

## UNIDAD 1: Resolución de problemas particulares

Vimos qué es un problema

método para resolver problemas (Polya)

- Determinar el objetivo
- Determinar los datos relevantes (que me ayudarán a conseguir el objetivo)
- Representación de los datos relevantes y objetivo, y las relaciones entre ellos (fórmulas, dibujo, geometría, matemática, lógica, diagramas, grafos, conjuntos, etc)
- Secuencia de pasos
- Resultado
- Verificación

## UNIDAD 2: Algoritmos (Clases de problemas)

¿Qué es un Algoritmo? = Receta = secuencia finita de pasos

**Especificar** (Sintaxis - escribir) y **ejecutar** (cómo funciona cada instrucción) un algoritmo

Propiedades:

Entrada/Salida

Precisión: Instrucciones no ambiguas

Determinismo: Si ejecuto el algoritmo varias veces, cada vez con la misma entrada, la salida es la misma.

Correctitud: es efectivo, es decir, resuelve el problema para el que se diseñó. Para cada entrada produce la salida deseada y termina en un tiempo finito.

Generalización: se aplica a un conjunto de entradas (por esto resuelve clases de problemas)

## UNIDAD 3: Expresiones → muy importantes, porque transforman la entrada en Salida

Tipos de datos (determinan el valor que puede almacenar una variable y por lo tanto que operaciones se puede hacer con ella). Ejemplo si es un String se puede concatenar, si un entero/float se puede sumar/restar, si un booleano se puede aplicar un operador lógico (Not And Or)


Clasificación de Operadores

Según la cantidad de operandos (unarios, binarios y ternarios)

Según el tipo de dato del resultado.

## Sintaxis / convenciones de Sintaxis

|                 | Convención para Pseudocódigo   | Convención para PHP   |
|-----------------|--|---|
| <b>Variable</b> | Identificador: utilizaremos notación <u>lowerCamelCase</u> , letras mayúsculas para inicializar las palabras, excepto la primera letra. No utilizaremos guión bajo (_). Los identificadores deben ser nombres significativos al problema que estamos resolviendo.<br><br>(ejemplos: ladoMenor, ladoMayor, perimetroRectangulo) | Identificador (con la misma convención que el Pseudocódigo) al que antepondremos el signo \$<br><br>(ejemplos: \$ladoMenor, \$ladoMayor, \$perimetroRectangulo) |

|                       |  |   |
|-----------------------|--|---|
|                       |  |   |
| <b>Tipos de Datos</b> | <p><b>Entero</b></p> <p><b>Boolean</b></p> <p><b>Float</b> (se utiliza el punto como separador de decimales)</p> <p><b>String</b> (para delimitar la cadena de caracteres usaremos comillas dobles: " ")</p> | <p>En PHP, el tipo de dato de una variable se establece dinámicamente, cuando se ejecuta el programa, y depende del valor que se le asigne a la variable. En otros lenguajes, ej. JAVA, los tipos deben declararse explícitamente.</p> <p><b>int</b></p> <p><b>boolean</b> (valores TRUE o FALSE)</p> <p><b>float</b> (se utiliza el punto como separador de decimales)</p> <p><b>String</b> (para delimitar la cadena de caracteres usaremos comillas dobles: " ")</p> <p>(<a href="http://php.net/manual/es/language.types.php">http://php.net/manual/es/language.types.php</a>)</p> <p>Parcial:</p> <pre>echo "hola \n"; //hola salto de linea echo 'hola \n'; //hola \n  echo "hola"; //hola echo 'hola'; //hola  if (\$tipoElectrodomestico == 'tv'){ .... }</pre> |
| <b>comentarios</b>    | <p>Comentarios de varias líneas:</p> <pre>(*comentario varias lineas *)</pre>  | <p>Comentarios de varias líneas:</p> <pre>/* comentario varias lineas */  //comentario 1 linea</pre>  |

