LICENCIATURA EN INFORMATICA

PROGRAMADOR UNIVERSITARIO

UNIVERSIDAD NACIONAL DE TUCUMÁN
facet
FACULTAD DE CIENCIAS EXACTAS Y TECNOLOGÍA

PROGRAMACIÓN

Introducción

Lic. Mariela A. Velázquez

Páginas de las Carreras

Pág. P.U:
https://www.facet.unt.edu.ar/programadoruniversitario/

Pág. Lic. en Informática:
https://www.facet.unt.edu.ar/licinformatica/

Redes sociales

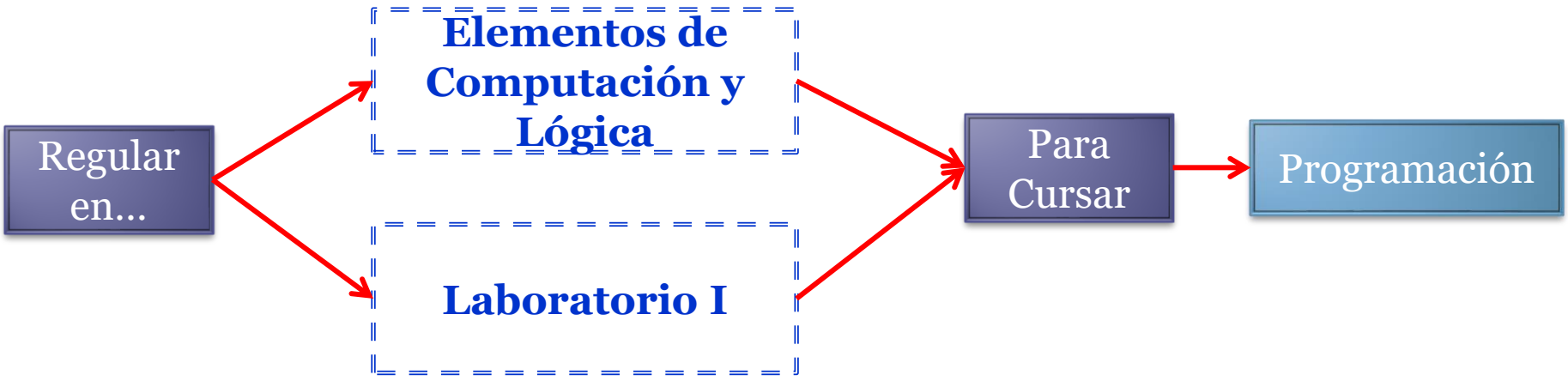
Grupo de Facebook: **https://www.facebook.com/groups/LiPuFacet**

Facebook: **https://www.facebook.com/estudiainformaticaunt/**

Twitter: **@programadorUNT**

Instagram: **@lipu.facet**

Correlativas de Programación



Fechas importantes:

- Inscripción en las materias del segundo cuatrimestre:
  - **Lunes 12/08/2019 al lunes 19/08/2019.** IMPORTANTE
- Semana de Parcial (Primer Parcial):
  - **Sábado 28/09/2019 a Sábado 05/10/2019**
- Semana de Parcial (Segundo Parcial):
  - **Sábado 16/11/2019 a sábado 23/11/2019**
- Finalización de clases del segundo Cuatrimestre 2019:
  - **Sábado 30/11/2019**

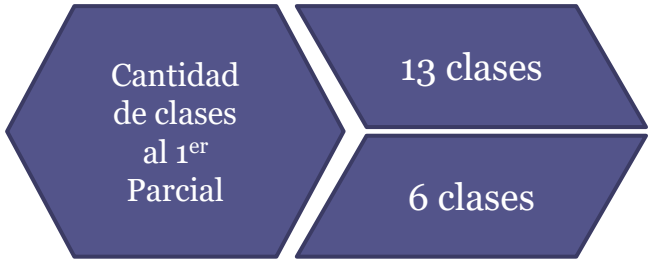
Mesas de Exámenes Finales
(Turno Diciembre)

