UE#03

PROBLEMAS DE SOLUCIÓN DIRECTA

FUNCIONES

Índice

- Taxonomía de problemas
- Problemas como funciones
- Concepto de función
- Especificación de una función. Cláusulas PRE/POST
- Funciones de usuario en Java
- Parámetros

TAXONOMÍA DE PROBLEMAS

- **Solución Directa:** El algoritmo se especifica a través de una fórmula. Se representa con una expresión.
- **Análisis de casos:** El algoritmo tiene que distinguir entre varios casos posibles. Se representa con construcciones sintácticas de selección de alternativas.
- **Recorrido:** El algoritmo requiere realizar una recombinación de cálculos. Se representa con una construcción sintáctica de repetición.

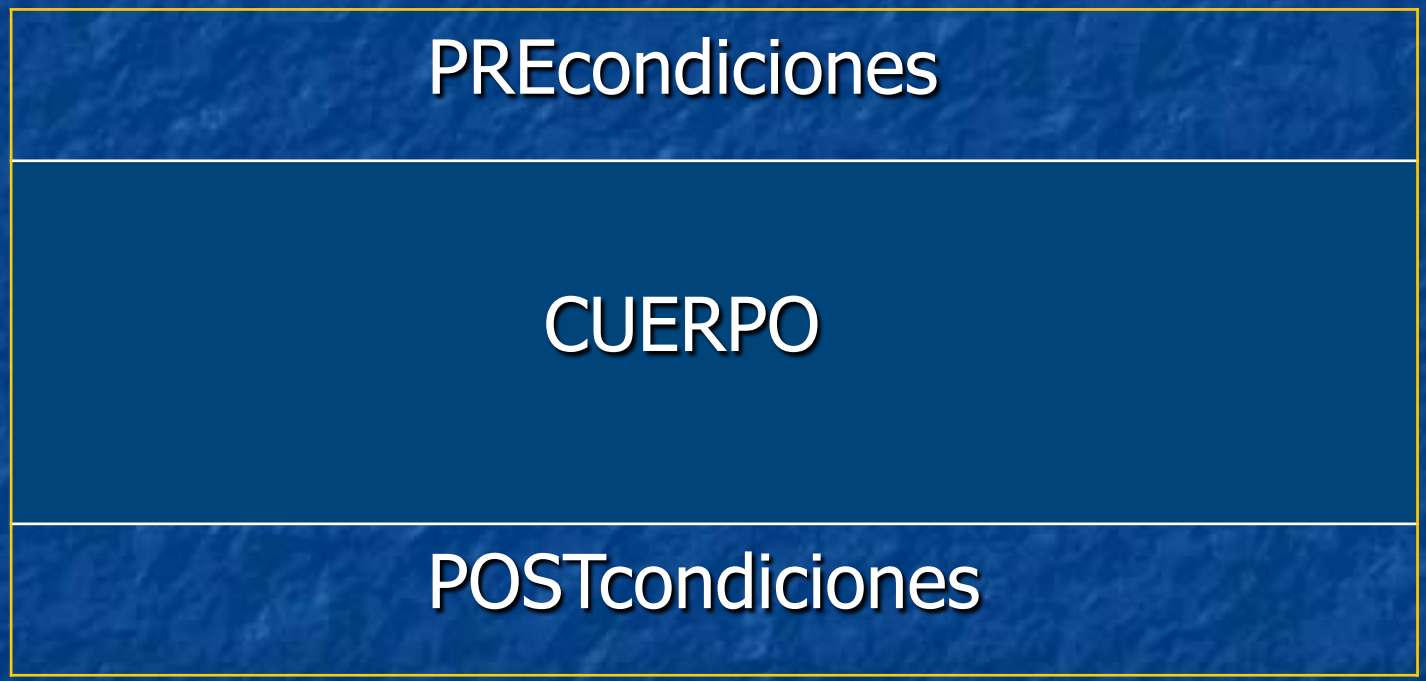
PROBLEMAS COMO FUNCIONES

- Hay que enmarcar, constreñir, acotar el problema
- Hay que identificar los datos y las ligaduras
- Hay que encasillarlo
- Hay que proporcionarle un aspecto. Buscarle un traje.

PROBLEMAS COMO FUNCIONES

- Las funciones ofrecen una sintaxis clara y conocida.
- Tienen un nombre.
- Hacen mención a los datos que manejan.
- Permiten expresar cálculos.
- Ofrecen un resultado.

CONCEPTO DE FUNCIÓN



CONCEPTO DE FUNCIÓN

- Operación sobre un conjunto de datos y sus restricciones, que emite un resultado.
- PRE: restricción que define el conjunto de estados para los que se asegura que el problema va a tener solución.
- POST: aserto que establece la relación entre los datos y el resultado. Lo que enuncia el problema.