|                                    |   |   |
|------------------------------------|---|---|
| <b>Instrucción de Asignación</b>   | <p>nombreVariable &lt;-- <b>Expresion</b></p> <p>Donde <b>Expresion</b> puede ser:</p> <ul style="list-style-type: none"> <li>• un valor</li> <li>• una variable con valor</li> <li>• combinación de operandos y operadores</li> <li>• funciones que retornen valores (tema de unidad 4)</li> </ul> <p>Ejemplo:</p> <p>ladoMenor ← 20</p> | <p>\$nombreVariable = <b>Expresion</b> ;</p> <p>Donde <b>Expresion</b> puede ser:</p> <ul style="list-style-type: none"> <li>• un valor</li> <li>• una variable con valor</li> <li>• combinación de operandos y operadores</li> <li>• funciones que retornen valores (tema de unidad 4)</li> </ul> <p>Ejemplo:</p> <p>\$ladoMenor = 20 ;</p> <p>(observación: en php las instrucciones terminan con ; )</p> |
| <b>Instrucción de Entrada</b>      | <p>Ejemplo:</p> <p>LEER( ladoMenor )</p>  | <p>Ejemplo:</p> <p>\$ladoMenor = trim(fgets(STDIN)) ;</p> <p>(observación: en php las instrucciones terminan con ; )</p>  |
| <b>Instrucción de Salida</b>       | <p>Ejemplo:</p> <p><b>ESCRIBIR</b>("El perímetro del rectángulo es ", perimetroRectangulo)</p> <p>(observación:<br/>Para la operación de concatenación de string se utiliza coma ",")</p>   | <p>Ejemplo:</p> <p><b>echo</b> "El perímetro del rectángulos es " . perimetroRectangulo;</p> <p>(observación: en php las instrucciones terminan con ;<br/>Para la operación de concatenación de string se utiliza punto ".")</p>  |
| <b>Separación de instrucciones</b> | <p>Una instrucción por renglón, indentando adecuadamente las instrucciones.</p>   | <p>Utilizando punto y coma ";"</p> <p>(Por prolijidad siempre conviene 1 instrucción por renglón, indentando adecuadamente las instrucciones)</p>   |

|                                  |   |  |
|----------------------------------|---|--|
| <b>Estructura<br/>Secuencial</b> | <b>PROGRAMA</b> Nombre<br> (*descripción del algoritmo*)<br> Declaración de variables: Tipo<br>nombresVariables<br> instruccion1<br> ...<br>  instruccionN<br><b>FIN PROGRAMA</b> | <pre>&lt;?php //PROGRAMA Nombre /* Descripcion del algoritmo */ /* Declaración de variables: Tipo nombresVariables */ instruccion1; ... instruccionN;</pre> <p>Obs: En lenguajes como PHP, donde el tipo de las variables es dinámico, no hay declaración de tipos. Por eso comentaremos la declaración o no la incluiremos. Podemos decir que es el tipo que esperamos tenga una variable por los valores que serán asignados a dichas variables.</p> |
|----------------------------------|---|--|

## Modularización: Funciones

```
(**
 *Descripción del módulo sin retorno (¿qué hace?)
 *)
MODULO nombre1(tipo1 pf1, tipo2 pf2,...,tipoN pfN)
RETORNA vacío
|Declaración de var internas: tipo var1, var2,...
|instruccion1
|...
|instruccionN
FIN MODULO
```

```
(**
 * Descripción del módulo con retorno (¿qué hace?)
 *)
MODULO nombre2(tipo1 pf1, tipo2 pf2,...,tipoN pfN)
RETORNA tipoDato
|Declaración de var internas: tipo var1, var2,...
|instruccion1
|...
|instruccionN
RETORNA( E)
FIN MODULO
```

(obs.: tipoDato se reemplaza por boolean, integer, float, string, etc.)

