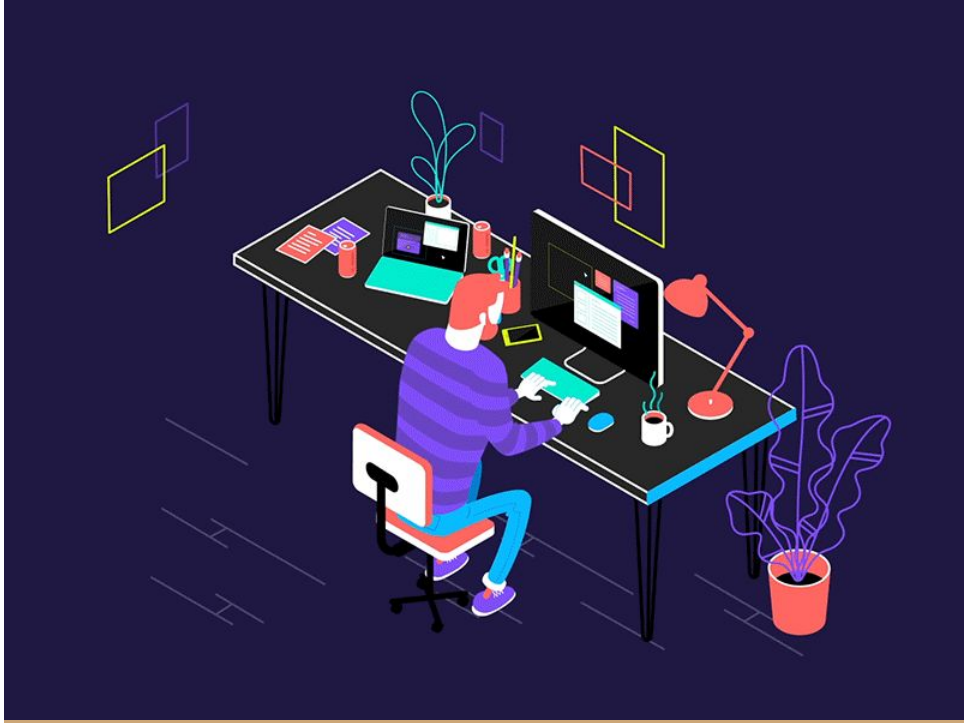

I design and build complex software systems
I create websites and applications
I write text on a computer
I press keys on a keyboard
I force electrons to do math
imgflip.com

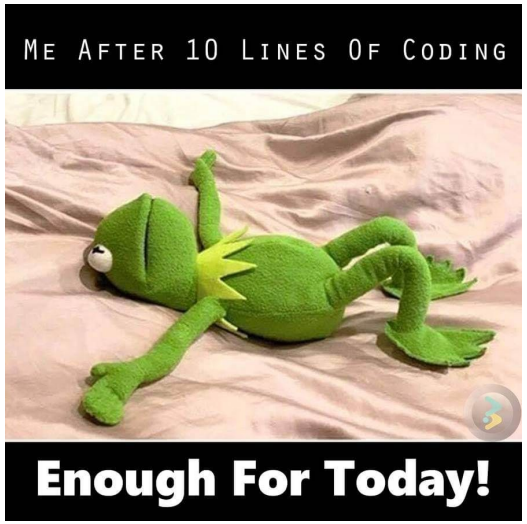


“Simplicity, carried to the extreme, becomes elegance.”

Jon Franklin

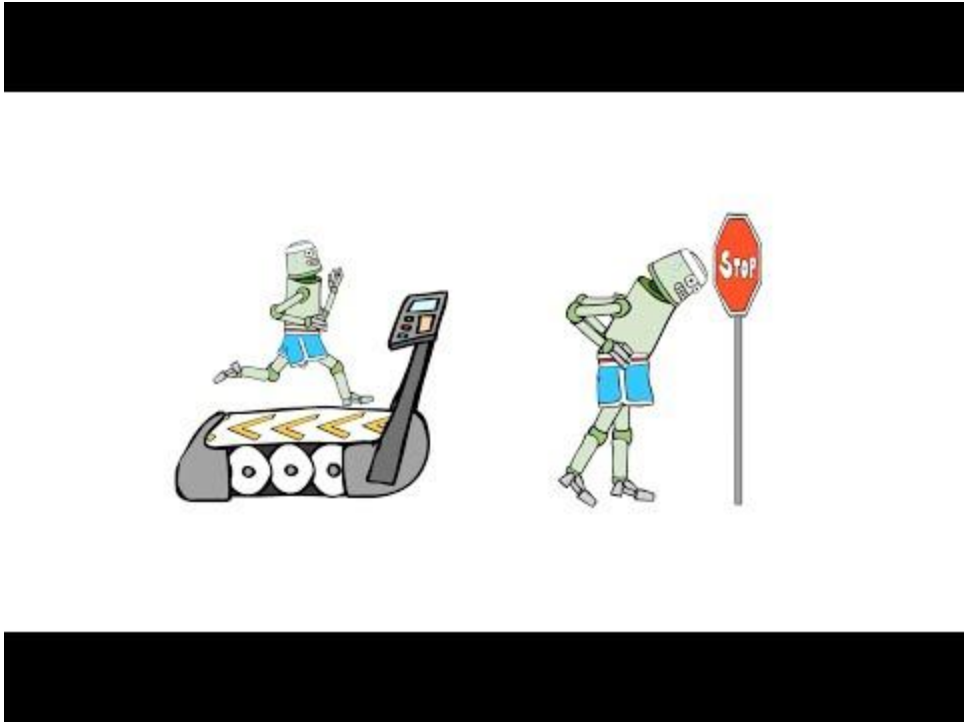
“Make everything as simple as possible, but not simpler.”

