COMPUTER SCIENCES – 2011-2012

Lab 4

Exercise Goals

- Solve problems involving many consecutive logical choices
- Experiment with the concept of loop

Technical Contents

- Introduction to the *switch* construct
- Use of looping constructs: *while, do-while* and *for*
- Inserting of decision constructs *if-then-else* within loops

---

To solve in lab preferably

Exercise 1. Write a C program that reads in input integer numbers from the keyboard until the user inserts the value *0*.
Hint: use the loop construct *while* or *do-while*.
In depth: modify the program by accumulating during the acquisition process into the variable *sum* the values inserted before placing the number *0*; at the end of the acquisition, the program will print on the screen the calculated value *sum*.

Exercise 2. Write a C program that reads in input integer numbers from the keyboard until the user inserts the value *0* and calculates the average of the inserted values. The program must:

a. Sum all acquired values in a variable called *sum*, opportunely defined and initialized
b. Count the number *i* of acquired values
c. Divide *sum* by *i* and show the results

In depth: modify the program so that during the acquisition the values that are not within an interval bounded by two constants *MIN* and *MAX* (defined by the programmer) are discarded.

Exercise 3. Write a C program to display on the screen the first 20 values of the Fibonacci series.
Hint: the first values of the series are: 0 1 1 2 3 5 8…
Formally, the series is implemented by using the following relation:
$X_i = X_{i-1} + X_{i-2}$, with $X_0 = 0$ and $X_1 = 1$;
In depth: modify the series as follows:
$X_i = X_{i-1} * X_{i-2}$, with $X_0 = 1$ and $X_1 = 2$; how many elements of this series can be represented with integer variables?

Exercise 4. Write a C program that reads in input from the keyboard a positive integer value $N <= 40$ corresponding to the base of a right and isosceles triangle, and represents the triangle on the screen by using of '*' characters.
Example: if the value inserted by the user is 3, the following sequence of characters has to be displayed:
*
**
***
In depth: consider different geometrical figures, such as isosceles triangle, square, etc…

Exercise 5. Write a C program that given an integer number between 1 and 12 representing the current month, is able to display the extended name of the month using a *switch* construct (1→"January", 2→"February", 3→"March", …, 12→"December").
The program must also handle wrong inputs from the user.

Exercise 6. Write a C program that reads in input from the keyboard a decimal number *N* and then acquires from the keyboard a sequence of integer numbers until the following conditions are fulfilled:
a. The average of the inserted values is greater than *N*
b. Less than 10 numbers have been acquired.

Exercise 7. Write a C program to evaluate the maximum value that can be stored in variables of types *int*, *long* and *unsigned int*.
Hint: following the path shown below, use the step-by-step debugging mode and analyze the results of the various assignments.
a) Verify that there is not a practicable way to try to assign values progressively larger; if, for example, you write the instructions *value = 3000000000*, the compiler does not report error, but (maybe) only a warning. What do you see if you observe *value* with the watch after the execution of instruction?
b) Try to get these values in an "empirical" way, i.e. by acquiring and printing them through *scanf* and *printf* functions. Verify that also this is not a correct procedure: the behavior of *scanf* in case of error is not controllable by the programmer.
c) At this point implement an algorithm that, taking into account the binary representation of unsigned and two's complement numbers, allows to detect the maximum value: for signed numbers, you can initialize *value* to 0, then increases it repeatedly. It is known that if you increase by 1 the

maximum positive value you get an overflow, and the value becomes negative; so the searched value is the value that precedes the first negative value found. Translate this procedure into a program and test it. How can you modify the algorithm (and the program) to work with unsigned numbers?