LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

UNIVERSIDAD NACIONAL DE TUCUMÁN
facet
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

PROGRAMACIÓN

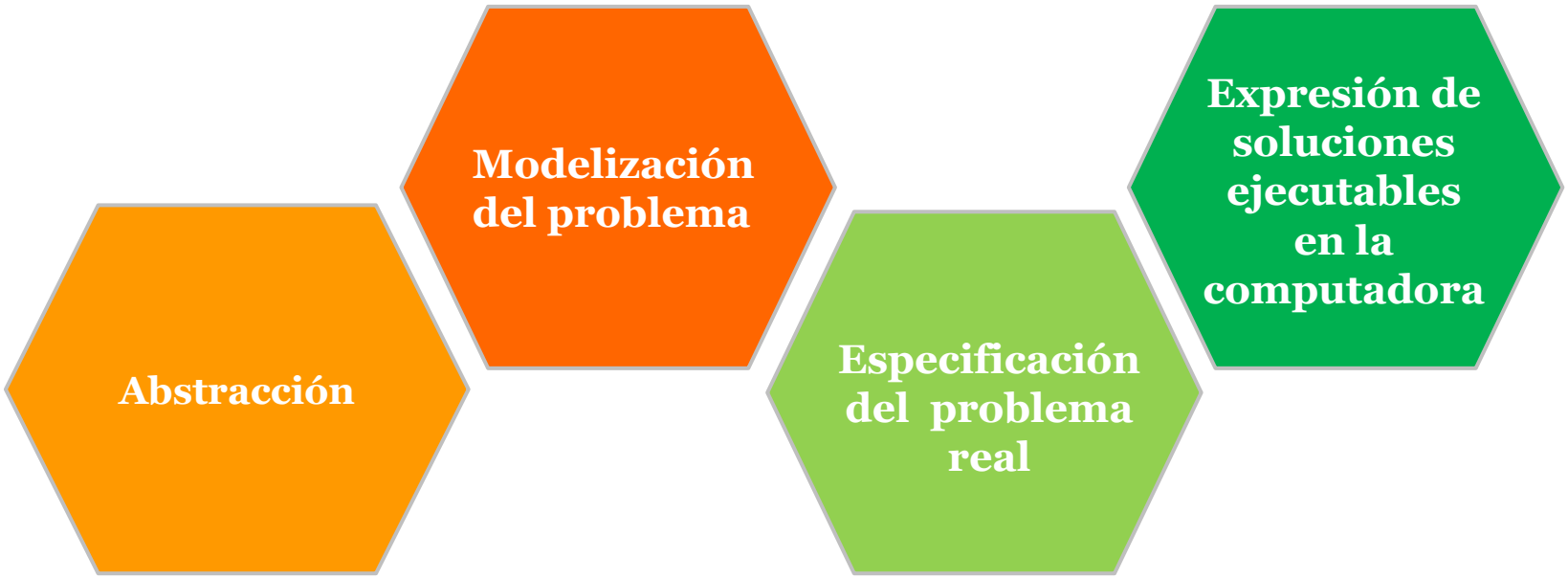
Unidad 1: El proceso de programación: Etapas en la resolución de problemas con computadora. Algoritmos. Formas de reducción de complejidad de problemas del mundo real.

Lic. Mariela A. Velázquez

Algunas preguntas iniciales…

- ¿Cómo se resuelve un problema del mundo real con una computadora?
- ¿ Cómo se expresa la solución al problema planteado?
- ¿Cómo se reduce la complejidad de los problemas ?

El programador debe realizar algunos procesos intelectuales

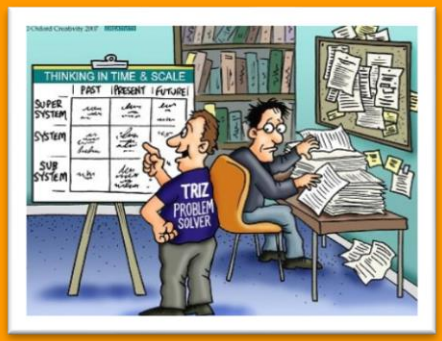


Abstracción

Interpretar los aspectos esenciales de un problema y expresarlo en términos precisos.

