TP 1 : subprograms (functions and procedures)

1. Introduction :

- Subprograms allow programs to be divided into modules to facilitate understanding and maintenance and alsofor avoid the repetitions in the code.
- As we has seen in course, he exist two types of subprograms: procedures and functions.
- Contrary to in algorithmic, the C++ language generalizes the structure of the function on the two types of subprograms. In other words, in C++ we have only functions, procedure are specialisation of function
- For that, we go to start by the study of functions in C++, then next we will see the structure of the procedures.

2. Functions:

The general structure of a function in language C++ is as follows:

function_return_type ***function_name*** ***(*** *type_1* **_variable1** , *type_2* **variable2,** .... *typeN* **variableN** ***)***

{

< **Declaration of the function variables** >

<instructions1>;

<instructions2>;

...

return (result); /* For return the result */

}

Example :

An algorithm who read three numbers positive not null (A, B and C), calculate and display the following sum: **A! + B! + C!**

Language Algorithmic	Language C++
Algorithm Example A ,B,C , Sum: whole --- */* Definition of there function factorial */* **Function** Factorial (N: integer): **integer** fact , i : integer **Begin** fact ← 1; **For** i ←N to 1 **DO** fact ← facts * i ; **END For** ; **Return** (fact) ; **END** --- */* Program main */* **Begin** Read (A ,B,C) ; Sum ← Factorial(A) + Factorial(B) + Factorial(C) ; Write(Sum) ; **END.**	**#include** _ **<iostream>** _ _ using namespace **std ;** --- */* Definition of there function factorial */* **int** Factorial (**int** N) **{** **int** fact , i ; fact =1; **for** (i=N; i>=1; i= i-1) **{** fact = fact * i; **}** **return** (fact); **}** --- */* Main programmen ( ) */* **int** main () **{ int** A, B, C, Sum; cin >>A>>B>>C; Sum = Factorial(A) + Factorial(B) + Factorial(C); cout << Sum ; **getchar ( );** return 0; **}**

Exercise 1:

Write a C++ program that will read three numbers positive not null (A, B and C) and then calculate and display the following sum: $A! + B^C$. You must define a function to calculate B^C.

3. Procedures:

A procedure in C++ is represented by a function with a type **void** (Nothing). The exit parameters of procedures are passed by ***reference*** in C++ (pass by variable) using the operator **' & ' (The & operator is equivalent to Var in algorithmics).**

Example 1 : A program that display the sum and the product of two number whole HAS And b.

Language Algorithmic	Language C++
Algorithm Example1 A ,B, Sum, Product: whole --- */* Definition of there procedure Calculation */* **procedure** Calculate **(** X1, X2:integers; **var** S,P: integer**)** **Beginning** S ← X1 + X2; P ← X1 * X2; **END ;** --- */* Main Program */* **Begin** Read (A ,B); **calculate** (A,B,Sum,Product); Write(Sum, Product) ; **END.**	**#include <iostream>** _ _ using namespace **std ;** --- */* Definition of the procedure Calculate */* **void** calculation (**int** X1, **int** X2, **int &** S, **int &** P) **{** S = X1 +X2; P = X1 *X2; **}** --- */* M a i n p r o g r a m m e n ( ) * / _ _* **int** main () **{ int** A,B, Sum, Product; cin >>A>>B>>C; **calculate** (A,B, Sum, Product); cost << Sum << Product; **getchar ();** return 0; **}**

Pass by reference

Exercise 2:

Write a C++ program that performs the permutation of two variables A and b using a procedure.

4. Reuse of subprogram

We can put the main program, functions and procedures in separated files, which allow several programs to reuse a same subprogram.

For example we can put the main program in the file " **TP.cpp** " and the factorial subprogram in the file " **functions_tp.cpp** " as follows:

File "TP.cpp"

```
#include <iostream> _
#include "functions_tp.cpp"
using namespace std ;

/* Main programmen ( ) */___ __ __
int hand ()
{ int ABC, Sum;
cin >>A>>B>>C;
Sum = Factorial(A) + Factorial(B) + Factorial(C);
cost << Sum ;
getchar ( );
return 0;
}
```

For indicate or find this there function Factorial

File "functions_tp.cpp"

```
/* Definition of there function factorial */
int Factorial ( int NOT)
{
int Result , fact , i ;
fact =1;
for (i=N; i>=1; i= i-1)
{
fact = fact * i;
}
Result = fact ;
return ( Result );
}
```

Noticed :

You must put the files " **TP.cpp** " and " **functions_tp.cpp** " in the same directory before the compilation of the "TP.cpp" program.

Exercise 3 (Home Work): consider the exercise number 4 of the first series of Directed Work (TD). Implement in C++ the algorithmic solution of this exercise.

> We define a ***bi-prime*** number as being a prime number whose inverse (or mirror) is a prime number. For example the number 17 is bi-prime because it is a prime number and its inverse 71 is also a prime number. We want to display all bi-prime numbers less than an integer A.