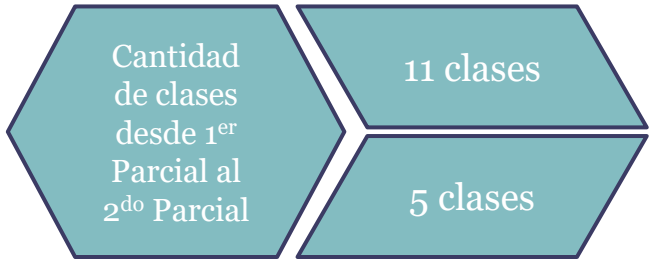
- Jueves 05/12/2019 Mesa independiente de la anterior
- Jueves 12/12/2019 Mesa independiente de la anterior
- Jueves 19/12/2019 Mesa independiente de la anterior

Link de calendario académico 2019: Calendario académico

Reglamento de la materia

- ✓ A la fecha de cada parcial, el alumno deberá tener el 80% de asistencia a los Trabajos Prácticos y Teorías dados por la Cátedra, caso contrario se lo considerará libre.





- ✓ A la fecha de cada parcial deben tener el 80% de los trabajos prácticos.
- ✓ La asistencia a los parciales es obligatoria, la falta a cualquiera de ellos determina que el alumno quede libre en la materia.
- ✓ Sólo se podrán justificar las inasistencias a parciales con certificados médicos.

Reglamento de la materia

- ✓ Condiciones para regularizar :

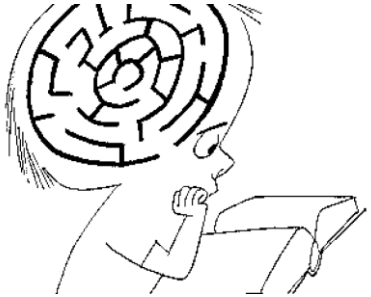
Si (notaPrimerParcial 1>= 5 ∧ notaSegundoParcial >= 5) entonces

Regulariza

Sino

Si (notaPrimerParcial 1< 5 ∧ notaSegundoParcial >= 5) entonces

Recupera el primer Parcial



Sino

Recupera Integral

- ✓ Condiciones para promocionar:

Si (notaSegundoParcial >= 7 ∧ promedio>=7 ∧ evaluativo_1 = AP evaluativo_2 = AP) entonces