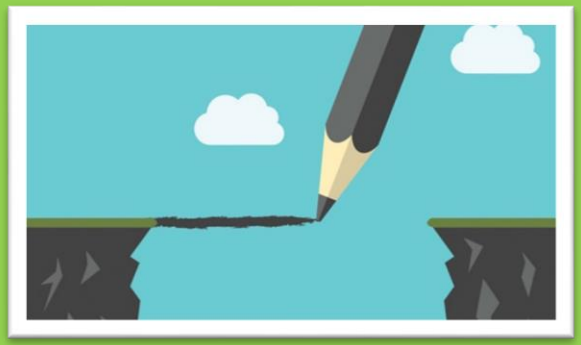
Modelización del problema

Simplificar su expresión, encontrando sus aspectos principales, los datos que se habrán de procesar y el contexto.



Especificación del problema real

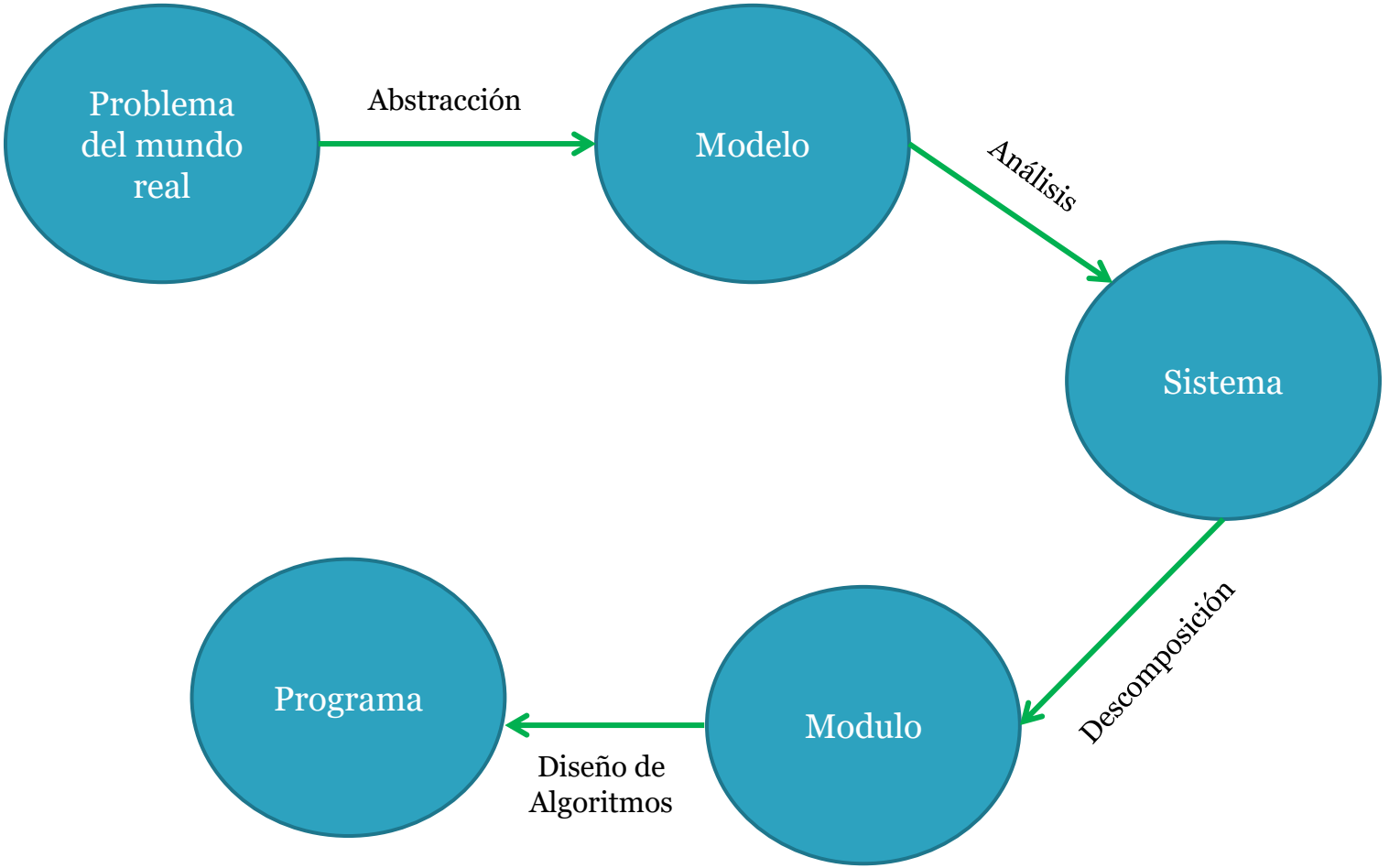
Determinar en forma clara y concreta el objetivo que se desea.



