UE#08

PROBLEMAS DE RECORRIDO

BUCLES

Índice

- Problemas de recorrido.
  - Características.
  - Especificación.
- Taxonomía de problemas.
- Mecanismos de cómputo.
  - Recursividad.
  - Bucles.
- Bucles en Java.
  - *MIENTRAS-HACER*
  - *HACER-MIENTRAS*
  - *FOR*

PROBLEMAS DE RECORRIDO

- Aquellos para los que se necesita recorrer un dominio. Se precisa realizar cálculos para algunos o para todos los elementos del dominio.

- Estos problemas presentan dos frentes:
  - El propio recorrido del dominio.
  - Los cálculos efectuados sobre los elementos.

PROBLEMAS DE RECORRIDO

- Ejemplo: ”Recolectar todos los níscalos de una zona del bosque”
- Sobre el dominio:
  - Saber entrar en esa zona del bosque
  - Saber ir de una seta en otra, sin saltarnos ninguna.
  - Distinguir el níscalo.
  - Saber salir de esa zona del bosque.
- Sobre los cálculos:
  - Cortar el níscalo.
  - Colocarlo en la cesta.

CARACTERÍSTICAS DE LOS PROBLEMAS DE RECORRIDO

- ¿Qué parámetros? La zona del bosque.
- ¿Qué resultado? La cesta con níscalos.
- Hay un filtro (la capacidad de distinguir níscalos).
- Hay un cálculo (el que construye la cesta con los níscalos). El cortar es más bien, literario.
- El recorrido del bosque precisa de una cierta pericia para no perder hongos y para no perdernos.

CARACTERÍSTICAS DE LOS PROBLEMAS DE RECORRIDO

- Para recorrer un dominio hay que contestar a estas 3 preguntas:
  - ¿Con qué elemento empiezo?
  - ¿Cómo avanzo de uno en otro sin dejarme ninguno?
  - ¿Con qué elemento acabo?
- Los filtros sirven para descartar ciertos elementos sobre los que no tienen que recaer los cálculos.

ESPECIFICACIÓN DE LOS PROBLEMAS DE RECORRIDO

- Cuantificadores (macro operadores que actúan sobre muchos operandos).
- Cada cuantificador está asociado a una operación.
- En matemáticas existen:
  - El sumatorio (Σ) asociado a la suma.
  - El productorio (∏), a la multiplicación.
  - El existencial (∃), a la disyunción.
  - Etc .

ESPECIFICACIÓN DE LOS PROBLEMAS DE RECORRIDO

- En general, toda expresión cuantificada se puede expresar como:
-
- $$\zeta\ \alpha \in |Đ \mid \varphi(\alpha) \bullet \rho(\alpha)$$
- Cuyos elementos son:
  - El cuantificador (ζ) que lleva oculta su operación asociada.
  - Su variable ligada (α).
  - El dominio de actuación de la variable ($|Đ$).
  - Un filtro (opcional) sobre los elementos del dominio (φ).
  - Una propiedad sobre los elementos (ρ).

ESPECIFICACIÓN DE LOS PROBLEMAS DE RECORRIDO

- Una expresión cuantificada se lee:

- "Efectúa sucesivamente, con la operación asociada a ζ, la propiedad $\rho(\alpha)$ de los elementos α del dominio $Đ$ que cumplan el filtro $\varphi(\alpha)$"

EJEMPLOS DE ESPECIFICACIÓN

- Problema1: "Ser un número múltiplo de otro"

- **FUNCIÓN EsMultiplo (|N m, n) -----→ |B**
- **PRE: (m>0) /\ (n>0)**
- **POST: res = ALGUN i IN |N+ . (m = i*n)**

- Problema2: "Cuántos 7 tiene un número"

- **FUNCIÓN CuantosSietes (|N n) -----→ |N**
- **PRE: cierto**
- **POST: res = SUM i IN [1,NC(n)] | (Cifra(i,n)=7). 1**
- **DONDE: *NC(m)* es el número de cifras del número "m"**
- ** *Cifra(j,m)* es el dígito de "m" que ocupa el**
- ** "j"-ésimo lugar.**

