# Introducción a la Computación (Matemática)

Primer Cuatrimestre de 2019

Especificación de Problemas

# Programa

Un programa es una **secuencia finita de instrucciones**: asignaciones, condicionales, ciclos, etc.

Ejemplo:

```
int fil = 5;
while (fil >= 1) {
   int col = 1;
   while (col <= fil) {
      cout << col << " ";
      col = col + 1;
   }
   cout << endl;
   fil = fil - 1;
}
```

# Especificación, algoritmo y programa

**Especificación de un problema:**

- ¿Qué problema tenemos?
- Lenguaje formal (ej. lógica de primer orden).

**Algoritmo:** Una solución abstracta del problema (escrita para humanos).

- ¿Cómo resolvemos el problema?
- Pseudocódigo.

**Programa:** Una solución concreta del problema (escrita para computadoras).

- ¿Cómo resuelve la computadora el problema?
- Lenguaje de programación (ej. Python).

# Algoritmo

Un **algoritmo** es una secuencia finita de instrucciones que permite hallar la solución a un problema.

El algoritmo es un concepto matemático bastante anterior a las computadoras. La palabra "algoritmo" deriva del nombre del matemático persa al-Juarismi (780-850).

No todos los programas son algoritmos: Los algoritmos deben terminar siempre; es decir, la cantidad de instrucciones a ejecutar debe ser **finita**.

# Programa vs. Algoritmo

Ejemplo:

1. Moje el cabello.
2. Coloque shampoo.
3. Masajee suavemente y deje actuar por 2 min.
4. Enjuague.
5. Repita el procedimiento (desde 1). una vez.

# Especificación de problemas

La especificación de un problema describe en un lenguaje formal:

- cuáles son las entradas aceptables, y
- qué propiedades debe cumplir la salida.

Si la entrada es aceptable, el algoritmo debe terminar exitosamente y la salida debe cumplir las propiedades especificadas.

# Especificación de problemas

**Pero, ¿para qué especificar?**

- Antes de sentarnos a programar, debemos tener bien claro **qué** tenemos que resolver.
  - Error común: programar sin un objetivo claro.

- Especificar es el primer paso para después poder probar la **correctitud** de un algoritmo.
  - Un algoritmo **correcto respecto de la especificación** ejecuta una transformación de la entrada en una salida que cumple las propiedades especificadas en la especificación.

# Especificación de problemas

Una **especificación** tiene 3 partes:

1. Encabezado
   Indica el nombre, el tipo y número de parámetros, y el tipo del valor de retorno ($RV$).
2. Precondición
   Es una descripción del valor inicial de los parámetros de entrada.
3. Poscondición
   Es una descripción del valor final de los parámetros de entrada y del valor de retorno.

# Ejemplo

Problema "sumaCuadrados": dados dos números enteros, devolver la suma de los cuadrados de ambos.

Encabezado: $sumaCuadrados : a \in \mathbb{Z} \times b \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{a = a_0 \wedge b = b_0\}$
Poscondición: $\{RV = a_0 * a_0 + b_0 * b_0\}$

$RV$ es el valor de retorno del problema.

# Lenguaje de especificación

- Tipos de datos: $\mathbb{Z}$ (enteros), $\cdot[\,]$ (arreglos), $\mathbb{B} = \{\mathit{true}, \mathit{false}\}$ (booleanos, o valores de verdad).
  - Ej: $n \in \mathbb{Z}$, $x \in \mathbb{R}$, $L \in \mathbb{Z}[\,]$, $b \in \mathbb{B}$.
- Un **término** puede ser una variable, una constante o una operación aplicada a otros términos.
  - Ej: $0$, $0 + 1$, $n - 1$, true, $a \wedge b$.
- Un **predicado** es un enunciado sobre términos, que puede tomar valor verdadero o falso.
  - Ej: $x > 0$, $i + 1 = 0 \wedge j - 2 \geq 0$, $2 > 3$.

# Lenguaje de especificación (cont.)

- Operaciones booleanas (conectivos lógicos): $\neg \wedge \vee \Rightarrow$
  - Ej: $p \wedge q$, $(\neg a \vee b) \Rightarrow c$.
- Operaciones de $\mathbb{Z}$: $+ - * / |\cdot|; < = \leq > \geq$...
  - Ej: $a + b$, $|-5| < 5$.
- Operaciones de arreglos: $\cdot[\cdot]$, $|\cdot|$ (longitud).
  - Ej: $A[3]$, $|A|$.
- Funciones matemáticas auxiliares.
  - Ej: $f(x \in \mathbb{R}) = \begin{cases} 0 & \text{si} \quad x \leq 0 \\ x + 1 & \text{si} \quad x > 0. \end{cases}$
- Predicados auxiliares.
  - Ej: $Par(n \in \mathbb{Z}) \equiv (\exists k \in \mathbb{Z})\ n = 2 * k$.