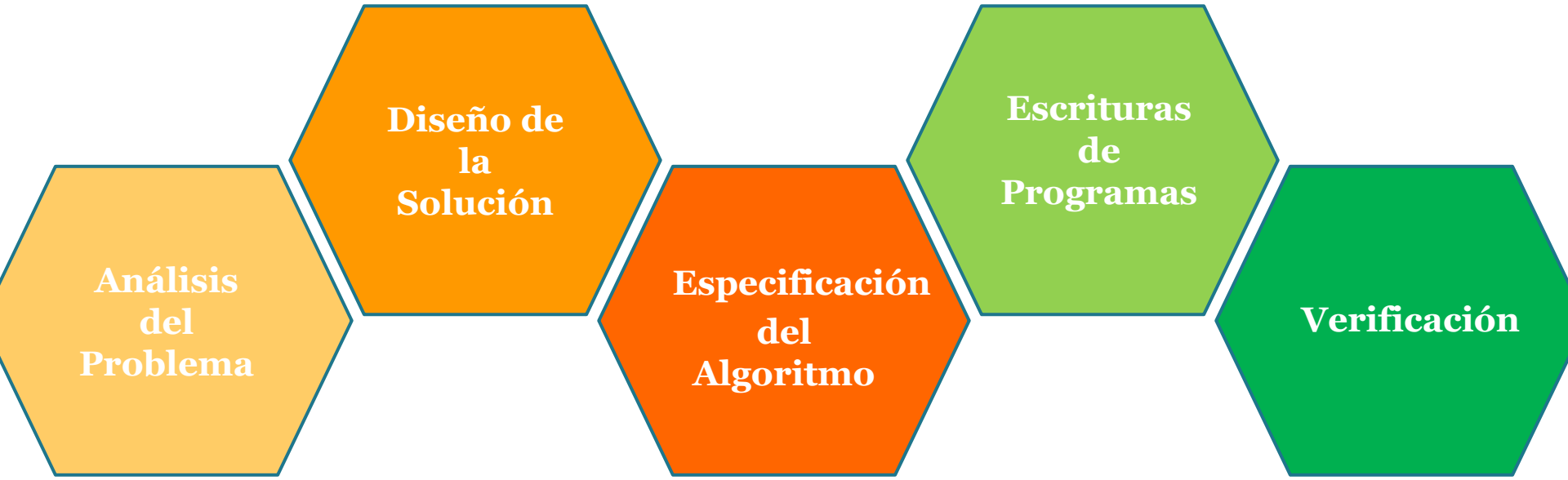
Expresión de soluciones ejecutables en la PC.

Realizar una solución ejecutable en una computadora usando un lenguaje de programación.

Del problema real a su solución por computadora



Etapas de Resolución de Problemas



Análisis del problema
La importancia del contexto

La definición del contexto es importante para analizar y diseñar la solución usando computadoras.

Impone restricciones y consideraciones.

Diseño de la solución
Descomposición - Modularización

Se usará la metodología top-down (arriba-abajo) de descomposición de problemas para desarrollar el sistema de software.

Se obtendrán módulos que deberán estar ligados entre si para obtener la solución final.

Algoritmos de los módulos

- Cada uno de los módulos habrá de tener su propio algoritmo.
- La elección del algoritmo es importante , de ella depende la eficiencia de la solución.

Escritura de programas

- Un algoritmo es una especificación simbólica que debe convertirse a un programa real sobre un lenguaje de programación concreto.

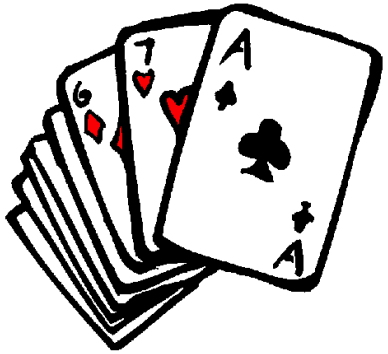
Resuelto el sistema, se continúa con la etapas (o fases) de implementación

- Pruebas (testing)
- Depuración
- Alternativas de diseño y estilo

Elementos básicos en su etapa de aprendizaje !!!!!

Planteemos un Problema de ejemplo

Problema



Cuando se baraja una mazo de cartas, se toma el mazo completo y se divide en dos, posteriormente se juntan los dos montones en un nuevo mazo poniendo una carta del primer montón y una carta del segundo montón, y así posteriormente hasta que no quede ninguna carta en ninguno de los montones.