EJERCICIOS DE ESPECIFICACIÓN

- Ejercicio 1: “Suma de los números de un intervalo”
- Ejercicio 2: “Ser par un número”
- Ejercicio 3: “Suma de los números pares de un intervalo”

EJERCICIOS DE ESPECIFICACIÓN

- FUNCIÓN SumaTodosIntervalo (|N m, n) -----→ |N
- PRE: m >= n
- POST: res = SUM i IN [m,n]. i

- FUNCIÓN EsPar (|N n) -----→ |B
- PRE: n > 0
- POST: res = ALGUN i IN |N+ . (n = 2*i)

- FUNCIÓN SumaParesIntervalo (|N m, n) -----→ |N
- PRE: m >= n
- POST: res = SUM i IN [m,n] | EsPar (i) . i

TAXONOMÍA DE PROBLEMAS

Clasificación y características

CLASE PROBLEMA	CUANTIFICADOR	OPERACIÓN	ELEMENTO NEUTRO	PROPIEDADES
Acumulación	Sumatorio (SUM)	Suma (‘+’)	0	Conmutativa, Asociativa
Acumulación	Productorio (MUL)	Multiplicación (‘*’)	1	Conmutativa, Asociativa
Búsqueda	ExisteAlgún(ALGÚN)	Disyunción (‘\/’)	Falso	Conmutativa, Asociativa
Búsqueda	ParaTodo (TODOS)	Conjunción (‘/\’)	Cierto	Conmutativa, Asociativa
Extremales	Máximo (MAX)	“Mayor2 (x,y)”	-	Asociativa
Extremales	Mínimo (MIN)	“Menor2 (x,y)”	-	Asociativa
Construcción	Filtrar (FIL)	“Construir (x,|Ð)”	“Vacío”	-
Construcción	Aplicar (MAP)	“Construir (x,|Ð)”	“Vacío”	-

PROBLEMAS DE ESPECIFICACIÓN

- Problema1: "Ser número con todos los dígitos iguales"

```
FUNCIÓN TodosIguales (|N n) ---→ |B
PRE: cierto
POST: res = TODOS i IN [1,NC(n)-1] .
                (Cifra(i,n) = Cifra(i+1,n))
```

- Problema2: "Máximo común divisor de dos números"

```
FUNCIÓN MCD (|N m, n) ---→ |N
PRE: (m>0) /\ (n>0)
POST: res = MAX i IN [1,Menor2(m,n)] |
                (Divide (i,m) /\ Divide (i,n)) . i
```

MECANISMOS DE CÓMPUTO

- La *recursividad* es un mecanismo de definición implícita donde en la definición, cabe lo definido. Se especifica con al menos, dos reglas: una en la que se aplica la recursividad y otra en la que no (se termina el proceso).

- Ejemplo del factorial de un número.

$$
n! = \begin{cases} 1 & \text{si } n = 0 \\ n \cdot (n-1)! & \text{si } n > 0 \end{cases}
$$

MECANISMOS DE CÓMPUTO

- Los *bucles* son construcciones sintácticas que permiten la repetición de un bloque de sentencias.
- Su especificación es muy dependiente del lenguaje de programación.

- Ejemplo del factorial de un número en notación cuantificada.

- $n! = \prod \alpha \in [1,n] \bullet \alpha$

MECANISMOS DE CÓMPUTO

- Elementos del esquema general de un bucle.
- 2 Variables:
  - `elemento`: la que contendrá al elemento actual del dominio.
  - `resultado`: la que contendrá el valor calculado por la recombinación.
- 4 Fases:
  - *Iniciación*: de `elemento` (el primero del dominio) y `resultado` (el elemento neutro de la operación de recombinación).
  - *Condición*: expresión lógica sobre lo que queda del dominio.
  - *Recombinación*: lo que se hace en `resultado` con los elementos del dominio.
  - *Modificación*: del `elemento` actual del dominio para pasar al siguiente.