### INVOCACIÓN:

Cómo invoco a una fc sin retorno:

nombre1(pa1, pa2, ... , paN)

Cómo invoco a una fc con retorno:

variable ← nombre2(pa1, pa2,..., paN)

pa: parametro actual. Son expresiones (valores, constantes, combinación de operadores y operando)

resultado ← raizCuadrada(144)

```
/**
 *Descripción del modulo sin retorno (¿qué hace?)
 * @param tipo $pf1
 * @param tipo2 $pf2
 * ...
 *
 * @param tipo $pfN
 */
Function nombre($pf1, $pf2,...,$pfN){
/*Declaración de var internas: tipo var1, var2,...*/
    instruccion1;
    ...
    instruccionN;
}

/**
 *Descripción del modulo con retorno (¿qué hace?)
 * @param tipo1 $pf1
 * @param tipo2 $pf2
 * ...
 * @param tipoN $pfN
 * @return tipoDato
 */
Function nombre($pf1, $pf2,...,$pfN){
    instruccion1;
    ...
    instruccionN;
    return E;
}
```

|  |   |   |
|--|---|---|
| <b>Alternativa:<br/>Instrucción IF</b> | <p>SI (condición) ENTONCES<br/>  instruccion1<br/>  instruccionN<br/> FIN SI</p> <p>SI (condición) ENTONCES<br/>  instruccionA1<br/>  instruccionAN<br/> SINO<br/>  instruccionB1<br/>  instruccionBN<br/> <b>FIN SI</b></p> <p>SI (condición) ENTONCES<br/>  instruccionA1<br/>  instruccionAN<br/> OTRO-SI (condiciónB) ENTONCES<br/>  instruccionB1<br/>  instruccionBN<br/> OTRO-SI (condiciónM) ENTONCES<br/>  instruccionM1<br/>  instruccionMN<br/> SINO<br/>  instruccionN1<br/>  instruccionNN<br/> FIN SI</p> | <pre> if (condición){     Instrucción1;     InstrucciónN; }  if (condición){     InstrucciónA1;     InstrucciónAN; } else{     InstrucciónB1;     InstrucciónBN; }  if (condiciónA){     InstrucciónA1;     InstrucciónAN; } else if (condiciónB){     InstrucciónB1;     InstrucciónBN; } elseif(condiciónM){     InstrucciónM1;     InstrucciónMN; } else{     InstrucciónN1;     InstrucciónNN; } </pre> |
|--|---|---|

|   |  |  |
|---|--|--|
| <b>Repetitiva:<br/>Instrucción WHILE</b><br>(ciclos indefinidos - repite 0 a mas veces)     | <b>MIENTRAS (condición) HACER</b><br> instruccion1<br> instruccionN<br><b>FIN MIENTRAS</b>   | <b>while (condición) {</b><br>instruccion1;<br>instruccionN;<br><b>}</b>   |
| <b>Repetitiva:<br/>Instrucción DO..WHILE</b><br>(ciclos indefinidos - repite 1 a mas veces) | REPETIR<br> instruccion1<br> instruccionN<br>MIENTRAS (condición)<br>....  | do{<br>instruccion1;<br>instruccionN;<br>} while (condición);  |
| <b>Repetitiva:<br/>Instrucción FOR</b><br>(ciclos definidos)                                | Entero j , n<br><br>(*LEER(n) o $n \leftarrow 5^*$ )<br><br><b>PARA j &lt;- 0 HASTA n PASO 1 HACER</b><br> instruccion1<br> instruccionN<br><b>FIN PARA</b><br><br><br><b>PARA j &lt;- 1 HASTA n+1 PASO 1 HACER</b><br> instruccion1<br> instruccionN<br><b>FIN PARA</b> | <b>for(\$j=0; \$j&lt;\$n; \$i++){</b><br>instruccion1;<br>instruccionN;<br><b>}</b><br><br>// \$i++<br>// \$i = \$i +1<br><br><b>for(\$j=1; \$j&lt;=\$n; \$i++){</b><br>instruccion1;<br>instruccionN;<br><b>}</b> |

## Traza

Realizar la traza de un programa/modulo equivale a realizar 1 ejecución del programa/módulo.