Escriba un programa que simule el barajeo perfecto de un mazo de cartas.

Análisis del Problema

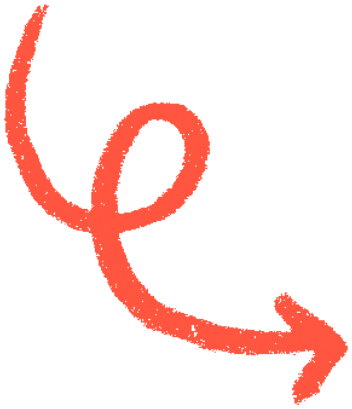
Importante! : Antes de comenzar, debemos asegurarnos de entender con claridad el problema antes de abocarnos a encontrar una solución.

Para realizar un análisis del problema debemos Contestarnos las siguientes preguntas:

* ¿Cuáles son los datos ha utilizar?
* ¿Qué transformaciones sufren los datos en el proceso?
* ¿Cuál es el objetivo a resolver?.

Es hora de responder

Pizarra interactiva

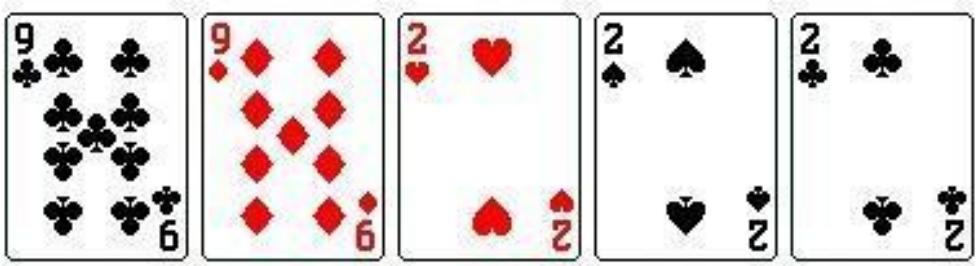


http://linoit.com/users/mvelazquez/canvases/Clase%20N%C2%B0%201%20-%20Programaci%C3%B3n

Diseño de la Solución

- Leer mazo de cartas
- Dividir el mazo en dos
- Mezclar mazo
- Mostrar mazo

Diseño de la Solución



Especificación del Algoritmo

Algoritmo Barajeo
ENTRADA: carta: entero>0
SALIDA: mazoMezcla: vector de entero>0

A1 - leer mazoEntero
A2 – Dividir mazo(mazoEntero)
A3 – mazoMezcla ← Mezclar_mazo(mazo1, mazo2)
A4 - ESCRIBIR(mazoMezcla)
A5- PARAR

Especificación del Algoritmo

A1 - leer mazoEntero
Hacer 10 veces (i=1, …, 10)
LEER(CARTA)
$mazoEntero_i \leftarrow$ carta
fin del Hacer

A2 - dividir mazo
Hacer 5 veces (i=1, …, 10)
$maz01_i \leftarrow mazoEntero_i$
$maz02_i \leftarrow mazoEntero_{i+5}$
fin del Hacer

Especificación del Algoritmo

FUNCION: Mezclar_mazo(mazo1, mazo2)
ARGUMENTO: mazo1, mazo2: entero > 0
RESULTADO: vector de enteros > 0
VBLE.AUX: mazoMezclado, k: entero > 0

HACER 5 VECES (k=1,..., 5)

$mazoMeclado_m \leftarrow mazo1_k$

$mazoMeclado_{m+1} \leftarrow mazo2_k$

$m \leftarrow m+1$

fin del Hacer

Mezclar_mazo $\leftarrow$ mazoMezclado

Comparación

```
int main()
{
   int
   mazoIni[10]={1,2,3,4,5,6,7,8,9,10};
   int mazo1[5];
   int mazo2[5];
   int mazoFinal[10];

   leerMazoEntero(mazoIni, 10);
   dividirMazo(mazoIni, mazo1,
   mazo2, 5);
   mezclarMazo(mazoFinal, mazo1,
   mazo2, 5);
   mostrarMazo(mazoFinal, 10);
   return 0;
 }
```

Algoritmo Barajeo
ENTRADA: carta: entero>0
SALIDA: mazoMezcla: vector de entero>0
Vbles. Aux.:

A1 - leer mazoEntero

A2 – Dividir mazo(mazoEntero)

A3 – mazoMezcla ← Mezclar_mazo(mazo1, mazo2)

A4 - ESCRIBIR(mazoMezcla)

A5- PARAR

Verificación

Antes de dar por finalizada cualquier labor de programación, es fundamental preparar un conjunto de datos representativos del problema que permiten probar el programa cuando se ejecute, y así verificar resultados.

