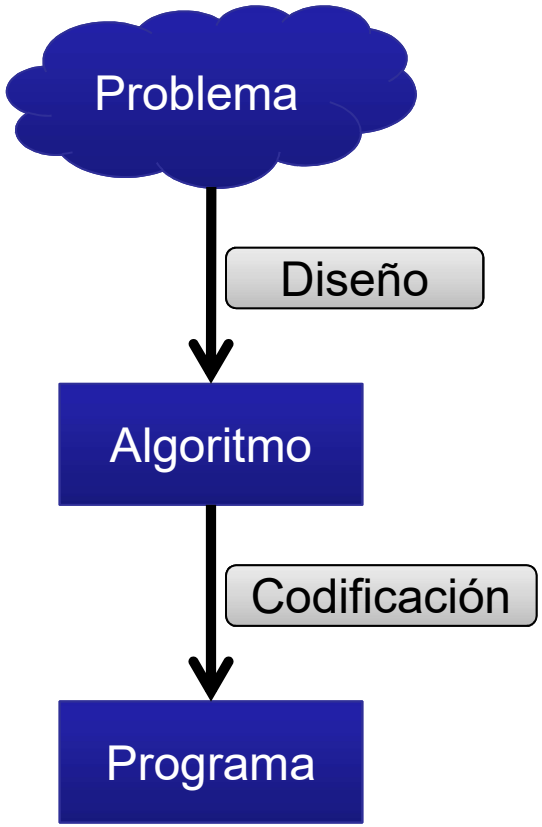
Fundamentos de Programación

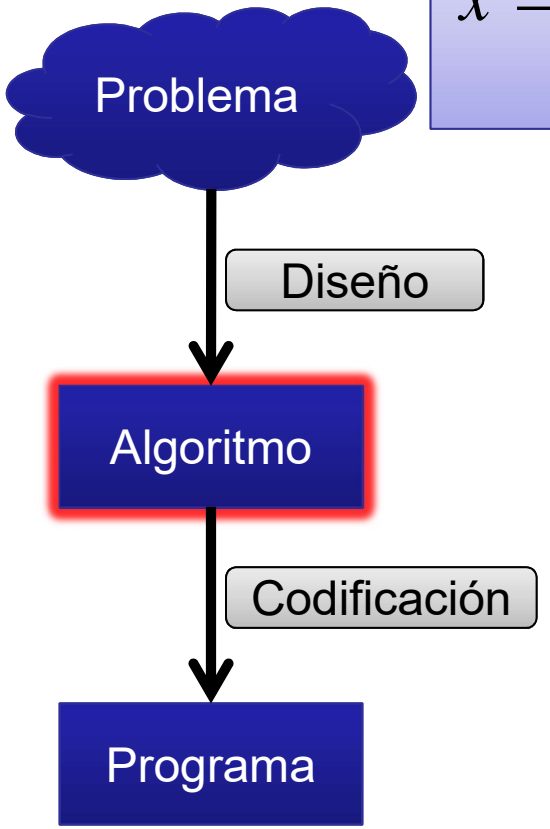
Algoritmos y Programas

Algoritmos y Programas



- **Algoritmo:** Descripción "formal" de los pasos a seguir para resolver un problema dado.
  - No ambiguo y preciso.
  - Lenguaje cercano a los humanos.
- **Programa:** Traducción de un algoritmo a un lenguaje de programación.
  - No ambiguo y preciso.
  - Lenguaje cercano a la máquina.
  - Podemos utilizar distintos lenguajes para un mismo algoritmo (C,JAVA, PHP, ...)