Albert Einstein

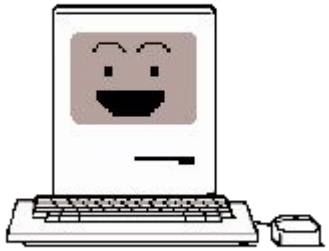


The Art of Writing Software

The Art of Writing Software



Ciclo de desarrollo de software



- ✓ **Planificación**
  Delimitación del ámbito del proyecto, estudio de viabilidad, análisis de riesgos, estimación de costos, planificación temporal y asignación de recursos.

- ✓ **Análisis** (¿qué?): Elicitación de requisitos.
  Descripción clara y completa de qué es lo que se pretende, incluyendo la presentación de los resultados que se desean obtener (formato de las salidas) y la forma en que se va a utilizar la aplicación (interfaz de usuario)

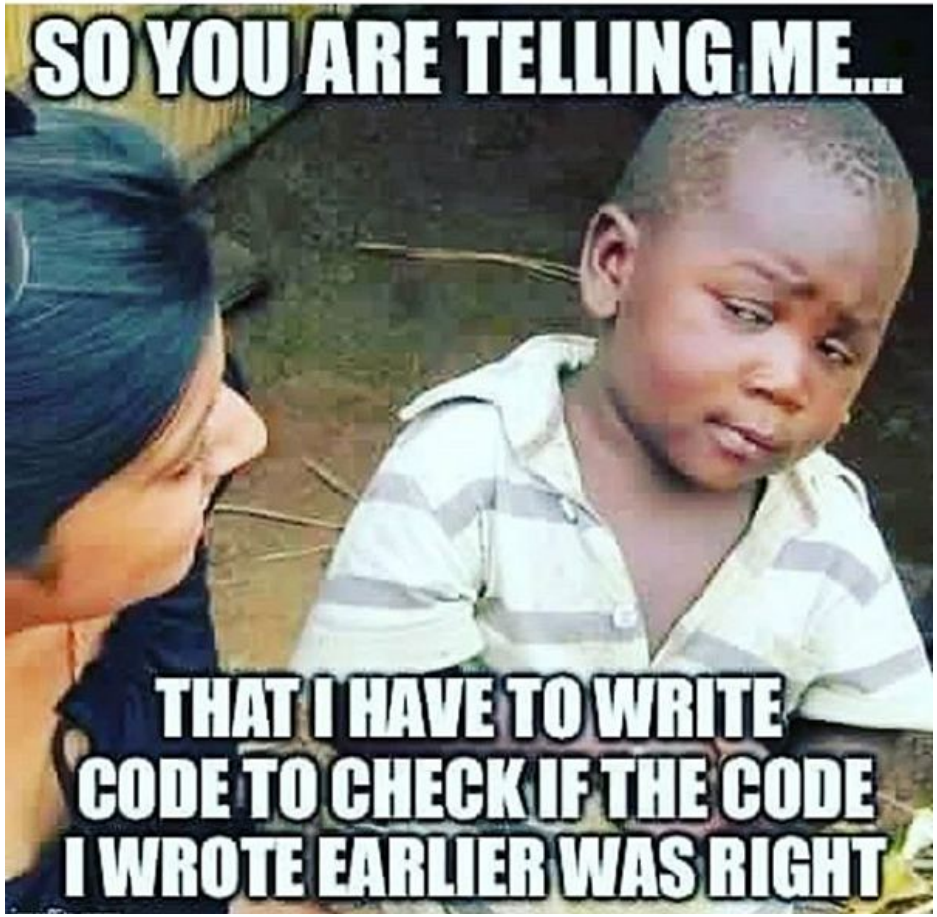
- ✓ **Diseño** (¿cómo?): Estudio de alternativas
  - *Diseño arquitectónico*: Organización de los distintos módulos que compondrán la aplicación (diseño arquitectónico).
  - *Diseño detallado*: Definición de los algoritmos necesarios para implementar la aplicación en lenguaje natural, mediante diagramas de flujo o en pseudocódigo [lenguaje algorítmico].

