Coloquios Abiertos:

Legibilidad del código fuente

¿Puedo hacer mis programas más fáciles de leer?

Jueves 5 de Junio
Edif. Politécnico
Salón de actos, 18h

J. Baltasar García Perez-Schofield
http://trevinca.ei.uvigo.es/~ca/
http://trevinca.ei.uvigo.es/~jgarcia/ca/

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 (I): Manteniendo el software

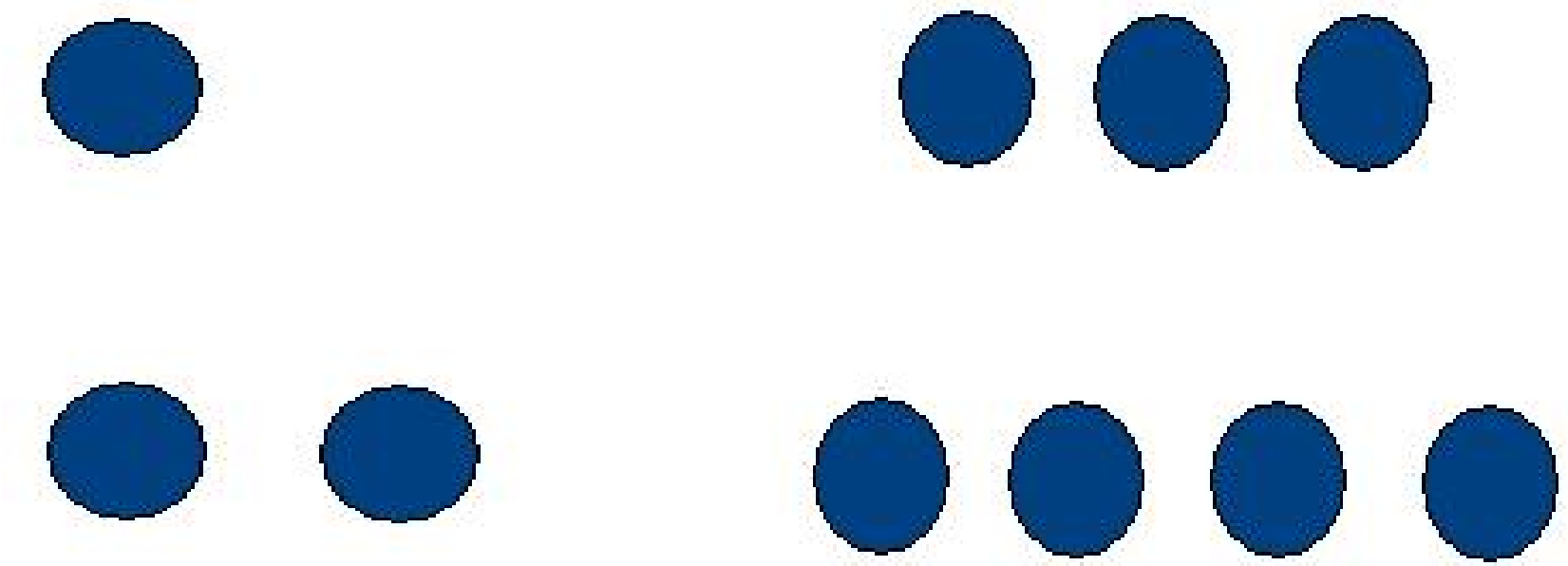
- Es realmente difícil que un solo programador mantenga una aplicación determinada a lo largo de toda su vida.
- Otros programadores deben ser capaces de leer el código
- Otros programadores deben poder aprender leyendo código.

Introducción (II): Mecánica de la lectura

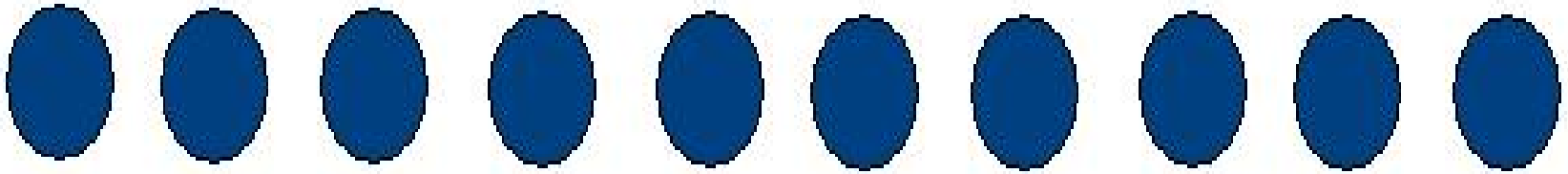
- Al leer, NO se lee cada letra individual y se forma la palabra al final.
- Se reconoce la forma visual de cada palabra, y se aplican técnicas de frecuencia de aparición.
- Esto explica por qué existen fuentes “difíciles de leer“

