03.10.2011

Supporting Material for Teachers

Explaining basic computational concepts II

indices

Loops

What is a 'While- loop'?

A while-loop is a sequence of instructions that is continually repeated until a certain condition is reached. In other words, it is a facility offered to the programmer to flow control and to instruct the computer to repeat a task upon a given condition.

How does a 'While- loop' look like?

The while construct allows the repetitive execution of a list of commands, as long as the condition controlling the while loop evaluates to true.

If condition evaluates to false another block of commands (instruction N) outside the body of the while- loop is being executed.

A while loop takes the following form:

```
While (condition){
instruction 1
instruction 2
....
}

instruction N
```

From theory into practice...

```
While (free space){
 a person enters
}
entrance denied
```

Imagine that a big venue takes place. A lot of people queued to enter but the space is limited.

What is this script for?

The script is for allowing the entrance while there is free space. If there is not free space the entrance is denied.

What will happen if the condition is true?

While there is free space (condition is true) a person enters. The condition is then being tested again.

What will happen if the condition is false?

If the condition is false (meaning: there is not free space), the entrance is denied.

```
bottles= 50
While (bottles ≠ 0){
grab a bottle
place it on the shelf
bottles= bottles -1
}
```

What is this script for?

This script is used for placing 50 bottles on a shelf. While bottles are not zero (meaning: bottles exist), a bottle is grabbed and placed on the shelf and the total number of bottles is reduced by one. So, the second time the number of the bottles will be 49, the condition will be true and a bottle will be grabbed and placed on the self. The bottles will be 48.Again the condition will evaluate to true; a bottle will be grabbed and placed on the shelf and the total number of bottles will be 47…

When the bottles will be zero nothing happens and the script terminates.

Note: bottles ≠ 0 means that the number of bottles is not zero.

```
a = 5
b = 0
While ( a > 2) {
a = a - 4
}
b = a + 1
```

What will be the value of a and b after the execution?

Initially **a** equals to **5** and **b** equals to 0. The condition is being tested evaluating to **true** due to the fact that **5 is bigger than 2**; The commands in the loop are executed. Thereby, **a** takes a new value. Now, **a** equals to 5– 4= **1**. The flow control goes again to the while– condition. This time the condition is **false** due to the fact that **1 is not bigger than 2.** The commands in the while– loop are not executed. The flow control goes **outside** the body of the loop. The command **b=a+1** is being executed. **b** takes a new value. It is 1 +1 = **2**. **So the value of a is 1 and the value of b is 2.**

A switch

What is a switch?

Switch is a programming construct consist of commands and multiple conditions. The construct includes **cases** related to the potential values of a condition, which may be more general than true and false, and connects cases series of statements, which in the simplest case may be just one statement, to be executed for each potential value. Notably, the **switch** statement is passing control to **one** of the **case** statements within its body.

How does a switch look like?

It takes the following algorithmic form:

```
switch (expression){
   case '1': statement 1;
   case '2': statement 2;
   case '3': statement 3;
   case '4': statement 4;
   default: statement 0;
}
```

And how does it operate?

Once the general conditional expression evaluates to one of the corresponding case, the corresponding statement (or block of statements) are executed accordingly. If *expression* does not match any *corresponding case*, control is transferred to the *statement*(s) that follow the optional **default** label. If there is no **default** label, control is transferred outside the **switch**.

From theory to practical examples...

```
Read budget
switch (type of budget){
   case (budget >= 3000): journey 1
   case (2000 <= budget< 3000): journey 2
   case (1000 <= budget< 2000): journey 3
   case (600  <= budget< 1000): journey 4
   case (300  <= budget< 600) : journey 5
   case (budget < 300): no suggestion
   }
```

Imagine that the computer asks the user to insert his/her budget. In accordance to the budget the computer suggests a destination for travelling.

What will happen if the user's budget is 3500 euro?

The computer will suggest journey 1.

What will happen if the user's budget is 2500 euro?

The computer will suggest journey 2.

What will happen if the user's budget is 1000 euro ?

The computer will suggest journey 3.

What will happen if the user's budget is 500 euro?

The computer will suggest journey 4.

What will happen if the user's budget is 100 euro?

The computer will suggest nothing.

```
Read day
switch (day){
case (day= Monday): schedule 1
case (day= Tuesday): schedule 2
case (day= Wednesday): schedule 3
case (day= Thursday): schedule 4
case (day= Friday) : schedule 5
case (day= Saturday): schedule 6
Case (day= Sunday): schedule 7
}
```

Imagine that the computer acts as reminder of the user's daily schedule. The user has to only enter the day and the schedule is appeared.

What will happen if the day is Monday?

Schedule 1 is appeared.

What will happen if the day is Tuesday?

Schedule 2 is appeared.

What will happen if the day is Friday?

Schedule 5 is appeared.

What will happen if the day is Sunday?

Schedule 7 is appeared.

Can you think of more examples like these two?

Let's bring all the concepts together

A script can include simple commands as well as a combination of programming constructs. This may guarantee more efficient and elegant solutions. A conditional statement for example can be place in the body of a 'While- loop'.

```
While ( condition1 ) {
  command 1
  command 2
  IF  (condition 2) then {
       command 3
  }
  else {
        command 4
  }
}
```

In a similar way a switch can be placed in the body of a 'While– loop'.

```
While ( condition1 ) {
  command 1
  command 2
  switch (expression){
  case '1': command 1;
  case '2': command 2;
  case '3': command 3;
  case '4': command 4;
  default: statement 0;
  }
}
}
```

Practical examples

```
While ( people in the queue) {
    IF  (free space) then
       a person enters the place
    else {
        entrance denied
        door closes
      }
}
```

Imagine that a big venue takes place and people are in a queue waiting to enter the place. The space is limited. Note: Assume that the denying of the entrance means that the people in the queue will stop waiting.

What will happen if there are 10 people left in the queue and there is room for only 3 people?

The three first individuals in the queue will enter the place. In the fourth check the while condition will be true due to the fact that there are still people in the queue. The IF– condition will be false and the **else-branch** will be executed. So, the entrance will be denied and the door will close. In the fifth check the While– condition will be false and the script will be terminated.

```
Read period
While (period= Winter) {
read day
switch (day){
  case (day= Monday): schedule 1
  case (day= Tuesday): schedule 2
  case (day= Wednesday): schedule 3
  case (day= Thursday): schedule 4
  case (day= Friday) : schedule 5
  case (day= Saturday): schedule 6
  case (day= Sunday): schedule 7
  }
 }
```

What will happen if it is Winter ?

The user is being asked about the day.

What will happen if the period is Winter and the day is Tuesday?

Schedule 2 is appeared.

What will happen if the period is Summer ?

Nothing will happen.

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.