**Algorithms and Pseudo-Code**

**COEN 10 -- Prof. Silvia Figueira**

**What is a computer program?**   
A program is a set of instructions that a computer follows to transform a set of input values into a set of output values. Programs are based on two main concepts:

* Variables -- Used to hold input, output, and intermediary or auxiliary values, which change as the input is processed and the ouput is generated.
* Statements -- Constitue the instructions given to the computer to manipulate these values, i.e., to process the input and generate the output.

**Variables**   
Variables constitute the most basic programming concept. Data values are stored in variables in memory and may change as the program executes. Variables have a unique name and can be assigned different values during the execution of the program.

**Example**

input in

out = in \* 2

output out

In the example above, we have 2 variables, in and out. Variable in receives the input, which is then processed to generate the output, which is held by variable out, which is the output of the program. This program basically outputs twice the value of the input.

**Statements**  
There are two kinds of statements: Flow Statements and Basic Statements. Basic statements represent simple instructions for the computer:

* Assignment statements, used to change the value of variables
* Function calls, used to invoke a procedure, for example, to interact with the operating system

**Example**

input in

out = in \* 2

output out

In the example above, the first and the third lines represent function calls, in which the operating system provide the progran with an input value and outputs the value provided by the program, respectively. The second line represents an assignment, in which the result of the multiplication is assigned to variable out.

Flow statements determine the flow of execution that a computer follows to execute a task, i.e., to transfom a set of input values into a set of output values.

The examples in the previous lesson exposed the four basic elements of programming.

* sequentiality -- instructions are executed in sequence
* decision making -- according to some condition, differents paths may be taken
* repetition -- instructions are executed in a loop
* concurrency -- instructions are executed concurrently

These elements constitute the four types of flow statements that determine the control flow of any program:

* Sequential statements
* Conditional statements
* Loop statements
* Concurrent statements

A program is a combination of basic and flow statements, and programming is the art of combining statemenst in a correct and efficient way. Statements can be combined in two ways: sequentially or nested, in which case basic statements and flow statements can be nested within flow statements.

Every program is a combination of these statements, which step by step, or steps by steps in case of concurrency, transforms a set of input values in a set of output values. An efficient program will receive a set of input values and, using the minimum amount of resources, which could be time, space in memory, energy, etc., will process the input and generate the correct set of output values.

In practice, programming is divided in two activities, developing an algorithm and coding. An algorithm basically defines how the statements will be combined. It is done in a high-level language, called pseudo-code, which can then be coded, or translated into any programming language.

The algorithm, i.e., the combination of statements, determine the flow of instructions for the computer to follow.

Therefore, when writing an algorithm the first thing to do is to identify the input and output. Then, you start writing the pseudo-code, which use **variables** to hold the input, output, and auxiliar values, which change values as the input is processed and the output is generated.

To write pseudo-code, we need to specify how each statement is represented.

**Assignment Statements**

Assignment statements are used to assign values to variables. The character '=' is used to assign the value on the right to the variable on the left. The value on the right may come from different sources: a literal, a variable, or the evaluation of an expression, which can have literals and/or variables.

variable = value

variable = variable

variable = expression

**Loop Statements**

**Counting Loops**  
A counting loop will assign an initial value to a variable and increment (or decrement) this value in each iteration of the loop, until the final value is reached. Note that the value of the variable change in each iteration of the loop, in which the actions specified in the loop are executed.

for i = x to y

sequence of statements

end for

**Conditional Loops**  
A conditional loop will execute the actions specified in the loop body, while the condition specified is true.

In a while loop, the condition is evaluated before each iteration of the loop. Depending on the condition, the loop may execute zero or more iterations.

while condition is true

sequence of statements

end while

In a do-while loop, the condition is evaluated at the end, which means that the loop will always execute at least one iteration.

do

sequence of statements

while condition is true

**Conditional Statements**

A conditional statement will evaluate a condition and determine the course of action accordingly. The statement can have one or more alternative paths, or options, as shown below.

**Conditional -- One option**  
The statements are executed if the condition is true only. If the condition is not true, the next statement following the conditional statement will be executed.

if condition is true

sequence of statements

end if

**Conditional -- Two options**  
The first set of statements is executed if the condition is true. If the condition is not true, the alternative set of statements will be executed. Note that either the first or the alternative set executes.

if condition is true

sequence of statements

else

alternative sequence of statements

end if

**Conditional -- Multiple options** One of the sequences executes depending on the conditions specified. The conditions are going to be checked in order. When the first true condition is found, the

if condition1 is true

1st sequence of statements

else if condition2 is true

2nd sequence of statements

else if condition3 is true

3rd sequence of statements

else

last sequence of statements

end if

**Examples**

Example 1: Given two numbers, x and y, write the pseudo-code to output the greater one.

input x and y

if x > y

output x

else

output y

end if

Example 2: Given two numbers, x and y, write the pseudo-code to output the greater one, or "same", if they are equal.

input x and y

if x > y

output x

else if y > x

output y

else

output "same"

end if

Example 3: Assuming a sequence of integer numbers, from x to y, where x < y, write the pseudo-code to output all the numbers in the sequence.

input x and y

for i = x to y

output i

end for

Example 4: Assuming a sequence of integer numbers, from x to y, where x < y, write the pseudo-code to ouput all the even numbers in the sequence.

input x and y

for i = x to y

if i is even

output i

end if

end for

Example 5: Assuming a sequence of integer numbers, from x to y, where x < y, write the pseudo-code to count how many of the numbers are multiple of 3.

input x and y

counter = 0

for i = x to y

if the remainder of i/3 is zero

increase counter

end if

end for

output counter

The loop above will assign a new value to variable i as it iterates from x to y. In each iteration, the code checks if i is a multiple of 3, by calculating the remainder of the division. If the remainder is zero, the counter is incremented. At the end of the loop the program outputs the value of variable counter.

**Important: when counting, set the value of the counter to zero before start counting.**

**Exercises:**

* Assuming a sequence of integer numbers, from x to y, where x < y,
  1. write the pseudo-code to output yes, if variable z's value is between x and y, and no otherwise.
  2. write the pseudo-code to output yes, if variable z's value is not between x and y, and no otherwise.
  3. write the pseudo-code to add all the numbers, from x to y, and output the sum.
  4. write the pseudo-code to multiply all the numbers, between x and y, and output the product.