Practical worksheet N°4. Loops

1) Objectives

The objective of this lab sheet is to explain, through various examples and practical exercises, the different iterative structures that can be used in the C programming language and the cases for using each of these structures.

Upon completion of this lab, you should be able to solve any problem involving repetitions and use all the studied loops correctly.

2) The « `while` » loop

Consider the following program:

```
#include <stdio.h>
int main() {
  float a,b,c;
  printf("Please enter two real numbers : ");
  scanf("%f%f",&a,&b);
  while (b == 0) {
     printf("Provide a non-zero value for the divisor: ");
     scanf("%f", &b);
  }
  c = a / b;
  printf("%f / %f = %f", a, b, c);
}
```

1. Create a new project and type the code above.
2. Compile and run the program. What does this program do?
3. What will happen if we give a zero value to the variable **b**?
4. What will happen if we give a non-zero value to the variable **b** from the beginning?
5. Remove the `scanf("%f",&b)` statement, then compile and run the program. What happened?
6. What is the difference between using `while` and using `if` in this example?
7. Rewrite the code using a `do…while` loop and explain the difference with `while`.

3) The « `for` » loop

Consider the following program:

```
#include <stdio.h>
```

```
int main() {
    int i,n;
    printf("Enter a positive integer : ");
    scanf("%d", &n);
    for (i = 1; i <= n; i++) {
        if(i%2==0)printf("%d is an even number\n",i);
        else printf("%d is an odd number\n",i);
    }
}
```

1. Create a new project and type the code above.
2. Compile and run the program. What does this program do?
3. Add the necessary instructions for it to also calculate and display the number of values in each class.
4. Add the instruction **`i=i+1`** at the end of the loop. What do you notice?
5. Change the last instruction to **`i=i-1`**. What happened?
6. Rewrite the program using a **`while`** loop.

4) Application exercises

1. Write a program that asks the user to enter a positive integer **n**, then calculates and displays its factorial (**n!**) with the formula: n! = 1 * 2 * 3 * ... * n. **(The algorithm was covered in TW)**.
2. Write a program that asks the user to enter an integer between 1 and 9, then displays its multiplication table. If the user enters a number less than 1 or greater than 9, the program should display an error message. **(The algorithm was covered in lecture)**.
3. Write a program that asks the user to enter a positive integer *n*, then calculates and displays the sum of its digits.

   Execution example:

   `Enter a positive integer: 148`

   `The sum of the digits in 148 is 13.`

4. Write a program to help students calculate the weighted average of first-semester courses. For each course, the student must provide the coefficient and the mark. When he finishes entering all the courses, the student should enter 0 in place of the coefficient. The weighted average (using the coefficients) is calculated using the following formula:

$$weighted\ average = \frac{\sum mark \times coefficient}{\sum coefficient}$$

5. Write a program that reads a sequence of real numbers ending with 0, and calculates and displays the minimum and maximum of the entered numbers.

6. The Greatest Common Divisor (GCD) of two integers is their largest common divisor. We want to use the Euclidean algorithm to calculate the GCD of two positive integers (the GCD of two integers is equal to the GCD of their absolute values, so we restrict ourselves to positive integers). The algorithm is based on the following observations:
   - The GCD of two numbers is not changed if we replace the larger of the two by their difference.
     In other words, for a > b, GCD(a, b) = GCD(a - b, b).
   - The GCD of any number and 0 is always the number itself.
     In other words, GCD(a, 0) = a.

   Exemple : Calculate the GCD of 35 and 20.

   35 – 20 = 15 ⇒ GCD(20, 15).
   20 – 15 = 5 ⇒ GCD(15, 5).
   15 – 5 = 10 ⇒ GCD(10, 5).
   10 – 5 = 5 ⇒ GCD(5, 5).
   5 – 5 = 0 ⇒ GCD(5, 0) = 5.
   So GCD(35, 20) = 5.

   Write a program that asks the user to provide two positive integers, then uses the Euclidean algorithm to calculate their GCD.

7. Write a program that converts a number from decimal format to binary format (base 2), and then to octal format (base 8).

   Modify the program to convert the input to any base provided by the user.

8. The Fibonacci sequence is defined as follows :
   - $F_1 = 1$
   - $F_2 = 1$
   - $F_i = F_{i-1} + F_{i-2}$ for **i > 2**.

   Write a program to calculate the n^{th} term of the Fibonacci sequence for a positive integer **n** given by the user.

5) Additional exercises

The exercises in this section are additional self-study exercises.

9. Write an algorithm that allows to read a positive non-zero integer **x** and calculate the following sum up to the n^{th} term. The number ***n*** is entered by the user. **(The algorithm was covered in TW)**.

$$S = \sum_{i=1}^{n} \left(x + \sum_{j=1}^{i-1} j \right)$$

   For example, if ***x*** = 3 and ***n*** = 4, the sum ***S*** is:

   S = 3 + (3+1) + (3 +1+2) + (3+1+2+3) = 3+4+6+9 =22

10. Write a program that displays a right-angled triangle made of asterisks, with the height entered by the user.

    Example of a triangle with a height of 5:

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

11. Write a program that tests whether an integer provided by the user is prime or not, knowing that a prime number is an integer greater than 1 that is divisible only by 1 and itself. **(The algorithm was covered in TW)**.

12. Write an algorithm that reads two positive non-zero integers, A and B, and finds their Least Common Multiple (LCM).

13. We can approximate the value of π using the following expression:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - ....$$

    Write an algorithm that calculates and displays an approximate value of π using the above expression. The calculation stops when the difference between two consecutive values of this expression becomes strictly less than $\mathbf{10^{-3}}$.

14. Write a program that asks for a real number ***M*** and returns the value of the first integer ***n*** for which the following sum is strictly greater than ***M***:

$$u_n = \sum_{k=1}^{n} \frac{1}{k}$$

15. Write a program that allows to enter *N* integer values, then calculates and displays the largest multiple of 3.

    Example 1: *N* = 5, the numbers entered are: 7, 9, 15, -30, 11. The program should display 15.

    Example 2: N = 3, the entered numbers are: 4, -8, 23, 2. The program should display “There is no multiple of 3”.