Iteration

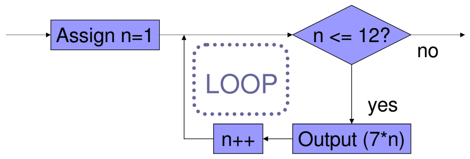
Hussein Suleman
UCT Dept of Computer Science
CS115 ~ 2004

Problem

- Output the 7x table.

What is Iteration?

- Executing the same task or set of statements multiple times
  - e.g., print the 7x table (from 1 to 12)



Counter-controlled Loops

- Loops that usually execute for a fixed number of times
- A special counter variable is used to control the loop and may be referred to within the loop
- Java provides the "for" statement

The "for" statement

```
for ( initialisation;
      condition;
      increment )
{
   statements …
}
```

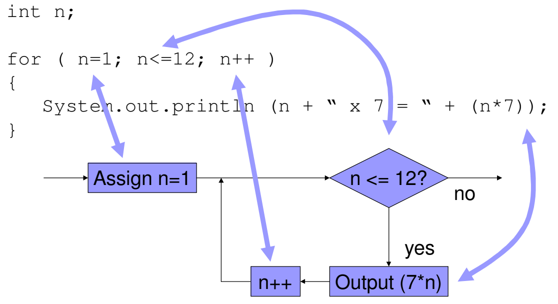
Example Usage

```
int n;

for ( n=1; n<=12; n++ )
{
   System.out.println (n + " x 7 = " + (n*7));
}

Output:
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
...
```

Flowchart vs Java

UCT-CS

```
int n;

for ( n=1; n<=12; n++ )
{
   System.out.println (n + " x 7 = " + (n*7));
}
```



Additional "for" syntax

UCT-CS

- We can define a variable in the initialisation section, which is local to the body of the loop
  - `for ( int i=1; i<=10; i=i+1 )`
- Multiple comma-separated expressions can appear in the "increment" section, even decrements
  - `for (int i=10; i>0; i--)`
  - `for (int i=1,j=7; i<=12; i++,j+=7)`

Problem revisited

UCT-CS

- Output the `n x table` for any integer value of n. Encapsulate this functionality into a class, with a method called **printNTimesTable**, taking *n* as a parameter.

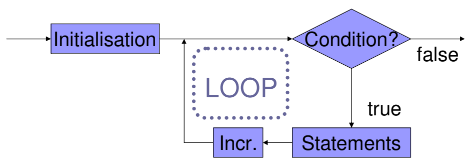
Solution?

UCT-CS

```
class Kiddies
{
   public void printNTimesTable ( int n )
   {
      for ( int i=1; i<=12; i++ )
         System.out.println
            (i+" x "+n+" = "+(n*i));
   }
}
```

General Semantics of "for"

UCT-CS



Problem

UCT-CS

- Find the product of the integers from 1..n, corresponding to n!.

Problem

UCT-CS

- Calculate a^b using a **for** loop, assuming that *a* is a float and *b* is an integer.

Nesting of statements

UCT-CS

- **for** and **if** are both statements, therefore they can each appear within the statement body
  - for (int I=1; I<=10; I++)
    { if (a<b) max=b; }
  - if (a<b)
    { for (int I=1; I<=10; I++) }
  - for (int I=1; I<=10; I++)
      for (int j=1; j<=10; j++)

Nested loops

UCT-CS

- Where a task is carried out multiple times and a subtask within that is carried out multiple times
- Example:
  - Draw a triangle of arbitrary height on the screen, such as:

```
*
**
***
****
```

Problem

UCT-CS

- Write programs to generate (on the screen) the following triangles of user-specified height:

```
   *        *      ****
  **       ***      ***
 ***      *****      **
****     *******      *
```

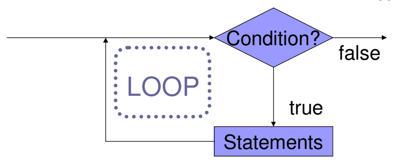
Condition-controlled Loops

UCT-CS

- If we do not know the number of iterations a priori (in advance), we can use a condition-controlled (or event-controlled) loop - where the loop executes while a condition is true
- Two statements:
  - while (*<condition>*) { *<statements>* }
  - do { *<statements>* } while (*<condition>*)

“while” Example

UCT-CS



```
int sum = 0;
int num = Keyboard.readInt ("Enter a no: ");
while (num != 0)
{
   sum = sum + num;
   num = Keyboard.readInt ("Enter a no: ");
}
```

Problem

UCT-CS

- Approximate the logarithm (with a base of 10) of an integer using repeated division.

Problem

UCT-CS

- Approximate the logarithm (with a base of 10) of an integer using repeated division.
- Design a user interface where the user can continue to ask for logarithms until a value of 0 is supplied.

Menus

UCT-CS

- A menu is a list of choices presented to the user, with the means to select one
- Example:

```
Souper Sandwich Menu
   1. Chicken, cheese and chilli sauce
   2. Chicken and cheese
   3. Cheese
   4. Exit Program
Enter the sandwich number:
```

Menu Example

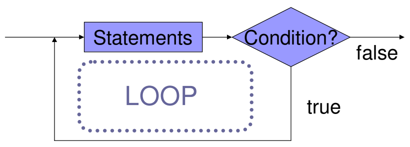
UCT-CS

```
Menu souper = new Menu ();
souper.print ();    // output options
int choice = Keyboard.readInt (); // get selection
while (choice != 4) // continue until exitted
{
   System.out.println (); // leave a line
   switch (choice)  // output ingredients
   {
      case 1 : System.out.println ("Add chilli");
      case 2 : System.out.println ("Add chicken");
      case 3 : System.out.println ("Add cheese");
   }
   souper.print (); // output options
   choice = Keyboard.readInt (); // get selection
}
```