- ✓ **Implementación**:

  Adquisición de componentes, creación de los módulos de la aplicación en un lenguaje de programación e integración de los recursos necesarios para que el sistema funcione.

- ✓ **Depuración y pruebas**:
  Comprobación del funcionamiento de la aplicación
  Pruebas de unidad y de integración, pruebas alfa, pruebas beta, test de aceptación.
  - *Verificación* (si se está realizando lo que se pretendía)
  - *Validación* (si se realiza lo correcto).

- ✓ **Explotación**: Uso y mantenimiento
  - *Mantenimiento correctivo*: Corrección de defectos o errores.
  - *Mantenimiento adaptativo*: Adaptación de la aplicación a nuevas circunstancias e inclusión de nuevas prestaciones.


SO YOU ARE TELLING ME...
THAT I HAVE TO WRITE
CODE TO CHECK IF THE CODE
I WROTE EARLIER WAS RIGHT

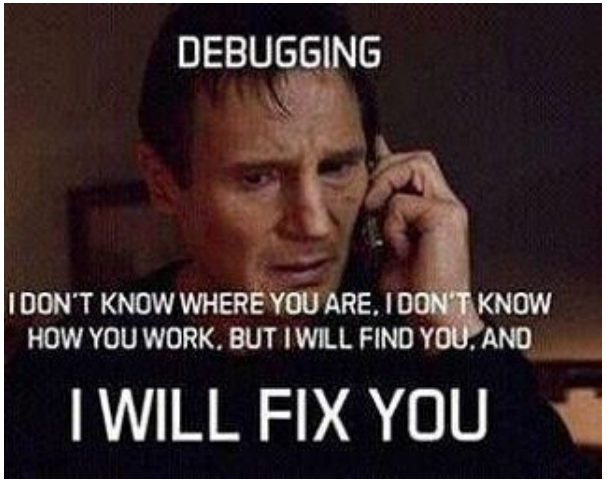
Errores de programación

Errores sintácticos

Errores detectados por el compilador en tiempo de compilación.

Errores semánticos

Sólo se detectan en tiempo de ejecución: Causan que el programa finalice inesperadamente su ejecución (p.ej. división por cero) o que el programa proporcione resultados incorrectos.



Compiler: Error at line 40

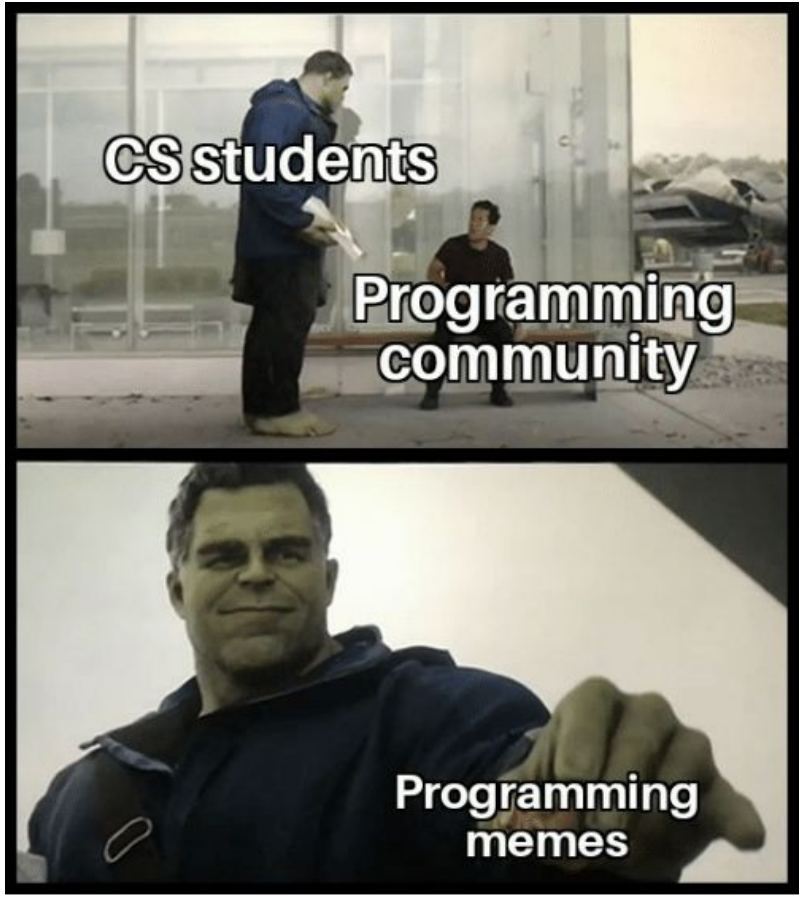
Me: "What? How? My code only has 30 lines

Compiler:

When you write 100 lines of code and it compiles on your first attempt

When your code compiles after 253 failed attempts

Programmers while reviewing the codes



Great contribution indeed.

