Part 1: Sub-programs in C language

Calculate the triple of an integer

The following program calculates the triple of a number and then displays it, using the concept of a sub-program.

```
#include <stdio.h>
#include <stdlib.h>
int triple (int n) /*définition de la fonction triple*/
{
int r; /*r integer variable local to the triple function*/
r=n*3; /* r is assigned the triple of n */
return r; /*r is the value returned as the result of the function*/
}
main( )
{
int i,j,k; /*i and j integer variables local to main (belong only to main )*/
i=2; /*i is assigned 2*/
j=triple(i); /* call of the function triple: j is assigned 6, the triple of 2*/
printf ("the triple of %d is %d  \n", i,  j); /* it displays 6*/
j=triple(4); /* call of the function triple: j is assigned 12, the triple of 4*/
printf ("the triple of %d is %d  \n",4, j); /*affiche 12*/
return 0;
}
```

a. Create a new project, type in this code, then compile and run.
b. Modify this program to enter an integer and display its triple.
c. Type the following program, then compile and run it :

```
#include <stdio.h>
#include <stdlib.h>
void triple (int , int *); /* prototype of the triple function which takes two integer parameters i and j
( where j is the triple of i). i is passed by value and j is passed by address. */
main( )
{
int i,j=0; /* i and j are integer variables local to main and j initialised at 0*/
i=2; /*i is assigned 2*/
triple(i,&j); /*Call of the function that displays 6, triple result of 2*/
printf ("triple of %d is %d  \n", i, j); /*Displays 6 (new value of j after the function is called))*/
}
void triple (int i, int *j) /*Definition of the function triple*/
{
*j=3*i; /*j is assigned 6 the triple of 2*/
printf ("THE TRIPLE OF %d IS %d  \n", i, j); /*displays 6*/
}
```

a. What do you notice ?
b. Correct errors.

Concluding remarks

1. The only form of sub-program in C is the functions.
2. A function is defined as:

   Result_Type Function_Name (Type Param1,…,Type Paramn);
   {
   <Declarations of Local Objects >
   <instructions >
   }

   With:

   Result_Type: corresponds to the type of the result of the function ((int, double, void...).

   Function_Name : is the name of the function (an identifier).
3. The ***return*** statement is used to specify the result that the function should return. You can use any expression after a ***return*** statement.
4. The ***return*** statement can appear several times in a function.
5. In C programming language, a procedure is a function that does not return any result. In such cases, the ***Result_Type*** is replaced by ***'void'***, indicating an empty return.
6. A parameter passed by reference must be declared as a pointer using ***'*'***.
7. Arrays are always (by default) passed by address.
8. The function is called by its name, possibly followed by the parameters enclosed in brackets:
   - If the function returns a result and admits parameters
     Variable = Function_Name (Actual parameters);
   - If the function returns a result and does not admit parameters
     Variable = Function_Name ( );
   - If the function does not return a result and accepts parameters
     Function_Name (Actual parameters);
   - If the function returns no result and accepts no parameters
     Function_Name ( );

Remarks

- The C language does not allow nesting of functions.
- If the return type of a function is not explicitly declared, it is automatically assumed to be of type ***int***.
- In the C language, a function must be known, but not necessarily fully defined, before its first usage in the program. This means its declaration must precede its call, but its definition can follow. This is known as a function prototype, and it is declared almost like a function:

  Result_Type Function_Name (Type Param1,…,Type Paramn);

  - In the parameters of the prototype, only the types are truly necessary; the identifiers are optional.
  - Place the prototype at the beginning of the program, and the function becomes usable anywhere in the code.

Part 3: Application Exercises

Exercise 1

Write a program that allows entering a positive non-zero integer number ***n*** and displays the twin primes between ***1*** and ***n***. Knowing that two numbers ***a*** and ***b*** are *twin primes* if:

- ***a*** and ***b*** are prime numbers
- ***a*** = ***b*** + 2 The program should consist of:

- A function ***is_prime(x)*** that returns true if ***x*** is a prime number, and false otherwise.
- A function ***are_twin_primes(a, b)*** that returns true if ***a*** and ***b*** are twin primes, and false otherwise.
- A procedure ***displayTwinPrimes(n)*** that displays the twin primes between ***1*** and ***n***.

Exercise 2

We want to write a Multiplication Game program that asks the user to recite his multiplication table.

1. The user begins by entering a number between 2 and 9 (if the number is incorrect, the program asks again).
2. The program then displays the lines of the multiplication table for this number one by one, leaving the result blank and waiting for the user to enter the result.
    - If the result is correct, it moves on to the next line,
    - If not, an error message is displayed, giving the correct value, and the program ends. If all the answers are correct, a congratulatory message is displayed.
3. The program must include at least:
    - a function to correctly input the number.
    - a procedure to display a line of the multiplication table.

Exercise 3

Consider the sequence defined by: $U_0 = 1$, $U_1 = 1$, $U_n = 2U_{n-1}+U_{n-2}$.
Write a program in C language that enters a positive non-zero integer ***n*** and calculates and displays the term of rank ***n*** in the sequence $\boldsymbol{U_n}$.

Exercise 4

Given that: $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots$ for x very close to zero.

Write a program that displays ***cos(x)*** using the formula above. The calculation stops when the difference between two consecutive terms becomes less than or equal to 10^{-4}. The program must contain procedures and functions.