Nos permite analizar los valores de las variables a medida que se ejecuta el programa/módulo, para determinar si a partir de la entrada, el programa genera la salida correcta.

## Traza programa principal

**Construcción:** Primero escribo el **titulo** “Traza Programa Nombre” y luego armo una tabla con tantas columnas como variables declare en el programa + una columna para la pantalla.

**Ejecución:** Comienzo con la primer instrucción del programa principal y sigo el orden de ejecución del programa.

- Cuando ejecuto una lectura, escribo el valor ingresado por teclado en la variable leída.
- Cuando ejecuto una asignación, resuelvo la expresión del lado derecho y el valor obtenido en la variable. (si la variable ya tenía almacenado un valor, tacho el valor anterior y escribo el nuevo valor)
- Cuando ejecuto una escritura, resuelvo la expresión y el valor obtenido los escribo en la columna pantalla.
- Si en la ejecución aparece la invocación a un módulo, suspendo la ejecución del programa principal en ese punto, y realizo la traza del módulo pasando los valores actuales. Cuando finaliza la ejecución del módulo el valor retornado es utilizado este punto.

## Traza Módulo

**Construcción:** Primero escribo el título “**Traza Módulo Nombre** ( lista de **valores** de parámetros actuales)” y luego armo una tabla con tantas columnas como variables declare en el programa + una columna para la pantalla.

Ejecución: Comienzo con la primer instrucción del programa principal y sigo el orden de ejecución del programa.

EJERCICIOS:

TP2:

- 1) calcule el área de un rectángulo. Solicitar los datos necesarios para el cálculo y mostrar el resultado.

Análisis PROGRAMA PRINCIPAL:

Datos Relevantes = datos de entrada / lectura = float base y altura

Objetivo = Salida / escritura = área del rectángulo

Cómo calculo el área de un rectángulo?  $\text{base} \times \text{altura}$

PROGRAMA PRINCIPAL

(\*Calcula el area de un rectangulo\*)

FLOAT base, altura, superf

ESCRIBIR("base: ")

LEER(base)

ESCRIBIR("altura: ")

LEER(altura)

$\text{superf} \leftarrow \text{base} * \text{altura}$

ESCRIBIR(" la superficie es: ", superf)

FIN PROGRAMA



ejemplo de lo que espero de mi programa: base = 5 y la altura = 6, la superf tiene que ser 30

traza PROGRAMA PRINCIPAL

| base | altura | superf | pantalla                                 |
|------|--------|--------|--|
| 5    | 6      | 30     | base:<br>altura:<br>la superficie es: 30 |

traza PROGRAMA PRINCIPAL

| base | altura | superf | pantalla                                |
|------|--------|--------|---|
| 2    | 3      | 6      | base:<br>altura:<br>la superficie es: 6 |

Modificación del ejercicio:

- 1) Especificar un módulo que dada la base y la altura retorne la superf del rect.
- 2) Especificar un programa calcule el área de un rectángulo. Solicitar los datos necesarios para el cálculo y mostrar el resultado.

- 1) Modulo: Entrada = Parámetros Formales: float base y altura,  
Salida = Retorno: float (superficie del rect.)

- 2) Programa PRINCIPAL

Datos Relevantes = datos de entrada / lectura = float base y altura

Objetivo = Salida / escritura = área del rectangulo

Cómo calculo el área de un rectangulo? utilizo el módulo calcSupRectangulo

Restricción: **Al invocar el módulo** los parámetros actuales deben ser distintos de los formales.