Comentarios

Los comentarios sirven para incluir aclaraciones en el código.

Java permite dos tipos de comentarios:

```
// Comentarios de una línea

/* Comentarios de varias líneas */
```

- Es bueno incluir comentarios que expliquen lo que hace el programa y sus características claves (p.ej. autor, fecha, algoritmos utilizados, estructuras de datos, peculiaridades…).

```
// Cálculo del MCD
// usando el algoritmo de Euclides
// © Fernando Berzal, 2004
```

- Los comentarios nunca han de limitarse a decir en lenguaje natural lo que ya está escrito en el código: Jamás se utilizarán para "parafrasear" el código y repetir lo que es obvio.

```
× i++;          // Incrementa el contador
```

- Los comentarios han de aclarar; esto es, ayudar al lector en las partes difíciles (y no confundirle). Si es posible, escriba código fácil de entender por sí mismo: cuanto mejor lo haga, menos comentarios necesitará.

✗
```
int mes;   // Mes
```

✓
```
int mes;   // Mes del año (1..12)
```

✗
```
ax = 0x723;   /* RIP L.v.B. */
```

NB: Beethoven murió en 1827 (0x723 en hexadecimal).

BE LIKE

When you get assigned to re-write old programs and there are comments on every line

90% of all code comments:

Sangrías

Conviene utilizar espacios en blanco o separadores para delimitar el ámbito de las estructuras de control de nuestros programas.

Líneas en blanco

Para delimitar claramente los distintos segmentos de código en nuestros programas dejaremos líneas en blanco entre ellos.

Identificadores

Los identificadores deben ser descriptivos (reflejar su significado).

- ✗ `p, i, s...`
- ✓ `precio, izquierda, suma...`

Declaraciones

- Usualmente, declararemos una única variable por línea.
- Nunca mezclaremos en una misma línea la declaración de variables que sean de distintos tipos o que se utilicen en el programa para distintos fines.

Expresiones

- Uso de paréntesis: Aunque las normas de precedencia de los operadores vienen definidas en el lenguaje, no abusaremos de ellas. Siempre resulta más fácil interpretar una expresión si ésta tiene los paréntesis apropiados. Además, éstos eliminan cualquier tipo de ambigüedad.

- Uso de espacios en blanco: Resulta más fácil leer una expresión con espacios que separen los distintos operadores y operandos involucrados en la expresión.

```
a%x*c/b-1    →    ( (a%x) * c ) / b - 1
```

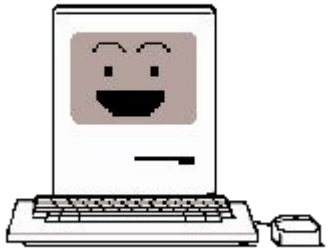
- Expresiones booleanas: Es aconsejable escribirlas como se dirían en voz alta.

```
!(bloque<actual)    →    ( bloque >= actual )
```

- Expresiones complejas:
  Es aconsejable dividirlas para mejorar su legibilidad

El código bien escrito
es más fácil de leer, entender y mantener
(además, seguramente tiene menos errores)

Programación Orientada a Objetos



Conceptos básicos

- Todo es un objeto
- Los objetos se comunican entre sí pasándose mensajes
- Cada objeto tiene un estado
  (contiene su propia memoria [datos])
- Un objeto es un caso particular (instancia) de una clase
- Las clases definen el comportamiento de un conjunto de objetos

Problema

Quiero enviar un paquete a un amigo que vive en otra ciudad

Opciones

a) Hacerlo todo yo mismo
   - Descomposición en subproblemas (programación estructurada)

b) Delegar en alguien para que lo haga (p.ej. Correos)
   - "Realizar un encargo" (programación orientada a objetos)

Solución orientada a objetos

- Se busca un objeto capaz de enviar un paquete
- Se le envía un mensaje con mi solicitud
- El objeto se hace responsable de satisfacer mi solicitud
- El objeto utiliza un algoritmo que yo no tengo por qué conocer

Consecuencias

- Un programa orientado a objetos se estructura como un conjunto de agentes que interactúan (programa como colección de objetos).
- Cada objeto proporciona un servicio que es utilizado por otros objetos (reutilización).
- La acción se inicia por la transmisión de un mensaje al objeto responsable de realizarla.
- Si el receptor acepta el mensaje, acepta la responsabilidad de llevar a cabo la acción solicitada.
- El receptor puede utilizar cualquier técnica que logre el objetivo deseado.


(LAUGHING)
Looking at programming memes
(CRYING)
Actually coding

sumas
leer desde input.
java y python