Programació I - Tema 4 - Composicions algorítmiques seqüencials, alternatives i iteratives

Grau en Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica
Curs 18-19

Programació I - Tema 4 - Composicions algorítmiques seqüencials, alternatives i iteratives

4.1. Composició seqüencial

- En el procés de disseny d'un programa, el problema (S) es descompon en una o varies sentències (o instruccions) de forma seqüencial (una darrera l'altra $S \equiv S_0 S_1 S_2 \ldots S_n$)

$$\begin{array}{ccc} & & S_0 \\ & & S_1 \\ & & S_2 \\ \mathsf{S} & \equiv & S_3 \\ & & \ldots \\ & & \\ & & S_n \end{array}$$

- S_k sempre s'executa després de S_{k-1}
- L'execució és incondicional: sempre es fa en l'ordre establert i cada S_k s'executa només un cop.

Exemple: Intercanvi de dues variables (IntercanviV0.java) I

```
import java.util.Scanner;
public class IntercanviV0 {
  public static void main (String[] args) {
    int x, y, tmp;
    Scanner sc;
    sc = new Scanner(System.in);
    System.out.println("X?"); // entrada per teclat
    x = sc.nextInt();
    System.out.println("Y?");
    y = sc.nextInt();

    //Afegir el codi que fa l'intercanvi




    System.out.println("X: " + x); // sortida per pantalla
    System.out.println("Y: " + y);
  }
}
```

Assignacions

- Pre-increments i post-increments:
  - `x = ++nomVariable;` és equivalent a dues sentències d'assignació (S1; S2;):

    ```
    nomVariable = nomVariable + 1;
    x = nomVariable;
    ```

  - `x = nomVariable++;` és equivalent a dues sentències d'assignació (S1; S2;):

    ```
    x = nomVariable;
    nomVariable = nomVariable + 1;
    ```

Problemes sobre seqüencial proposats I

- Donats dos enters x, y realitzeu l'intercanvi dels seus valors sense utilitzar cap variable temporal.
- Donat un valor de temps en segons, calculeu el nombre d'hores, minuts i segons que representa aquest valor.

4.2 Composició alternativa

if .. else

- Es parla de composició alternativa quan en l'anàlisi d'un problema cal executar unes o altres sentències segons si una condició (expressió booleana) és certa o no.

```
if (<condició>) { <sentènciesCondicioVeritat> }
else { <sentènciesCondicioFalse> }
```

  - `<condició>` = tota expressió que avalua a tipus booleà

4.2 Composició alternativa I

if

- `if (<condició>) { <sentènciesVeritat> }`
- `<condició>` = tota expressió que avalua a tipus booleà

Exemple 1 de composició alternativa I

- Si l'edat que introdueix l'usuari és major o igual a 18 s'escriu en la pantalla "Ja pots entrar a la universitat", en cas contrari, s'escriu "Encara no pots entrar a la universitat". Finalment, i en qualsevol cas, s'escriu "A reveure!"

```
import java.util.Scanner;
public class AlternativaBifurcacioV0 {
  public static void main (String[] args) {
    int edat; Scanner sc;

    sc = new Scanner(System.in);
    System.out.println("Quina edat tens?");
    edat = sc.nextInt();

    //Afegir codi aqui



  }
}
```

Exemple 2 de composició alternativa I

- Si l'edat que introdueix l'usuari és major o igual a 18 s'escriu en la pantalla "Ja pots entrar a la universitat", en cas contrari no s'escriu res. Finalment, i en qualsevol cas, s'escriu "A reveure!"

```
import java.util.Scanner;

public class AlternativaSimple {
  public static void main (String[] args) {
    int edat;
    Scanner sc;

    sc = new Scanner(System.in);
    System.out.println("Quina edat tens?");
    edat = sc.nextInt();

    // Afegir codi aqui



  }
}
```

