

```
class Nim {
    bool movimiento_chachi(Tablero t) {
        // ...
    }
    bool movimiento_chungo(Tablero t) {
        // ...
    }
}
```

El juego del Nim

- "*movimiento_chachi*()" prueba si un movimiento le lleva a ganar el juego:

```
bool movimiento_chachi(Tablero t) {
    // Proponer un movimiento
    // ...
    if (movimiento_chungo(t)) {
        // Realizar el movimiento
    }
    // ...
}
```

El juego del Nim

- "*movimiento_chungo()*" comprueba si un movimiento es malo ... ¿no?

```
bool movimiento_chungo(Tablero t) {
    if ( t.esVacio()
      || movimiento_chachi( t ) )
    {
        return false;
    }
    else return true;
}
```

El juego del Nim

- "*movimiento_chachi*()" puede ser modificado, eliminando "*movimiento_chungo()*":

```
bool hayMovimientoGanador(Tablero t) {
    // Proponer un movimiento
    // ...
    if (!( t.esVacio() )
     && !( hayMovimientoGanador( t ) ) ) {
        // Realizar el movimiento
    }
    // ...
}
```

Programas “autodocumentados”

Concurso Internacional "C ofuscado"
http://www.es.ioccc.org/

```
#define processor x86
#include <stdio.h>
#include <sys/stat.h>
#define l int*
#define F char
 struct stat t;
#define c return
#define I (P+=4,*L(P-4,0))
#define G (signed F)E(*L(P++,0))
#define C(O,D)E (D[B+V(010)/4+O*10])
#define U R[4]=E(V(17)-4),*(l)V(021)=
F       M [99],Q[99],b[9999],*ss,*d=b,*z;
#define O =(n=*(l)V(021),R[4]=E(V(17)+4),n)
#define p(a,b,c) system((sprintf(a,b,k[1]),c)),z
#define                             g (y/010&7)
#define                             R (B+13)
#define           x86               (F*)index\
(ss+V(i                             ),0100)
#define      D(y,n,a,m,i,c          )d+=sprintf( d,y,n,a,m,i,c ),(F*\
      )                             P
      l       B,i,n,a,r,y           ,
      P                             ;
#define     Tr(an,sl,at,or)         l an##i(d,sl){ c at? an##i(d,r):or; }
   \
l an(d,                             sl){ c \
r=V(014                             )&63,an##i(d,sl); }
#define add(Ev,Gv) + ...
main (char *ck, char **k) {  exit(E((ck?main((z?(stat(M,&t)?P+=a+'{'?
   0:3:execv(M,k),a=G,i=P,y=G&255,sprintf(Q,y/'@' ...

}
```

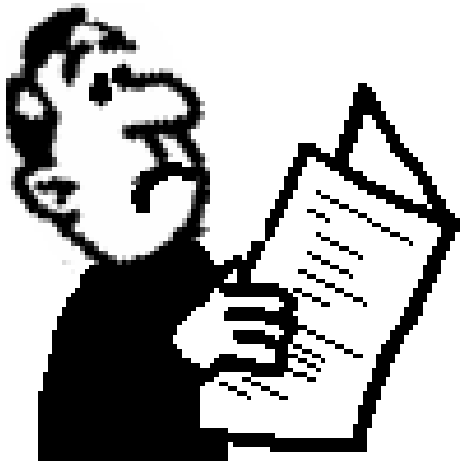
```
#include <stdio.h>
int l;int main(int o,char **O,
int I){char c,*D=O[1];if(o>0){
for(l=0;D[l                  ];D[l
++]-=10){D   [l++]-=120;D[l]-=
110;while   (!main(0,O,l))D[l]
+=   20;   putchar((D[l]+1032)
/20   )   ;}putchar(10);}else{
c=o+     (D[I]+82)%10-(I>l/2)*
(D[I-l+I]+72)/10-9;D[I]+=I<0?0
:!(o=main(c/10,O,I-1))*((c+999
)%10-(D[I]+92)%10);}return o;}
```

```
                                    #ifdef s
                                                          z
                                                         r(
                                                        ){z
                                                       k=0,l
                                                      =0,n,x
                                                  XQueryPointer(i
                                            ,XRootWindow      (i,j),&m,
                                  &m,&o,&p,&n,&n,(               ghj)&n),(o
                                                        >=s(g)||s(o
                                                       )<=0)&&(k=1),
                                                      (p>=h||p<=0)&&
                                                     (l=1),(e==1)&&(
                                                    c=o,d=p,e=0,1)||(
                                                    (k==0&&o-c-(z)(a+y
                                                   (a)*.5)!=0)&&(a=o-c
                                                  ),(l^-1==-1&&p-d-(z)(
                                          b+y(b)*.5)!=0)&&(b=p-d),a/=f,b/=f
                                   ,k=0,l=0);(o      >=s(g)||o<=0)&&(a=-a),(
                          p>=h||s(p)<=0)                &&(b=-b),c=o,d=p,I(XWarpP
                                                  ,ointer)(i,None,None,0,0,s
                                                 (g),h,(z)(a+y(a)*.5),(int)(
                                                b+y(b)*.5 JJ(float B;int)C,D;
                                               #else/*Egads! something has */
                                              #include<X11/Xlib.h>/*taken a*/
                                             #include<stdio.h>/*huge bite o-*/
                                            #include<stdlib.h>/*ut of the m-*/
                                           #include<time.h>/*ouse pointer!!!*/
                                       #define H(a, b) (((a)&(7<<3*(b)))>>3*(b))
                             #define G(c,d)  ((H(c,d)<<3*(d+1))|((H(c,d+1)<<3*d)|/*
                      _XSetPointer(display,     screen,GREASY|BOUNCY)*/c&~(63<<3*(d))))
#define                             s(e) (G(G(G(G(G(G(e,(z)0),1),2),1),0),1))
                                typedef int z;float a=0,b=0,c,d,f=1.03;z e
                                                  =s(512),g,h,j;
                                                  Display/**/*i;
                                             #define y(X)((X>0)-(X<0))
                                       #define x        o,p; Window m;
                                 #define ghj              unsigned  int*
                             #define                       I(aa,bb)aa##bb
                                                        #define JJ(X)\
                                                        ));return 0;}X
                                                       z r();int main
                                                       (z X,char**Y){
                                                      clock_t q=0;(X
                                                   ==2)&&(f=atof(Y[1])),((i
                                          =XOpenDisplay(0)      )==0)&&(exit(1
                                   ),1),j=I(Defa,                  ultScreen)(i),
                                                     g=s(I(Display,
                                                     Width)(i,j)-1)
                                                    ,h=I(DisplayH,
                                                    eight)(i,j)-1;
                                                   for(;;((I(clo,
                                                   ck)()-q)*100>(
                                                  CLOCKS_PER_SEC
                                                ))&&(r(),q=clock()));}
                                               #include __FILE__
                                                   #endif
```

Bibliografía

- Eckel (2000). *Thinking in C++*. Prentice Hall
- Caro, Ramos, Barceló (2002). *Introducción a la programación con orientación a objetos.* Prentice-Hall
- Steve McConell. *Code Complete*. Microsoft Press.
- Documentación de Microsoft: https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/inside-a-program/coding-conventions
- Documentación de Sun: https://www.oracle.com/technetwork/java/codeconvtoc-136057.html

Programando con estilo H



Baltasar García Perez-Schofield
http://jbgarcia.webs.uvigo.es/