ESPECIFICACIÓN DE UNA FUNCIÓN

- Cabecera:
  - Nombre
  - Lista de parámetros
  - Domino del resultado
- Precondición
- Postcondición

ESPECIFICACIÓN DE UNA FUNCIÓN

- La lista de parámetros da nombre a cada parámetro y le asigna un dominio.
- La PRE acota el dominio o describe alguna propiedad, de alguno de los parámetros.
- La POST relaciona todos los parámetros con el resultado.
- Cuanto más restrictiva sea la PRE, menos reutilizable es la función.

EJEMPLOS DE ESPECIFICACIÓN

- Problema1: “Área del triángulo”

- **FUNCIÓN AreaTriangulo (|R base, altura) -----→ |R**
- **PRE: (base>=0) /\ (altura>=0)**
- **POST: resultado = (base * altura) / 2**

- Problema2: “Ser un número múltiplo de otro”

- **FUNCIÓN EsMultiplo (|N m, n) -----→ |B**
- **PRE: (m>0) /\ (n>0)**
- **POST: resultado es cierto si existe un número entero positivo que multiplicado por “n” da “m” y resultado es falso en caso contrario.**

EJERCICIOS DE ESPECIFICACIÓN

- Ejercicio1: “Volumen del cilindro”
- Ejercicio2: “Ser un número par”
- Ejercicio3: “Menor de dos números”

EJERCICIOS DE ESPECIFICACIÓN

- Ejercicio1: “Volumen del cilindro”

- **FUNCIÓN VolumenCilindro1 (|R radio, altura) --→ |R**
- **PRE: (radio>=0) /\ (altura>=0)**
- **POST: resultado = pi * radio * radio * altura**

- **FUNCIÓN VolumenCilindro2 (|R radio, altura) --→ |R**
- **PRE: (radio>=0) /\ (altura>=0)**
- **POST: resultado = AreaCirculo(radio) * altura**

- **FUNCIÓN AreaCirculo (|R radio) --→ |R**
- **PRE: (radio>=0)**
- **POST: resultado = pi * radio * radio**

EJERCICIOS DE ESPECIFICACIÓN

- Ejercicio2: “Ser un número par”

- **FUNCIÓN EsPar (|N m) ------→ |B**
- **PRE: (m>0)**
- **POST1: resultado es cierto si existe un número entero positivo que multiplicado por “2” da “m” y resultado es falso en otro caso.**
- **POST2: resultado es cierto si “m” es múltiplo de “2” y es falso en otro caso.**
- **POST3: resultado es cierto si el resto de la división de “m” entre “2” da “0” y es falso en otro caso**

EJERCICIOS DE ESPECIFICACIÓN

- Ejercicio3: “Menor de dos números”

- **FUNCIÓN Menor2 (|N a, b) → |N**
- **PRE: cierto**
- **POST1: resultado es el menor de los dos valores “a” y “b”**
- **POST2: (resultado <= a) /\ (resultado <= b) /\ (resultado EN {a,b})**

- ¡Ojo! Sin esta última condición bastaría con que resultado fuese 0 para que se cumplieran las dos anteriores, para todo número natural y sin embargo, estaría mal.

FUNCIONES EN Java

- Formato: Cabecera y cuerpo

```
<<TipoRes>> <<Nombre>> (<<ListaParámetros>>)
{
   <<Bloque>>
}
```

- “{“ y “}” comienzo y fin del cuerpo de la función.

FUNCIONES EN Java

- *TipoRes*: Dominio (tipo) del resultado.
- *Nombre*: Identificador que da nombre a la función. Debe empezar por una letra minúscula.
- *ListaParámetros*: Secuencia de pares `TipoParametro NombreParametro` separados por “,”
- *Bloque*: Secuencia de órdenes (sentencias) separadas por “;”

FUNCIONES EN Java

- Sentencia return
- Formato: `return <<expresión>>;`
- Funcionamiento:
  - Se evalúa la expresión. El valor será el resultado de la función.
  - Se termina la función.
- Obligatoria en el bloque de una función.