Promociona

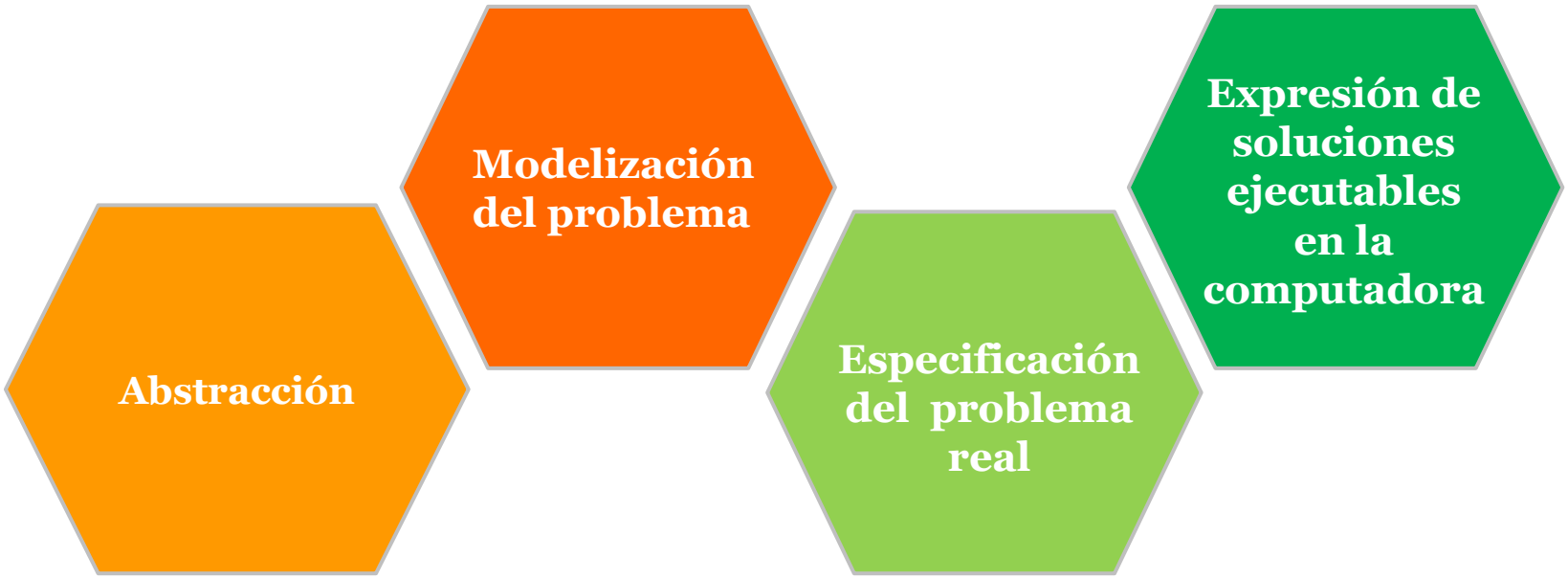
Fin_si

Comenzamos!!!

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

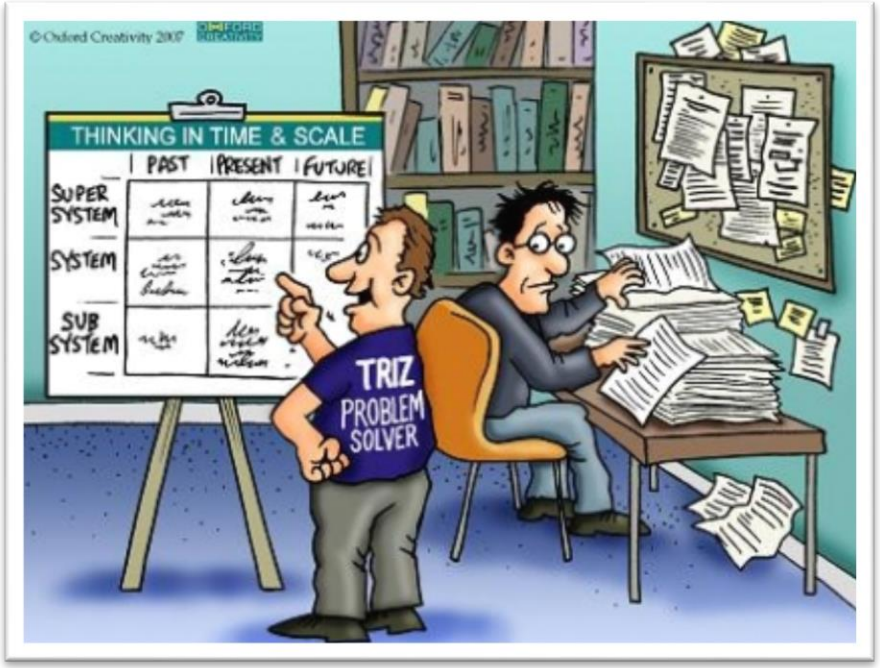


- **Abstracción**

Analizar el mundo real e interpretar los aspectos esenciales de un problema y expresarlo en términos precisos.