Importante! : Cuanto mas exhaustivas sean las pruebas mayor seguridad se tendrá que el funcionamiento del programa es correcto, por lo tanto menor posibilidad de errores.

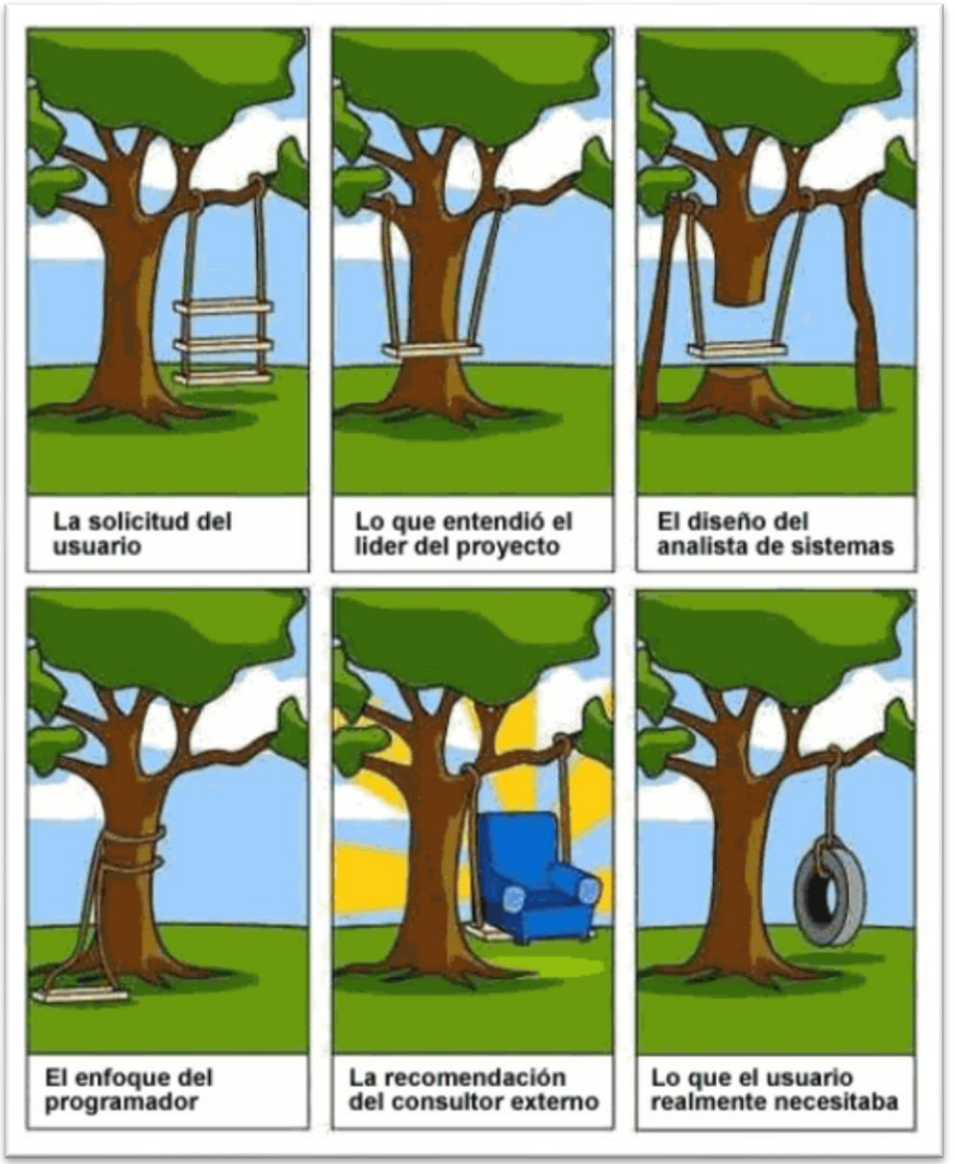
La buena programación y el buen estilo

Un buen estilo hace que un programa sea fácil de leer e interpretar.