Reconocimiento de formas y agrupaciones

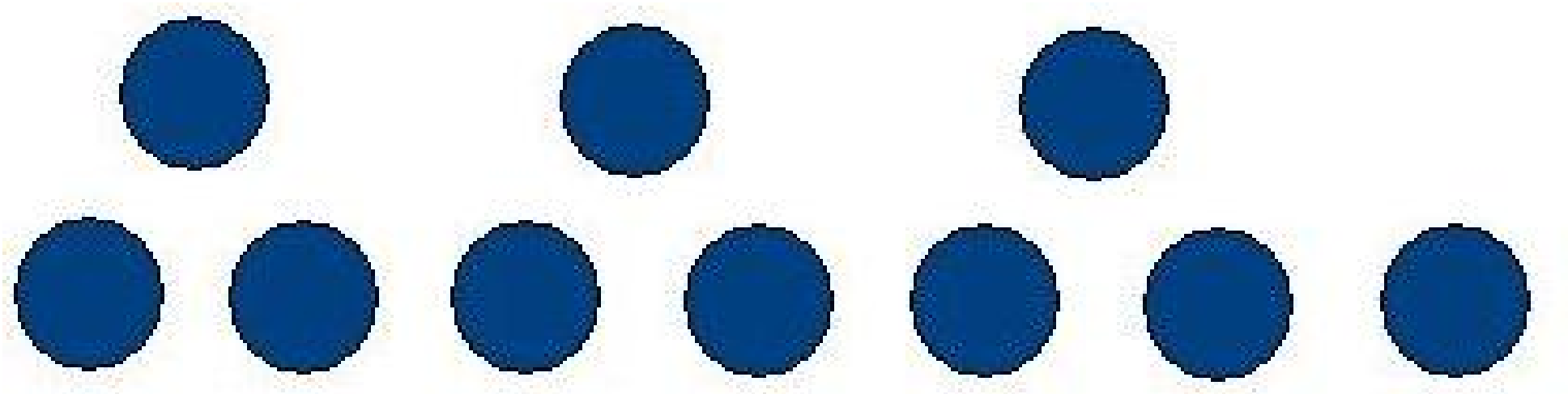
- En los jeroglíficos antiguos, las ideas numéricas se expresaban agrupando una forma básica que representaba la unidad.