4.2 Composició alternativa

if ... else if ... else

- S'avaluen en ordre les condicions i s'executen les sentències de la primera condició que és veritat (true). Si cap de les condicions són veritat, s'executen les sentències de l'últim else.

```
if (<condició_1>) {
<sentènciesCondicio_1Veritat>
}
else if (<condició_2>) {
<sentènciesCondicio_2Veritat>
}
...
else if (<condició_n>) {
<sentènciesCondicio_nVeritat>
}
else {
<sentènciesCapVeritat>
}
```

  - <condició_1>, <condició_2>,... <condició_n> = tota expressió que avalua a tipus booleà

Exemple 3 de composició alternativa I

- Si l'edat que introdueix l'usuari és major o igual a 18, s'escriu en la pantalla "Pots cursar estudis universitaris", si està compresa entre 14 i 17, s'escriu "Pots cursar estudis secundaris", si es menor o igual a 13 s'escriu "Pots cursar estudis primaris".

```
import java.util.Scanner;
public class AlternativaMultipleAniuament {
  public static void main (String[] args) {
    int edat; String resposta;

    Scanner sc = new Scanner(System.in);
    System.out.println("Quina edat tens?");
    edat = sc.nextInt();


    //Afegir codi aqui


    System.out.println("Pots cursar estudis " + resposta);
  }
}
```

Alternativa Abreujada I

- (<condició>)?<sentènciesV>:<sentènciesF>

```
import java.util.Scanner;
public class AlternativaAbreujada {
  public static void main (String[] args) {
    int edat; String resposta;
    Scanner sc = new Scanner(System.in);

    System.out.println("Quina␣edat␣tens?");
    edat = sc.nextInt();

    resposta = (edat >= 18) ? "Sí" : "No";

    System.out.println(resposta + "␣pots␣entrar␣a␣la␣
        universitat");
    System.out.println("A␣reveure!");
}
```

Múltiple - instrucció dedicada (switch)

- Implementa casos de comparacions per igualtat. Només és pot utilizar si l'expressió és un char, un byte, un short, un int o un String. La sintaxi és:
- `switch (<expressió>) {`

```
    case <valor1>:  <sentències1>; [break;]
    case <valor2>:  <sentències2>; [break;]
    <...>
    case <valorN>:  <sentènciesN>; [break;]
    default:  <sentènciesCapIgual>;
}
```

- S'avalua en ordre:
  - Si `<expressió>` és igual a `<valor1>` se executen les `<sentències1>` i el break fa que l'execució continui per la sentència que hi ha després del switch (finalitza el switch).
  - Si `<expressió>` no es igual a `<valor1>` s'avalua si `<expressió>` és igual a `<valor2>` i se executen les `(<sentències2>)` i el break fa que l'execució continui per la sentència que hi ha després del switch (finalitza el switch).
  - Així succesivament...
  - Si `<expressió>` NO és igual a cap valor de cap `<case>`, s'executen les `<sèntenciesCapIgual>`.

Múltiple - instrucció dedicada - exemple I

```
import java.util.Scanner;
public class AlternativaMultipleInstruccio {
  public static void main (String[] args) {
    int edat; String resposta;

    Scanner sc = new Scanner(System.in);
    System.out.println("Quina edat tens?");
    edat = sc.nextInt();

    switch(edat) {
      case 12:  resposta = "1r ESO"; break;
      case 13:  resposta = "2n ESO"; break;
      case 14:  resposta = "3r ESO"; break;
      case 15:  resposta = "4t ESO"; break;
      case 1: case 2:
          resposta = "escola bressol"; break;
      case 3: case 4: case 5:
          resposta = "llar d'infants"; break;
```

Múltiple - instrucció dedicada - exemple II

```
        default:
          if (edat >= 6 && edat <= 11) {
                  resposta = "primària";
          } else if (edat > 15) {
              resposta = "estudis␣superiors";
          } else {
              resposta = "res";
          }
      }
      System.out.println("Pots␣cursar␣" + resposta);
    }
}
```