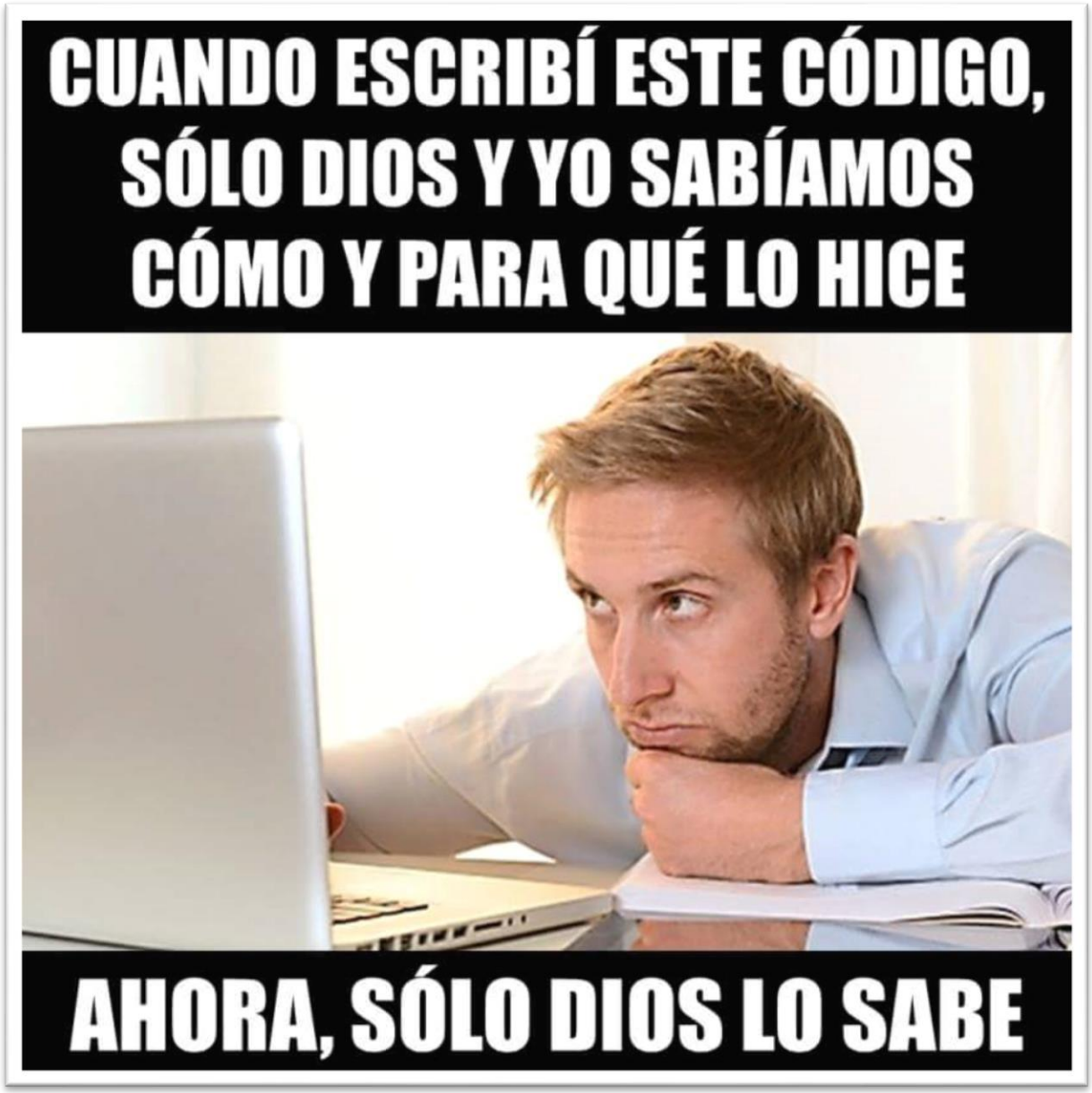
Los principios básicos : sentido común, lógica directa, expresión natural, nombres con significado, comentarios útiles, entre otros.

Tres criterios básicos a tener en cuenta

Correctitud…
Resultados deseados.



Claridad


CUANDO ESCRIBÍ ESTE CÓDIGO,
SÓLO DIOS Y YO SABÍAMOS
CÓMO Y PARA QUÉ LO HICE
AHORA, SÓLO DIOS LO SABE

Eficiencia…. rentabilidad en función de tiempo y espacio

```c
#include <stdio.h>

int main()
{
    printf("Hasta la próxima clase!!\n");

    return 0;
}
```