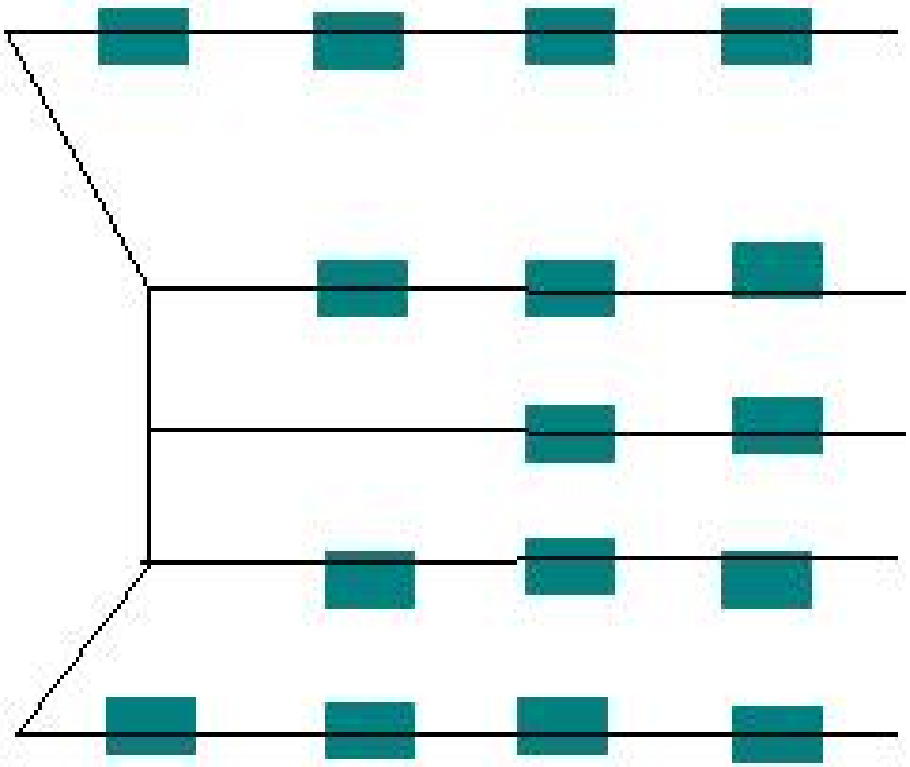
Reconocimiento de formas y agrupaciones



Reconocimiento de formas y agrupaciones



Agrupación básica recursiva de cualquier programa



Códigos de colores

- No son tan importantes como el agrupamiento y las formas, pero pueden ser un buen complemento.
- Colocar los identificadores de un color, los elementos de control (comienzo-final de bloque, comienzo-final de métodos/funciones, comienzo-final de clases) de otro, los comentarios de otro diferente ... etc, ayuda a reconocer partes en un programa.

Códigos de colores

```
Sub USR_Abrir(nombre)
    Declare(ret)

    ret:=FALSE

    // NOTAS:
    // este procedimiento se ejecuta cada vez que se intenta ABRIR
    // un objeto, podemos obviar la acción estándar de ABRIR si se
    // devuelve TRUE, por ejemplo:
    //
    // If nombre="PUERTA" Then
    //     If PSI[PSIJugador].Localidad:="HABITACION" Then
    //         If LOC["HABITACION"].Abierta("NORTE") Then
    //             Print("La puerta ya está abierta." + CR)
    //         Else
    //             Print("Abres la puerta." + CR)
    //             LOC["HABITACION"].Abrir("NORTE")
    //         EndIf
    //         ret:=TRUE
    //     EndIf
    // EndIf

Return ret
```

Códigos de colores

```
Sub USR_Abrir(nombre)
    Declare(ret)

    ret:=FALSE

    // NOTAS:
    // este procedimiento se ejecuta cada vez que se intenta ABRIR
    // un objeto, podemos obviar la acción estándar de ABRIR si se
    // devuelve TRUE, por ejemplo:
    //
    // If nombre="PUERTA" Then
    //     If PSI[PSIJugador].Localidad:="HABITACION" Then
    //         If LOC["HABITACION"].Abierta("NORTE") Then
    //             Print("La puerta ya está abierta." + CR)
    //         Else
    //             Print("Abres la puerta." + CR)
    //             LOC["HABITACION"].Abrir("NORTE")
    //         EndIf
    //         ret:=TRUE
    //     EndIf
    // EndIf

Return ret
```

Normas 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.
- La nueva técnica de moda (Java):
  - nombreDeIdentificadorLargo
- Otras técnicas:
  - nombre_de_identificador_largo
- Incluso (si está permitido):
  - 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 := PI * r * r; // cálculo del área del círculo
                 // PI es 3.14159927
```

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 }
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
- Trátese de seguir el esquema: inicialización, proceso, finalización
- Cortar las líneas antes de la columna 80

Secuencialidad del programa

```
iterator it = l->begin();

while(it != NULL)
{
   if (*it == x)
      break;
it = it->next();
}

return it;
```

Secuencialidad del programa

```
// Colocar la ventana
Left   = 0;
Height = 0;
Show();

// Activar los botones
Boton1->Enabled = true;
Boton2->Enabled = true;
```

Secuencialidad del programa: declaración de variables

```
Var
i         : integer;
area      : real;
longitud: real;
nombre    : string;
edad      : integer;
```

```
int    i;
int    x;
int    y;
string edad;
double area;
int x, *ptr; // NO
```

Expresiones Matemáticas

- a * b + c
  - ... era (a * b) + c
  - ... era a * (b + c)
- Lineas * caracteresPorLinea + espaciosMargen
  - ... ¡era a *(b + c)!
- Sin embargo, el código ejecutable generado es el mismo con paréntesis:
- ((a * b) + c)
- a * b + c

Expresiones Matemáticas (II)

- ¿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))
+ ((3.1451927 * b) + d);
```

Otras expresiones

- Por ejemplo:

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

```
while(it != NULL)
   it = it->sig == NULL? NULL :
          it->sig->iterador;
```

Otras expresiones(II)

- Añadiendo unos paréntesis:

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

```
while(it != NULL)
   it = ((it->sig == NULL)? NULL :
         (it->sig->iterador));
```

Otras expresiones(III)

- Versión real legible:

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

Otras expresiones(IV)

