Control Structures

Algorithmics and C Programming

Chapter 4
Karim Bouzoubaa
Chapter 4: control structures

- Objectives
  - Be able to write conditional statements
  - Be able to write repetitive statements

- outline
  - Introduction
  - The if selection
  - The if ... else selection
  - The while repetition
  - The for repetition
  - The do ... While repetition
  - Comprehensive program
Introduction

- algo so far
  - Assignements \((a = 5)\)
  - Input/Output \((\text{display, read})\)
- Computer programs are more complex et contain more elaborated statements
- Example (remind/chapter 2):

**Planting multiple trees:**

1. Dig a hole
2. Place a tree
3. Reseal the hole
4. **If** still other trees
   - Execute **actions 1, 2, 3 and 4**
   - **else** Execute next actions
5. Water the trees

- **action 4 checks if a condition is true or not**
  - \(\iff\) **Selection**
- **actions 1, 2, 3 and 4 are repeated n times**
  - \(\iff\) **Repetition**
Introduction

- Computer science provides other types of statement called control structures
- These structures allow to control the flow of execution of the whole statements

- Conditional structures (selections):
  - if
  - if ... else

- Repetitive structures (Iterations):
  - while
  - do ... while
  - for
The if structure

The if selection consists in executing a sequence of statements only when a condition is satisfied.

```
instruction_0;
if (condition) {
    instruction_1;
    instruction_2;
    ...
    instruction_n;
}
instruction_{n+1};
```
The if structure

- Consider a and b as two variables. An algo that determines and displays the largest of these two variables

Algorithm "maximum of 2 variables"
- Input : x, y type integer
- Output : max type integer
- Process:
  ```
  Begin
  read x, y
  if (x > y) max = x
  if (y ≥ x) max = y
  Display(" The maximum is: ", max)
  End
  ```
The if ... else selection consists in selecting between two sequences depending on the truth value of some condition.

```
instruction ;
if (condition) {
    instruction_1 ;
    instruction_2 ;
    ...
    instruction_n ;
}
else {
    instruction_1 ;
    instruction_2 ;
    ...
    instruction_m ;
}
instruction ;
```
Consider $a$ and $b$ as two variables. An algo that determines and displays the largest of these two variables.

Algorithm "maximum de 2 variables"

Input : $x$, $y$ type integer

Output : $\text{max}$ type integer

Process:

Begin
read $x$, $y$
if $(x > y)$ $\text{max} = x$
else $\text{max} = y$
display("" The maximum is: ", $\text{max}$)
End
1. An algorithm that determines and displays the minimum of three variables (many possible algo)

2. An algorithm that solves the equation $ax + b = 0$

3. An algorithm that solves the equation $ax^2 + bx + c = 0$
The while repetition structure

- The sequence is executed as many times as necessary while the condition is satisfied

```c
while (condition) {
    instruction_1;
    instruction_2;
    ...
    instruction_n;
}
```
The while repetition structure

- Algo that displays the sequence 0 2 4 6 8 ... 50

- Algorithm “display sequence 0 2 ... 50“
- Input :
- Output :
- temporary: val

- Process:
  Begin
  val = 0
  while(val < 52) {
    display (val)
    val = val + 2
  }
  Display(“End“)
  End
The while repetition structure

- Algo that computes the average of n values (n > 0)

Algorithm “avg of n values”

Input : n, val type integer

Output : avg type real

temporary: count, sum

Process:

Begin
read n
count= 0
sum = 0
while (count < n) {
    lire val
    sum = sum + val
    count= count+ 1
}
Avg = sum / n
Display("the avg is: ", avg)
End
Exercises

1. Algo that displays the sequence 0  5  10  …  100
2. Algo that displays the sequence 100  90  80  …  10
3. Algo that computes the factorial of n
4. Algo that reads a series of values, stops at -1 and displays the sum, product and avg of these values
5. Algo that computes the product of two numbers n and m using only the addition operator
The do ... while repetition structure

- The do ... While structure is mainly useful when we need execute statements of a loop one time at least.
- The condition is checked at the end of the loop instead of at the end
  - Even if the condition is false from the begining, the sequence is executed one time at least

```plaintext
do{
    instruction_1 ;
    instruction_2 ;
    ...
    instruction_n ;
} while (condition) ;
```
Algo that computes the product $P = 10 \times 9 \times 8 \times \ldots \times 1$

**Algorithm “product from 10 to 1”**

**Output**: $p$ type integer

**temporay**: count

**Process**:

```
Begin
count = 10
p = 1
Faire { 
  p = p \times count 
  count = count - 1
} while (count > 0)
Display(“the product is:“, p)
End
```
The do ... while repetition structure

- The most common use: reading a variable with condition

**Val in $[0, +\infty]$**

- ... 
- **input:** val integer
- ...  
- **Process:**
  
  ...  
  do{
    read val
  } while (val < 0)  
  ... 

**Val in $[5, 10]$**

- ... 
- **input:** val integer
- ...  
- **Process:**
  
  ...  
  do{
    read val
  } while (val<5 OR val>10)  
  ...
The for repetition structure

- For is used to execute a loop a fixed number of times

```c
for var = initial to final GAP gap {
    instruction_1;
    instruction_2;
    ...
    instruction_n;
}
```

- When gap = 1, it could be ignored
The for repetition structure

- Algo that displays the sequence 0 2 4 6 8 … 50

```
• Algorithm “sequence evens < 50”
• temporary: i integer
• Process:
  Begin
  for i=0 to 50 gap 2 {
    display (i)
  }
  End
```
The for repetition structure

- Algo that displays the n first power of 2

```
• Algorithm “n power of 2”
• Input : n integer
• temporary: i, p integer
• Process:
  Begin
  read n
  p = 1
  for i=1 to n {
    p = p * 2
    display (p)
  }
  End
```
The for repetition structure

- Algo that computes
  - \( f(x,y) = \frac{x}{x^2 + y^2} \)
  - \( x = 0, 2, \ldots 10 \)
  - \( y = 1, 3, \ldots 21 \)

- We have nested loops, i.e. the second loop is part of the first one. The second loop is entirely executed for every iteration of the first one.

Algorithm “\( f(x,y) \)”

`temporay:`
- \( i, j \) integer
- \( f \) real

`Process:`

```plaintext
Begin
  for \( x = 0 \) to 10 gap 2 {
    for \( y = 1 \) to 21 gap 2 {
      f = x / (x^2 + y^2)
      display (f)
    }
  }
End
```
1. While exercises

2. Algo that computes the first n prime numbers that are greater than 100

3. Algo that computes the sum $S$ of the 100 first members of the mathematical sequence

$$S = 1 - \frac{1}{2} + \frac{1}{4} - \ldots + \frac{(-1)^p}{2^p}$$

4. Algo that computes the sum

- Adding elements from left to right
- Adding elements from right to left
- Adding separately positive and negative members

$$S = 1 - \frac{1}{2} + \frac{1}{3} \ldots + \frac{1}{999} - \frac{1}{1000}$$