BUCLES EN Java

- **Bucle** *MIENTRAS-HACER*
- Esquema

```
+-----------------------+
| Iniciación;           |
+-----------------------+
| while (Condición)     |
+---+-------------------+
    | Recombinación;    |
    +-------------------+
    | Modificación;     |
    +-------------------+
```

BUCLES EN Java

- **Bucle** *MIENTRAS-HACER*

- `while (<<expresión_booleana>>)`
- `    <<Bloque de sentencias>>;`

- Funcionamiento: Se evalúa la expresión booleana. Si el resultado es falso, se ignora el bloque y se sale del bucle. Si el resultado es cierto, se ejecuta el bloque y vuelve a evaluarse la expresión booleana.

BUCLES EN Java

- "Factorial de un número"

```
int factorial (int n)
{
  int resultado = 1;
  int i = 1;
  while (i <= n)
  {
    resultado = resultado * i;
    i = i + 1;
  }
  return resultado;
}
```

BUCLES EN Java

- **Bucle** *HACER-MIENTRAS*
- Esquema

```
+-------------------------+
| Iniciación;             |
+-------------------------+
| do                      |
+--+----------------------+
      | Recombinación;    |
      +-------------------+
      | Modificación;     |
+--+----------------------+
| while (Condición);      |
+-------------------------+
```

BUCLES EN Java

- **Bucle** *HACER-MIENTRAS*

```
do
    <<Bloque de sentencias>>;
while (<<expresión_booleana>>);
```

- Funcionamiento: Se ejecuta el bloque Y se evalúa la expresión booleana. Si el resultado es falso, se sale del bucle. Si el resultado es cierto, vuelve a ejecutarse el bloque y la expresión booleana.

BUCLES EN Java

- "Factorial de un número"

```
int factorial (int n)
{
  int resultado = 1;
  int i = 1;
  do
  {
    resultado = resultado * i;
    i = i + 1;
  }
  while (i <= n)
  return resultado;
}
```

BUCLES EN Java

- **Bucle** *FOR*
- Esquema

```
+-----------------------------------------+
| Iniciación; (solo del resultado)        |
+-----------------------------------------+
| for (Iniciación; Condición; Modificación)  |
+---+-------------------------------------+
    | Recombinación;                      |
    +-------------------------------------+
```

BUCLES EN Java

- **Bucle** *FOR*

```
for (<<declaración/iniciación>>;
      <<expresión_booleana>>;
      <<modificación>>)
   <<Bloque de sentencias>>;
```

- Funcionamiento: Como el del bucle *MIENTRAS-HACER*.

BUCLES EN Java

- "Factorial de un número"

```
int factorial (int n)
{
   int resultado = 1;
   for (int i = 1; i <= n; i++)
     resultado = resultado * i;
   return resultado;
}
```

EJERCICIOS DE BUCLES

```
int sumaTodosIntervalo (int m, int n)
 {
   int res = 0;
   for (int i = m; i <= n; i++)
     res = res + i;
   return res;
 }

int sumaParesIntervalo (int m, int n)
 {
   int res = 0;
   for (int i = m; i <= n; i++)
     if (esPar (i))
       res = res + i;
   return res;
 }
```

EJERCICIOS DE BUCLES

```
boolean todosIguales (int n)
{
  boolean todosCumplen = true;
  int i = 1;
  int totalCifras = numeroCifras(n);
  while ((i < totalCifras) && todosCumplen)
    if (cifraI(i,n) != cifraI(i+1,n))
      todosCumplen = false;
    else
      i = i + 1;
  return todosCumplen;
}
```

EJERCICIOS DE BUCLES

```
int mcd (int uno, int otro)
{
   int m = uno;
   int n = otro;
   while (m != n)
      if (m > n)
         m = m - n;
      else
         n = n - m;
   return m;
}
```