EJEMPLOS DE CODIFICACIÓN

```
FUNCIÓN AreaCirculo (|R radio) --→ |R
PRE: (radio>=0)
POST: resultado = pi * radio * radio
double areaCirculo (double radio)
{
  return Math.PI * radio * radio;
}

FUNCIÓN VolumenCilindro2 (|R radio, altura) --→ |R
PRE: (radio>=0) /\ (altura>=0)
POST: resultado = AreaCirculo(radio) * altura
double volumenCilindro (double radio, double altura)
{
  return altura * areaCirculo (radio);
}
```

EJERCICIOS DE CODIFICACIÓN

- Ejercicio4: “Área del triángulo”
- Ejercicio5: “Ser un número múltiplo de otro”
- Ejercicio6: “Menor de dos números”

EJERCICIOS DE CODIFICACIÓN

- FUNCIÓN AreaTriangulo (|R base, altura) -----→ |R
- PRE: (base>=0) /\ (altura>=0)
- POST: resultado = (base * altura) / 2
- double areaTriangulo (double base, double altura)
- {
- return (base * altura) / 2.0;
- }

- FUNCIÓN EsMultiplo (|N m, n) -----→ |B
- PRE: (m>0) /\ (n>0)
- POST: resultado es cierto si existe un número entero positivo que multiplicado por "n" da "m" y resultado es falso en caso contrario.
- boolean esMultiplo (int m , int n)
- {
- return (m % n) == 0;
- }

EJERCICIOS DE CODIFICACIÓN

- **FUNCIÓN Menor2 (|N a, b) → |N**
- **PRE: cierto**
- **POST: (resultado <= a) /\ (resultado <= b) /\ (resultado EN {a,b})**
- `int menor2 (int a , int b)`
- `{`
- `  return (a + b - Math.abs (a-b)) / 2;`
- `}`

PARÁMETROS

- **Declaración de una función**: El código completo de la función (cabecera y cuerpo).
- **Invocación de una función**: El nombre seguido de la lista de parámetros actuales.
- **Parámetros formales**: Los que aparecen en la declaración.
- **Parámetros actuales**: Los que aparecen en la invocación.

PARÁMETROS

- FUNCIÓN Cubo (|N numero) → |N
- PRE: cierto
- POST: resultado = numero^3

```
int cubo (int numero)
{
   return numero * numero * numero;
}
```

- Invocaciones válidas:

```
int    dato = 2;
int prueba1 = cubo(dato);
int prueba2 = cubo(3);
int prueba3 = dato + cubo(dato);
int prueba4 = cubo(cubo(dato));
```

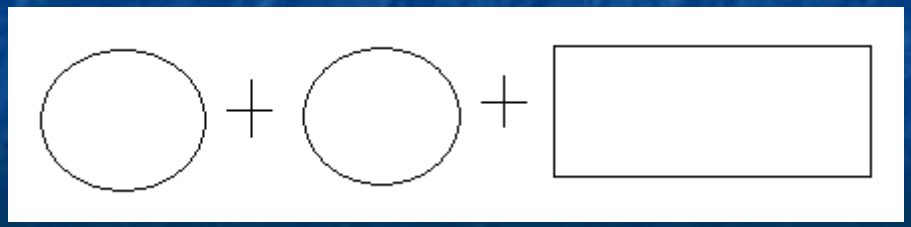
PARÁMETROS

- Parámetro formal de `cubo`:
- `numero`

- Parámetros actuales en cada prueba:
- En la 1: `dato`
- En la 2: `3`
- En la 3: `dato`
- En la 4: `cubo(dato)`

EJERCICIO DE RECAPITULACIÓN

- Problema: “Superficie o área total del cilindro”
- La superficie comprende las dos bases, que son círculos, más el área lateral o área del contorno, que es un rectángulo.



EJERCICIO DE RECAPITULACIÓN

```
FUNCION superficieCilindro (|R radio, altura) →|R
PRE: (radio >= 0) /\ (altura >= 0)
POST: resultado = 2 * AC + AR
DONDE: AC = AreaCirculo (radio)
       AR = AreaRectangulo (LC, altura)
       LC = LongitudCicunferencia (radio)
double superficieCilindro (double radio,
                           double altura)
{
   return 2 *
          areaCirculo (radio) +
          areaRectangulo (longitudCircunferencia (radio),
                          altura);
}
```

EJERCICIO DE RECAPITULACIÓN

```
FUNCION areaRectangulo (|R base, altura) → |R
PRE: (base >= 0) /\ (altura >= 0)
POST: resultado = base * altura
double areaRectangulo (double base, double altura)
{
  return base * altura;
}
FUNCION longitudCircunferencia (|R radio) → |R
PRE: radio >= 0
POST: resultado = 2 * pi * radio
double longitudCircunferencia (double radio)
{
  return 2 * Math.PI * radio;
}
```

VISIBILIDAD

- Todo lo declarado en el cuerpo de la función tiene consideración local.
- Los parámetros formales quedan declarados en la cabecera. También tienen consideración local.
- Lo local es invisible desde fuera.