Simplificació de sentències alternatives I

Són necessàries aquestes sentències alternatives? Es pot fer el mateix amb una assignació?

```
//Exemple1
if (x > 3)
  mesgran3 = true;
else
  mesgran3 = false;

//Exemple2
if (trobat)
  perdut = false;
else
  perdut = true;

//Exemple3
if (trobat == true)
   System.out.println("Trobat");
else
   System.out.println("Perdut");

//Exemple4
boolean hies, trobat, j;
int a;
```

Simplificació de sentències alternatives II

Són necessàries aquestes sentències alternatives? Es pot fer el mateix amb una assignació?

```
if (hies)
  if (trobat && a > 5)
    j = true;
  else
    j = false;
else
  j = true:
```

Problemes I

1. Feu un programa que indiqui si el número enter entrat per l'usuari és parell.
2. Donats tres números enters entrats per l'usuari, implementeu un programa que els escrigui amb ordre descendent.
3. Feu un programa que donat el número de més [1..12] indiqui quants dies té (28,30,31) ignorant els anys de traspàs.
4. Dissenyeu un programa que donada una nota acadèmica de [0..10] amb un decimal entrada per l'usuari, indiqui el seu equivalent en text (*Insuficient*, *Suficient*, *Bé*, *Notable*, *Excel·lent*).
5. Modifiqueu el programa anterior per a que indiqui: en cas de *Notable*, si és *Alt* o *Baix*; i en cas d'*Insuficient*, la nota exacta.
6. Codifiqueu un programa que pregunti quin tipus de plat vol escollir l'usuari (*Primers*, *Segons* o *Postres*) i a continuació li ofereixi 5 opcions diferents (ex. *Sopa*, *Amanida*, *Pasta*...). Al final el programa ha d'indicar l'opció amb una frase tipus *"Com a* $< tipus_{plat} >$ *heu triat* $<$ *plat* $>$*"*.

4.3. Composició iterativa (bucles)

- Quan s'ha de repetir l'execució d'una acció o conjunt d'accions segons si es compleix una propietat (o condició booleana), s'utilitza la composició iterativa.
- **Sintaxi bàsica (amb sentència while):**
  - S_i (Sentències inicials)
    while (condició booleana) {
        < S (sentències) >
    }
    S_f (Sentències finals)

Composició iterativa

- **Mecanisme:**
  - Inicialment s'executen les sentències S_i, després s'avalua l'expressió booleana del while. Si és certa s'executen les sentències S. Es torna a analitzar la condició després d'executar S.
  - Les accions del cos del while (S) només s'executen si la condició booleana és certa.
  - Quan la condició booleana és falsa es passen a executar les sentències finals de després del mentre (S_f)
- **Correctesa:** S'han de garantir l'estat inicial abans del while, la condició booleana de control del while i les sentències internes per a que la composició iterativa acabi en algun moment.

4.3. Composició iterativa

- Quan s'ha de repetir l'execució d'una acció o conjunt d'accions n vegades, s'utilitza la composició iterativa.
- **Sintaxi bàsica (amb sentència for):**
  - S_i `(Sentències inicials)`

```
for (int idx = 0 ; idx < n ; idx++) {
    < S ( sentències) >
}
```

S_f `(Sentències finals)`

Exemples: Analitzar els següents programes I

```
public class Escriu100Enters {

  public static void main (String[] args) {
    int num;

    num = 1;
    while (num <= 100) {
      System.out.println("␣"+num+"␣");
      num = num + 1;
    }
  }
}

public class Escriu100EntersInf {
  public static void main (String[] args) {
    int num;

    num = 1;
    while (num <= 100) {
      System.out.println("␣"+num+"␣");
    }
  }
}
```

Exemples: Analitzar els següents programes II

```
public class CondicionsInicialsIterativa {
  public static void main (String[] args) {
    int num = 0;
    /* I num = 3? i num = -1? i num = 4? */
    while (num != 0) {
      num = num - 2;
    }
  }
}
```