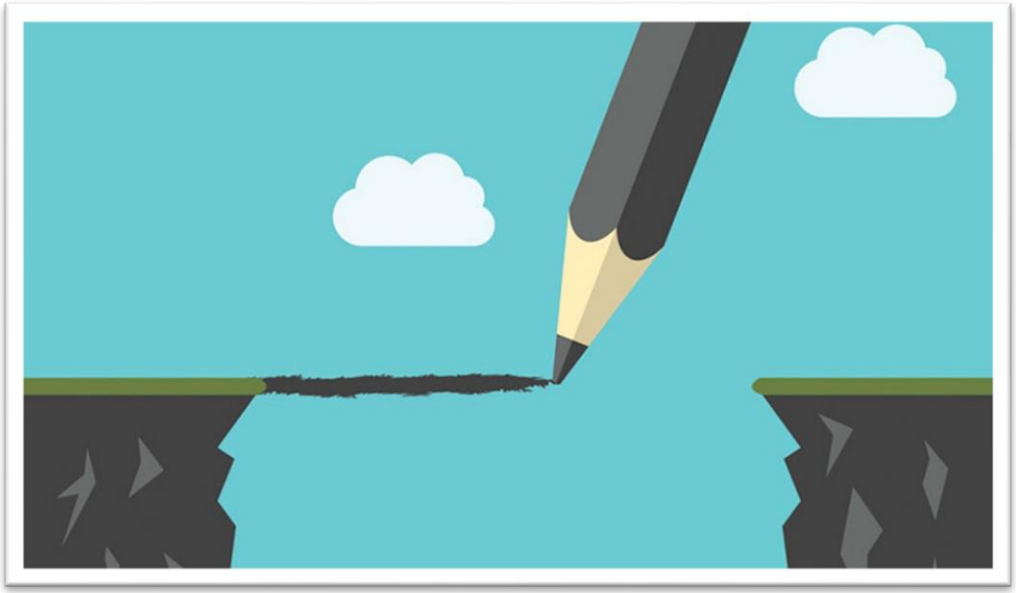
- **Modelización del problema**

Abstraer un problema concreto y simplificar su expresión, encontrando sus aspectos principales, los datos que se habrán de procesar y el contexto.



- **Especificación del problema real**

El proceso de analizar los problemas del mundo real y determinar en forma clara y concreta el objetivo que se desea.



- **Expresión de soluciones ejecutables en la computadora.**

Es la escritura de un programa que represente una solución ejecutable en una computadora usando un lenguaje de programación.

¿Cómo determinar la altura de esculturas en vertical? Por ejemplo, ¿cuál es la altura del Cristo de San Javier?

Abstracción

Los aspectos mas relevantes:

Lo mas notable que hay en la escultura del problema es la apariencia , ¿ cómo se ve la escultura? ¿Cuál es su forma ? ¿Puedo pensarlo desde la geometría?

Aspectos esenciales del ejemplo

Es una escultura

- Con eje vertical, montada sobre una base.
- Está en la cumbre de una montaña y llegar a su base es imposible…

Modelización del problema

Para simplificar el problema, se considera un Modelo geométrico para su solución.

Se considerará un triangulo rectángulo y se aplicará la fórmula de la tangente.

Contexto geométrico: mínimamente necesito un par de datos más….¿desde donde voy a mirar el punto mas alto de la estatua?

Modelo matemático para resolver el problema, se usan conceptos de trigonometría

¿Especificación de los problemas reales?