(\*\* Calcula el área del rectangulo\*)

MODULO **calcSupRectangulo( float base, altura )** RETORNO float

float superficie

**superficie ← base \* altura**

**RETORNO(superficie)**

FIN MODULO

```

(** Calcula el área del rectángulo*)
MODULO calcSupRectangulo( float base, altura ) RETORNO float
    RETORNO(base * altura)
FIN MODULO

```

```

PROGRAMA PRINCIPAL
    (*muestra el calculo de la sup. del rectángulo*)
    FLOAT b, h, superf
    ESCRIBIR("base: ")
    LEER(b)
    ESCRIBIR("altura: ")
    LEER(h)
    superf ← calcSupRectangulo(b, h) + 0
    ESCRIBIR("la sup es: ", superf)
FIN PROGRAMA

```

Traza PROGRAMA PRINCIPAL

| b | h | superf | pantalla                          |
|---|---|--------|-----------------------------------|
| 5 | 6 | 30     | base:<br>altura:<br>la sup es: 30 |

Traza calcSupRectangulo( 5, 6 )

| base | altura | superficie | RETORNO |
|------|--------|------------|---------|
| 5    | 6      | 30         | 30      |

- 2) Especificar un programa calcule el área de un rectángulo, y también calcule el área del rectángulo donde la base y la altura está duplicada.

```

PROGRAMA PRINCIPAL
    (*muestra el calculo de la sup. del rectángulo*)
    FLOAT b, h, superf, superfDuplicada
    ESCRIBIR("base: ")
    LEER(b)
    ESCRIBIR("altura: ")
    LEER(h)
    superf ← calcSupRectangulo(b, h)
    superfDuplicada ← calcSupRectangulo( b*2 , h*2)
    ESCRIBIR("la sup es: ", superf)
    ESCRIBIR("la sup duplicada es: ", superfDuplicada )

```

FIN PROGRAMA

Traza PROGRAMA PRINCIPAL

| b | h | superf | superfDuplicada | pantalla   |
|---|---|--------|-----------------|--|
| 2 | 3 | 6      | 24              | base:<br>altura:<br>la sup es 6<br>la sup duplicada<br>es 24 |

traza calcSupRectangulo( 2, 3)

| base | altura | superficie | RETORNO |
|------|--------|------------|---------|
| 2    | 3      | 6          | 6       |

traza calcSupRectangulo( 4 , 6 )

| base | altura | superficie | RETORNO |
|------|--------|------------|---------|
| 4    | 6      | 24         | 24      |

## Ejercicio 13

Dado un número N calcular la siguiente sumatoria:

Ejemplo: para N= 5 la sumatoria resulta ser 9.56 proceso =  $2/1 + 3/2 + 5/3 + 8/4 + 12/5 = 9.56$

Ejemplo: para N= 7 la sumatoria resulta ser 15.68 proceso =  $2/1 + 3/2 + 5/3 + 8/4 + 12/5 + 17/6 + 23/7 = 15.68$

ANALISIS:

MODULO

ENTRADA (param formal): n entero

SALIDA (retorno): float (resultado de la sumatoria)

### PROGRAMA PRINCIPAL

ENTRADA (lectura): n entero → CICLO DEFINIDO = FOR

SALIDA (escritura): float (resultado de la sumatoria)

N= 5 la sumatoria resulta ser 9.56

proceso =  $2/1 + 3/2 + 5/3 + 8/4 + 12/5 = 9.56$

ESPECIFICACIÓN:

### PROGRAMA PRINCIPAL

(\*Este programa realiza una sumatoria con una determinada cantidad de terminos\*)

ENTERO numero, denominador, numerador  
FLOAT suma

suma ← 0

numerador ← 2

ESCRIBIR("Ingrese un numero entero:")

LEER (numero)

PARA denominador ← 1 HASTA numero+1 PASO 1 HACER

    suma ← suma + (numerador/denominador)

    numerador ← numerador + denominador

FIN PARA

ESCRIBIR ("La sumatoria es:", suma)

FIN PROGRAMA

traza PROGRAMA PRINCIPAL

| numero | denominador                | numerador                    | suma  | pantalla   |
|--------|----------------------------|------------------------------|---|--|
| 5      | 1<br>2<br>3<br>4<br>5<br>6 | 2<br>3<br>5<br>8<br>12<br>17 | 0<br>2<br>3.5<br>5.1666<br>7.1666<br>9.5666 | Ingrese un numero entero:<br>La sumatoria es: 9.5666 |

1<5+1 4<6

2<6 5<6

3<6 6<6