# Lenguaje de especificación (cont.)

- Cuantificadores: $\forall$ (universal), $\exists$ (existencial).
  - $(\forall i \in \mathbb{Z})\ 0 \leq i < |A| \Rightarrow Par(A[i])$
    Todos los elementos del arreglo $A$ son pares.
  - $(\exists i \in \mathbb{Z})\ 0 \leq i < |A| \wedge Par(A[i])$
    Existe algún elemento par en el arreglo $A$.
- Acumuladores: $\sum$ (sumatoria), $\#$ (contador).
  - $RV = \sum_{i \in \mathbb{Z}\ /\ 0 \leq i < |A| \wedge Par(A[i])} A[i]$
    La suma de los elementos pares del arreglo $A$.
  - $RV = (\#i \in \mathbb{Z})\ 0 \leq i < |A| \wedge Par(A[i])$
    La cantidad de elementos pares en el arreglo $A$.

# Especificación de problemas

Las especificaciones son abstractas.

Por lo tanto, no modelan los problemas de representación (p.ej., *overflow*).

- `int` $\rightsquigarrow \mathbb{Z}$

Dichos problemas de representación deberán ser tenidos en cuenta durante la implementación del algoritmo, en un lenguaje de programación concreto.

# Ejemplo

Problema "div": dados dos números enteros, devolver la división entera del primero por el segundo.

Encabezado: $div : a \in \mathbb{Z} \times b \in \mathbb{Z} \to \mathbb{Z}$

Precondición: $\{a = a_0 \wedge b = b_0 \wedge b_0 \neq 0\}$

Poscondición: $\{(\exists r \in \mathbb{Z})\ 0 \leq r < |b_0| \wedge a_0 = RV * b_0 + r\}$

# Parámetros

Los parámetros se comportan como variables.
Como tales, pueden ser modificadas libremente en el algoritmo.

En la precondición le ponemos nombre a sus valores iniciales:

Precondición: $\{x = x_0 \land y = y_0\}$

En la poscondición predicamos sobre esos valores iniciales:

Poscondición: $\{RV = x_0 + y_0\}$

Si queremos que una variable de entrada cumpla cierta propiedad al final, lo especificamos en la poscondición:

Poscondición: $\{RV = x_0 + y_0 \ \land \ x = 8\}$

# Problemas sin valor de retorno

Un problema puede no tener un valor de retorno.

**Ejemplo:**
Problema "inc": dado un entero, sumarle uno.

Encabezado: $inc : x \in \mathbb{Z} \rightarrow \emptyset$
Precondición: $\{x = x_0\}$
Poscondición: $\{x = x_0 + 1\}$

Notar que no se menciona $RV$ en la poscondición.

# Problemas sin valor de retorno

**Otro ejemplo:**
Problema "incTodos": dado un arreglo de enteros, suma uno a todos sus elementos.

Encabezado: $incTodos : A \in \mathbb{Z}[] \rightarrow \emptyset$
Precondición: $\{A = A_0\}$
Poscondición: $\{|A| = |A_0| \wedge$
$(\forall i \in \mathbb{Z})\ 0 \leq i < |A| \Rightarrow A[i] = A_0[i] + 1\}$

# Más ejemplos

Problema "distinto": dado un número entero, dar otro distinto.

Encabezado: $distinto : a \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{a = a_0\}$ sobreespecificación
Poscondición: $\{RV \neq a_0 \wedge RV > 0\}$

Encabezado: $distinto : a \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{a = a_0 \wedge a_0 > 0\}$ subespecificación
Poscondición: $\{RV \neq a_0\}$

# Sobreespecificación y Subespecificación

**Especificar** un problema es describirlo formalmente, a partir de un texto en lenguaje natural.

**Sobreespecificar** es (sin advertirlo) imponer una poscondición más restrictiva que lo necesario, o una precondición más débil.

- Limita excesivamente los posibles algoritmos a considerar.

**Subespecificar** es (sin advertirlo) imponer una poscondición más débil que lo necesario, o una precondición más restrictiva.

- No limita suficientemente los posibles algoritmos a considerar.

# Ejemplo

Encontrar la raíz cuadrada entera de un número entero. Ejemplos: $raizEntera(8) \to 2$, $raizEntera(9) \to 3$.

Encabezado: $raizEntera : n \in \mathbb{Z} \to \mathbb{Z}$
Precondición: $\{n = n_0\}$
Poscondición: $\{RV * RV \leq n_0 < (RV + 1) * (RV + 1)\}$

¿Está bien? No, porque no hay solución cuando $n_0 < 0$. La especificación **no puede ser satisfecha** por un algoritmo, porque la precondición es demasiado débil (ejemplo de sobreespecificación).

Agregamos entonces $n_0 \geq 0$ a la precondición.

El lenguaje de especificación es más expresivo que el modelo de cómputo: nos permite especificar **problemas sin solución**.

# Repaso de la clase de hoy

- Especificación de problemas.
- Algoritmo vs. programa.
- Encabezado, precondición y poscondición.
- Lenguaje de especificación.
- Sobre- y subespecificación.

**Próximos temas**

- Verificación formal de algoritmos.
- Verificación formal de asignación, secuencialización y condicional.