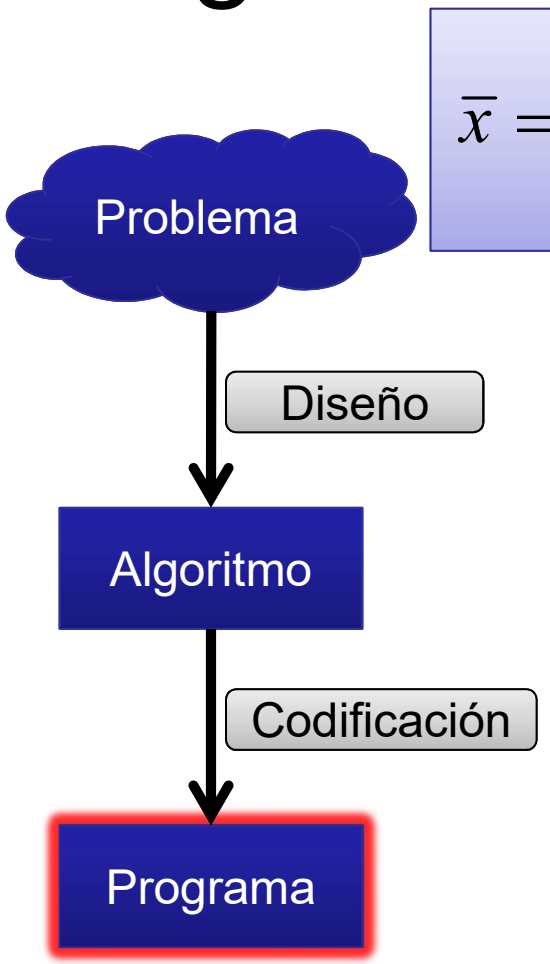
Algoritmos y Programas



$$\bar{x} = \frac{1}{k}\sum_{i=1}^{k} x_i$$

```
algoritmo media
    var
        n, suma, i, k: entero;
        resultado: real;
    fvar
    suma := 0;
    i := 1;
    leerEntero(k);
    mientras i ≤ k hacer
        leerEntero(n);
        suma := suma + n;
        i := i + 1;
    fmientras
    resultado := enteroAReal(suma)/enteroAReal(k);
    escribirReal(resultado);
falgoritmo
```

Algoritmos y Programas



$$\bar{x} = \frac{1}{k}\sum_{i=1}^{k} x_i$$

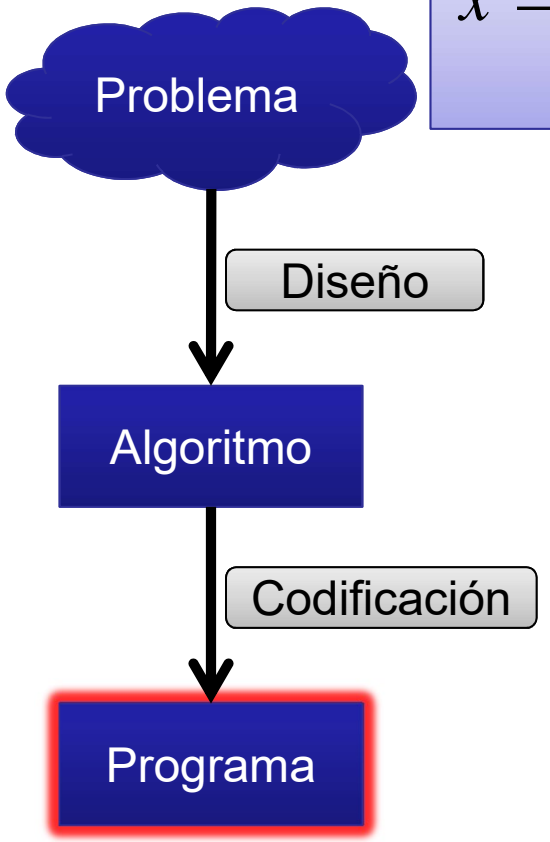
```
void main(void) {
    int n, suma, i, k;
    float resultado;

    suma=0;
    i=1;

    scanf("%d",&k);
    while(i <= k) {
        scanf("%d",&n);
        suma=suma+n;
        i=i+1;
    }
    resultado = (float)suma/(float)k;
    printf("%f",resultado);
}
```