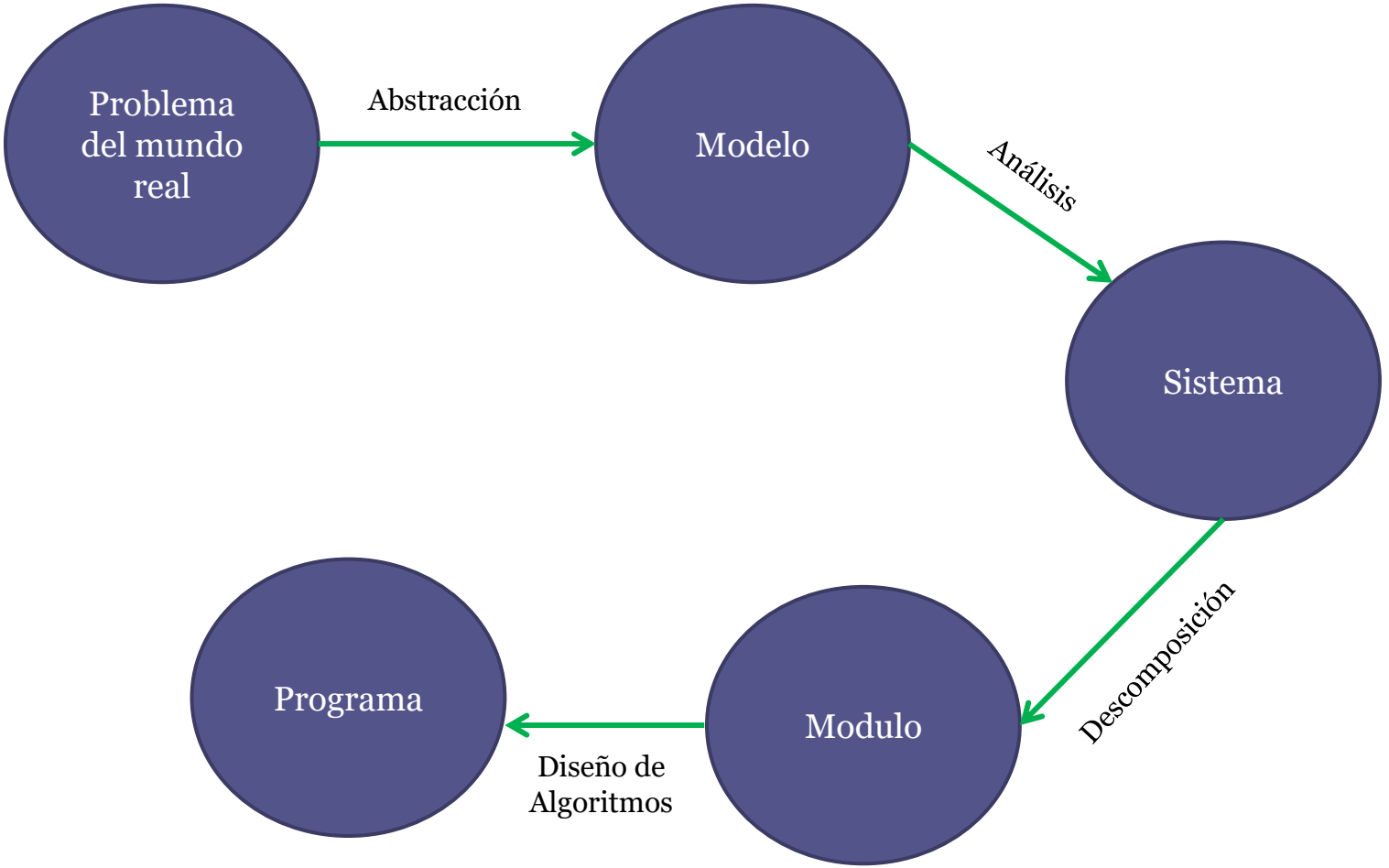
Dado el modelo elegido, es necesario replantear el problema y los datos a considerar. El objetivo sigue siendo encontrar a altura del Cristo de San Javier.

“ Si no sabe donde ir no sabrá donde debe llegar”

Expresión de soluciones ejecutables en la computadora

Es la escritura de un programa que represente una solución ejecutable en una computadora usando un lenguaje de programación.

Del problema real a su solución por computadora



- La programación estructurada tradicional se basa fundamentalmente en la ecuación de Niklaus Wirth:

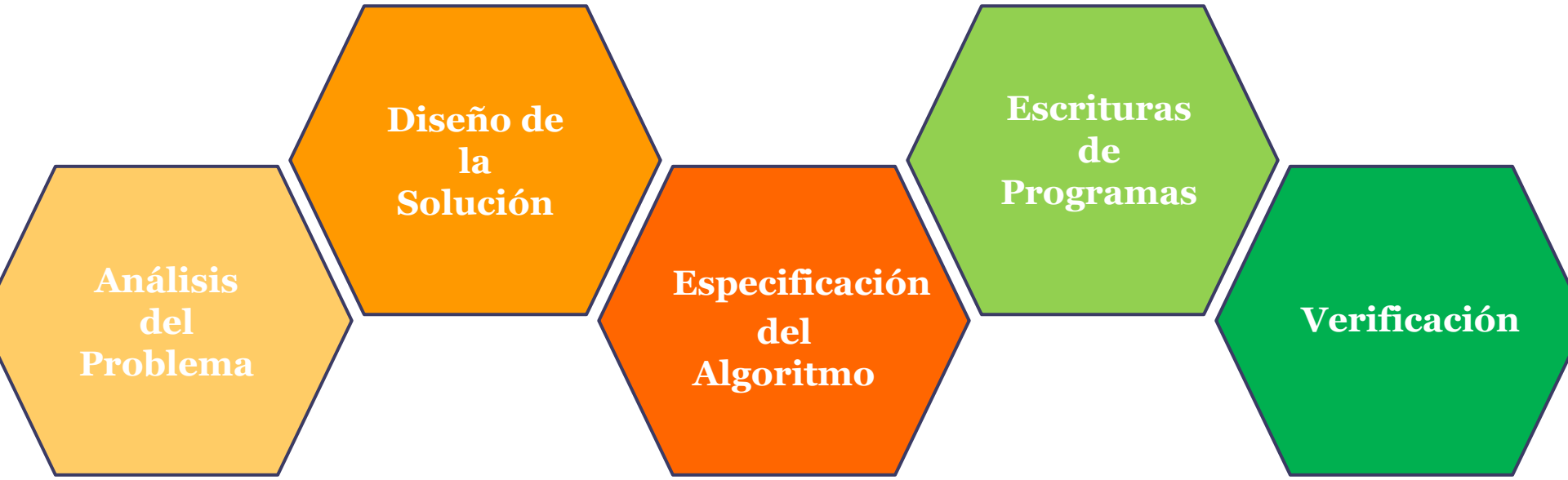
Algoritmos + Estructura de Datos = Programas

Esta ecuación significa que en la programación estructurada u orientada a procedimientos los datos y el código se trata por separado y lo único que se realiza son funciones o procedimientos que tratan esos datos y los van pasando de unos a otros hasta que se obtiene el resultado que se desea.

Concepto de Algoritmos

- Algoritmo es la especificación rigurosa de la secuencia de pasos (instrucciones) a realizar sobre un autómata para alcanzar un resultado deseado en un tiempo finito.

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.

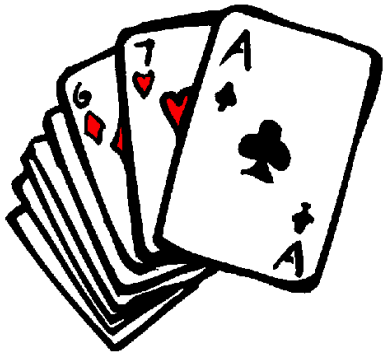
Resuelto el sistema, se continúa con la etapas (o fases) de implementación

- 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.

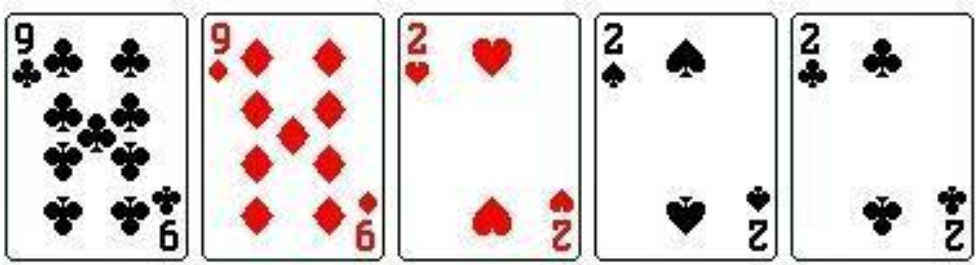
Para realizar un análisis del problema debemos Contestarnos las siguientes preguntas:

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

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)
$mazoEntero_i$
carta
fin del Hacer

A2 - dividir mazo
Hacer 5 veces (i=1, …, 10)
$mazo1_i$ $mazoEntero_i$
$mazo2_i$ $mazoEntero_{i+5}$
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)