- Versión real legible:

```
while(it != NULL)
{
  if (it->sig != NULL)
        it = it->sig->iterador;
  else it = NULL;
}
```

Bloques

- La forma más recomendable de colocar los bloques es marcar el inicio y el fin de un bloque en líneas separadas

```
if (divisor <> 0) then
begin
   resultado := dividendo / divisor;
   writeln(resultado);
end;
```

Bloques de una sola línea

- Son casi siempre poco recomendables porque introducen confusión. La línea tiende a ser ilegible.
- Recuérdese que los bloques de una sola línea sin marcas de bloque son una posibilidad, no una obligación.

```
if (divisor <> 0) then
     write(resultado);
```

Estructuras de flujo y repetición

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

Estructuras de flujo y repetición

```
if  (not(eof(fichEntrada))
 and (bytesLeidos < 0
   or bytesLeidos > 1000)
 and caracter <> FINAL)
begin
   read(fichEntrada, caracter);
end;
```

Estructuras de flujo y repetición y bloques de una sola línea

- Deben evitarse siempre:

```
if  (not(eof(fichEntrada))
 and (bytesLeidos < 0
   or bytesLeidos > 1000)
 and caracter <> FINAL) then
 read(fichEntrada, caracter); {ilegible}
```

Normas de estilo de Sun para Java

Estilo de nombres de identificadores
Comentarios del código fuente

Comentarios formales

- javadoc genera la documentación de una clase automáticamente.
- Entiende “funciones“ embebidas en los comentarios:
  - @return
  - @param
  - @see

Comentarios javadoc

- Por ejemplo:

```
/* getSqr(x)
* @return el cuadrado de x
* @param x el número a elevar al
  cuadrado
*
*/
```

Normas de estilo de Microsoft para C/C++

Notación húngara
Formato de comentarios

Normas de estilo de Microsoft. Notación Húngara

b	Booleano
c	Carácter (un byte)
dw	Entero largo de 32 bits sin signo (DOBLE WORD)
f	Flags empaquetados en un entero de 16 bits
h	Manipulador de 16 bits (HANDLE)
l	Entero largo de 32 bits
lp	Puntero a entero largo de 32 bits
lpfn	Puntero largo a una función que devuelve un entero
lpsz	Puntero largo a una cadena terminada con cero
n	Entero de 16 bits
p	Puntero a entero de 16 bits
pt	Coordenadas (x, y) empaquetadas en un entero de 32 bits
rgb	Valor de color RGB empaquetado en un entero de 32 bits
sz	Cadena terminada en cero
w	Entero corto de 16 bits sin signo (WORD)

Ejemplos de identificadores

nContador
lpszNombreAplicacion
CMenuItem

Comentarios formales en el código

Class Declaration Header

```
/*C+C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C+++C
  Class:    CMyClass

  Summary:  Short summary of purpose and content of CMyClass.
            Short summary of purpose and content of CMyClass.

  Methods:  MyMethodOne
              Short description of MyMethodOne.
            MyMethodTwo
              Short description of MyMethodTwo.
            CMyClass
              Constructor.
            ~CMyClass
              Destructor.
C---C---C---C---C---C---C---C---C---C---C---C---C---C---C---C---C---C---C---C---C-C*/
```

Class Method Definition Header

```
/*M+M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M+++M
  Method:   CMyClass::MyMethodOne

  Summary:  Short summary of purpose and content of MyMethodOne.
            Short summary of purpose and content of MyMethodOne.

  Args:     MYTYPE MyArgOne
              Short description of argument MyArgOne.
            MYTYPE MyArgTwo
              Short description of argument MyArgTwo.

  Modifies: [list of member data variables modified by this method].

  Returns:  MYRETURNTYPE
              Short description of meaning of the return type values.
M---M---M---M---M---M---M---M---M---M---M---M---M---M---M---M---M---M---M---M---M-M*/
```

Programas autodocumentados

Concurso Internacional “C ofuscado“
http://www.es.ioccc.org/main.html

```
#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
- Documentación de Microsoft: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/stg/stg/coding_style_conventions.asp
- Documentación de Sun: http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

Coloquios Abiertos:

Legibilidad del código fuente

¿Puedo hacer mis programas más fáciles de leer?

Jueves 5 de Junio
Edif. Politécnico
Salón de actos, 18h

J. Baltasar García Perez-Schofield
http://trevinca.ei.uvigo.es/~ca/
http://trevinca.ei.uvigo.es/~jgarcia/ca/