media.c

Algoritmos y Programas

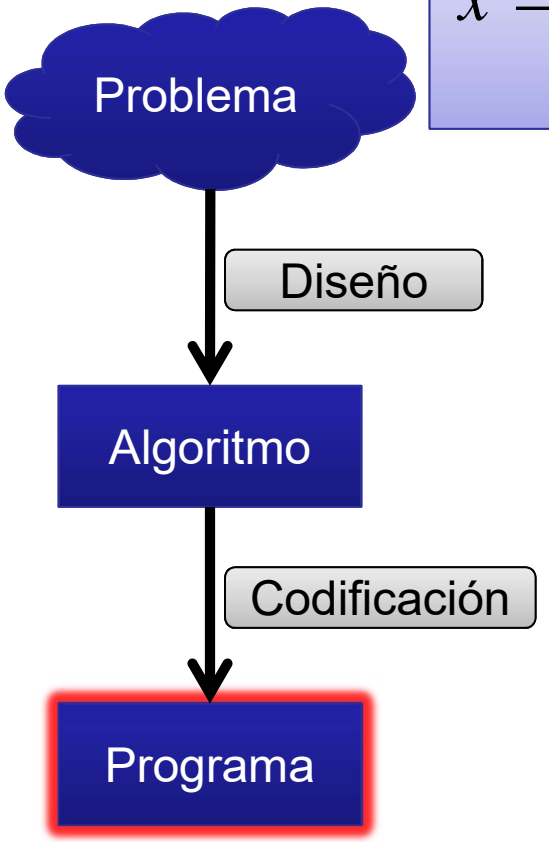


$$\bar{x} = \frac{1}{k}\sum_{i=1}^{k} x_i$$

```
import java.io.*;
public class Media{
   public static void main(String[] args) {
      int n, suma, i, k;
      float resultado;
      String line;
      BufferedReader is = new BufferedReader(
         new InputStreamReader(System.in));
      suma=0;
      i=1;
      try {
         line = is.readLine();
         k = Integer.parseInt(line);
      } catch (NumberFormatException ex) {
         System.err.println("No es un entero valido: " + line);
      } catch (IOException e) {
         System.err.println("Error desconocido: " + e);
      }
      while(i <= k) {
         try {
            line = is.readLine();
            n = Integer.parseInt(line);
         } catch (NumberFormatException ex) {
            System.err.println("No es un entero valido: " + line);
         } catch (IOException e) {
            System.err.println("Error desconocdo: " + e);
         }
         suma=suma+n;
         i=i+1;
      }
      resultatdo= (float)suma/(float)k;
      System.out.print(resultado);
   }
}
```

Media.java

Algoritmos y Programas



$$\bar{x} = \frac{1}{k}\sum_{i=1}^{k} x_i$$

```
<html>
<body>
<p> Programa media</p>
<?php

  $line = $_GET["line"];
      $valores = explode(" ",$line);

      $suma = 0;
      $i = 1;

      $k = array_shift ($valores);
      while ($i <= $k)
      {
            $n = array_shift ($valores);
            $suma = $suma + $n;
            $i = $i +1;
      }
      $resultado = $suma / $k;
      print $resultado;
?>
</body>
</html>
```

media.php

Algoritmos y Programas

- **Comentarios finales:**
  - **Los algoritmos** os ayudarán a organizar los pasos para resolver un determinado problema. Aunque en las actividades teóricas se pide trabajar con algoritmos, en las partes prácticas podéis trabajar directamente con el lenguaje de programación, haciendo el diseño y la codificación en el mismo paso. De todos modos, hasta que no adquiráis mucha práctica programando, os será de ayuda pensar la estructura del lenguaje algorítmico para evitar que vuestros programas sean caóticos y desorganizados. En problemas complejos, también es aconsejable hacer un buen diseño inicial.
  - **Un buen programa** no es *solo* ese que funciona, o sea, que compila y se ejecuta sin errores, obteniendo los resultados esperados. Un buen programa *también* debe ser legible, optimo, mantenible, estable, ... A continuación os damos algunos consejos y buenas prácticas:
    - Comentad bien el código, con comentarios que ayuden a entender qué se está haciendo, y evitando comentarios obvios.
    - Utilizad nombres de variables entendibles, que faciliten entender para qué se utiliza cada variable
    - Utilizad controles de errores, comprobando que los valores de los parámetros están dentro de los rangos que esperáis, y desconfiando siempre de cualquier entrada externa a vuestro programa, especialmente de la entrada de información por teclado por parte del usuario. Si un paso de vuestro programa puede fallar, verificad que ha funcionado antes de continuar.
    - Intentad pensar en el coste computacional de vuestras operaciones, especialmente dentro de composiciones iterativas. Hacia el final de la asignatura veremos maneras de calcular éste coste de modo formal, pero el sentido común os ayudará si paráis a penar que hace vuestro algoritmo.
    - Haced un código modular, con funciones y acciones para aquellas operaciones que sean generales, o que se repitan muchas veces. Reutilizar código ahorra mucho tiempo y facilita la detección de errores.
    - Probad vuestro código, tanto por lo que a resultados finales refiere, cómo a resultados parciales, probando vuestras funciones y acciones en casos conocidos. Debéis buscar la peor situación, y comprobar que vuestros mecanismos de control de errores funcionen.