$mazoMeclado_m$ $mazo1_k$

$mazoMeclado_{m+1}$ $mazo2_k$

m m+1

fin del Hacer

Mezclar_mazo mazoMezclado

Escritura de Programas

```
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;
 }
```

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;
   getch();
}
```

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.

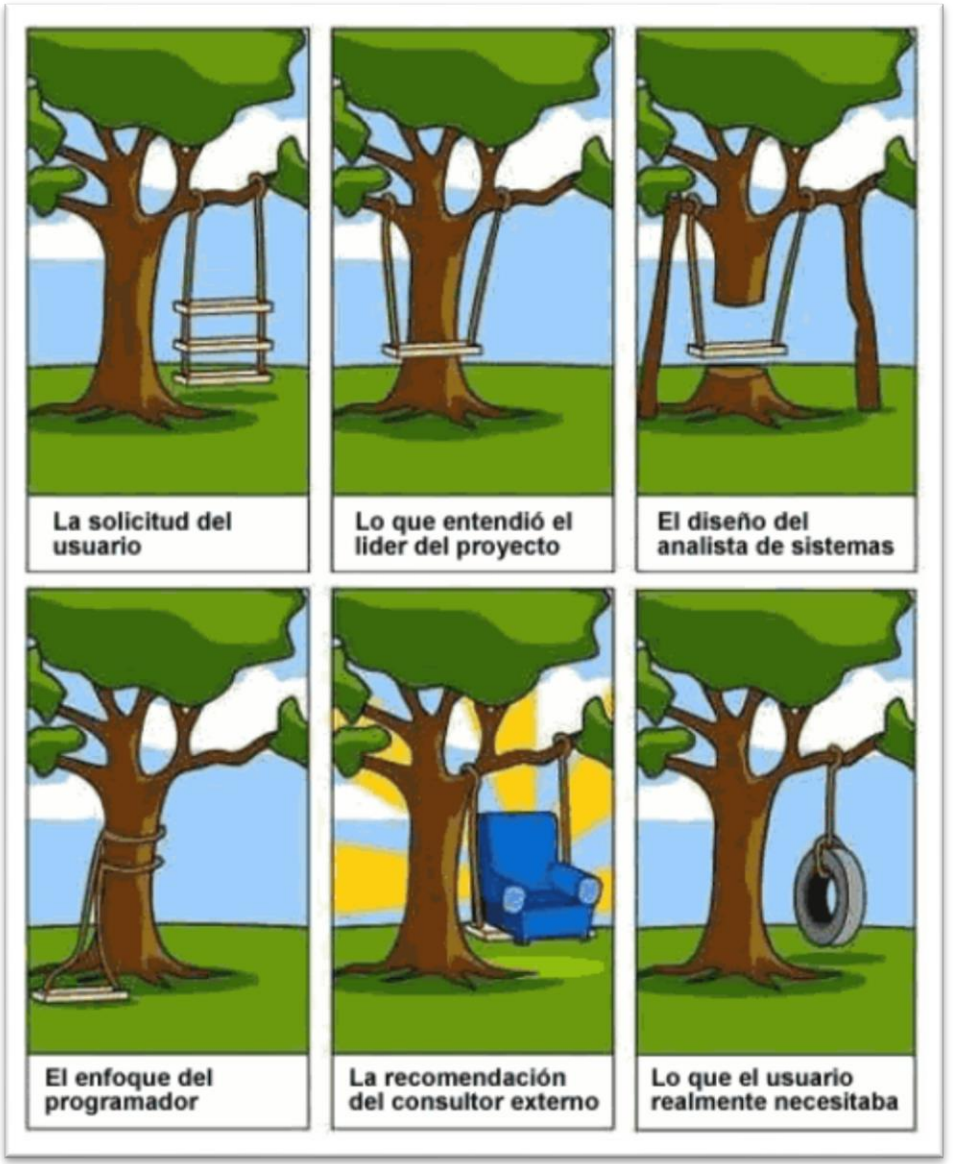
La buena programación y el buen estilo

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

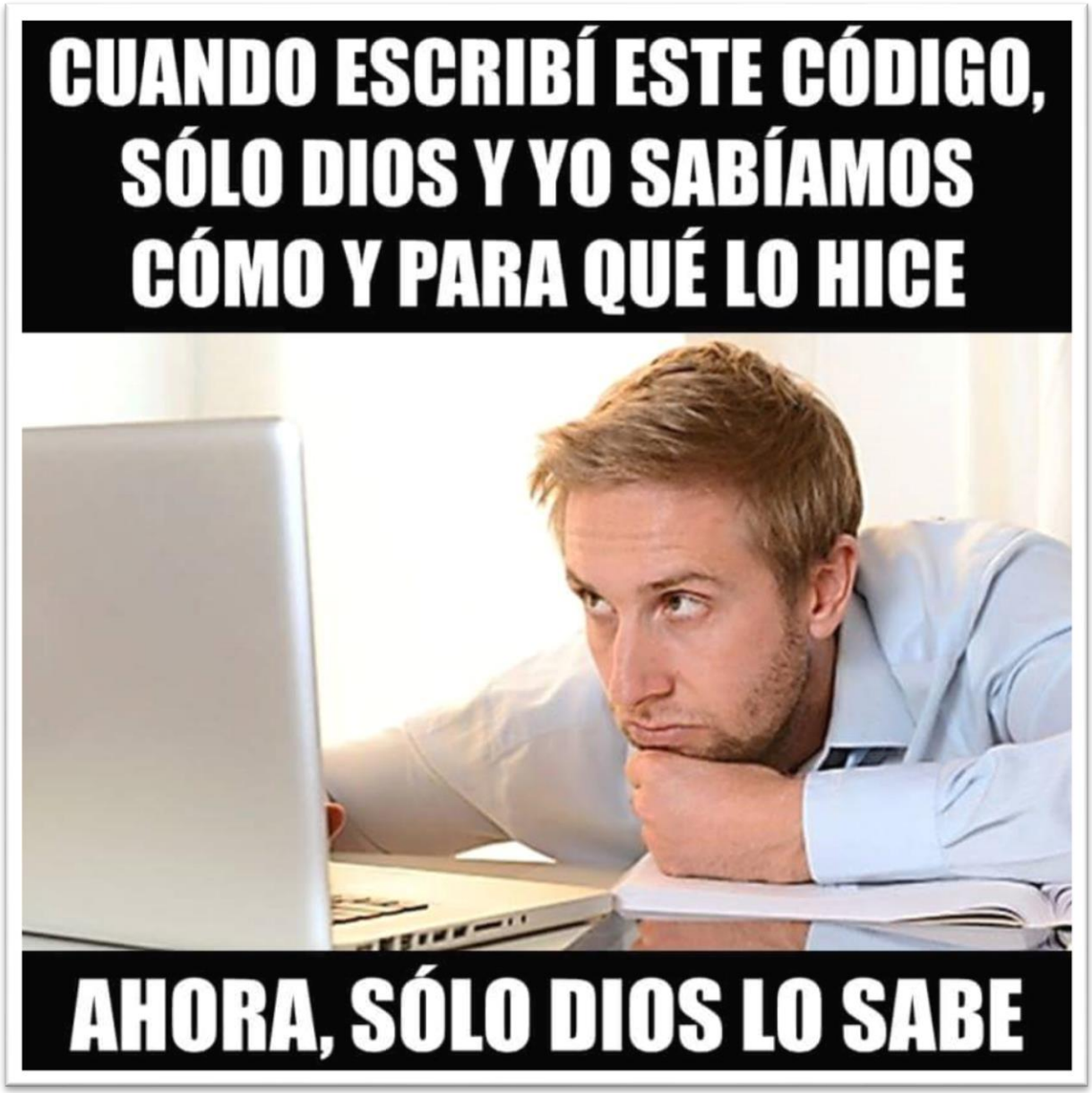
Los principios básicos : sentido común, lógica directa, expresión natural, nombres con significado, comentarios útiles, entre otros.

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