“do..while” statement

UCT-CS

- When the “loop body” is going to be executed at least once, we can check the condition after the loop (instead of before)



“do..while” Example

UCT-CS

```
Menu souper = new Menu ();
int choice;
do {
   souper.print ();       // output options
   choice = Keyboard.readInt (); // get selection
   System.out.println (); // leave a line
   switch (choice)        // output ingredients
   {
      case 1 : System.out.println ("Add chilli");
      case 2 : System.out.println ("Add chicken");
      case 3 : System.out.println ("Add cheese");
   }
} while (choice != 4)     // continue until exitted
```

Problem

UCT-CS

- Find the reverse of an integer.
- For example, the reverse of the integer 12345 is 54321 and the reverse of 98 is 89. Use only integer manipulations - do not convert the number to a String.

Infinite Loops

UCT-CS

- Loops where the condition is always true
- Example:

```
while (true)
{
   System.out.println ("Wheeee!");
}

do { … } while (true);
for ( int i=1; i<10; ) { … }
```

break

UCT-CS

- exits immediately from a loop
- Example:

```
int i = 0;
while (true)
{
   i++;
   System.out.println (i);
   if (i == 10) break;
}
```

continue

UCT-CS

- immediately starts next iteration
- Example:

```
for ( int i=0; i<=10; i++ )
{
   if (i % 3 == 0)
      continue;
   System.out.println (i);
}
```

Selecting Loops

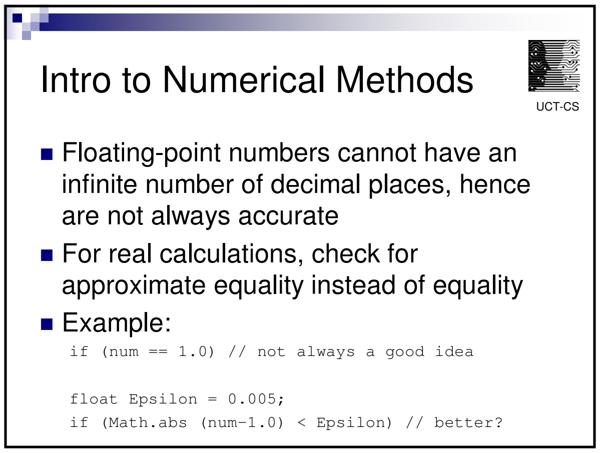
UCT-CS

- General Rules:
  - When you know the number of iterations, use a “for”
  - When the iterations depend on a condition,
    - use a “do..while” if the loop must execute at least once
    - otherwise, use a “while”

Converting Loops

UCT-CS

- How do we write the equivalent of
  - “while” using “for”
  - “do..while” using “for”
  - “for” using “while”
  - “do..while” using “while”
  - “for” using “do..while”
  - “while” using “do..while”

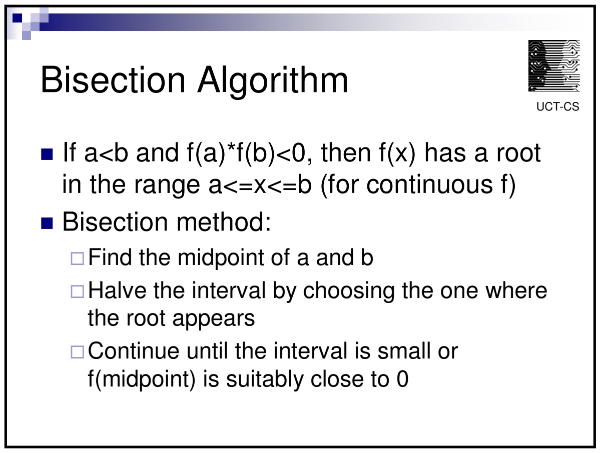


Intro to Numerical Methods

UCT-CS

- Floating-point numbers cannot have an infinite number of decimal places, hence are not always accurate
- For real calculations, check for approximate equality instead of equality
- Example:

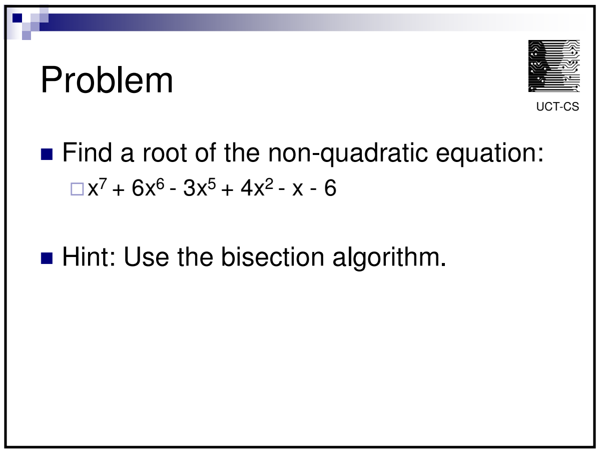
```
if (num == 1.0) // not always a good idea

float Epsilon = 0.005;
if (Math.abs (num-1.0) < Epsilon) // better?
```



Bisection Algorithm

UCT-CS

- If a<b and f(a)*f(b)<0, then f(x) has a root in the range a<=x<=b (for continuous f)
- Bisection method:
  - Find the midpoint of a and b
  - Halve the interval by choosing the one where the root appears
  - Continue until the interval is small or f(midpoint) is suitably close to 0



Problem

UCT-CS

- Find a root of the non-quadratic equation:
  - $x^7 + 6x^6 - 3x^5 + 4x^2 - x - 6$
- Hint: Use the bisection algorithm.