

Practical Work series N°1: Subroutines

1. Introduction:

- Subroutines allow programs to be divided into modules to facilitate understanding and maintenance of programs and also to avoid repetition in the code.
- As we saw in the course, there are two types of subroutines: procedures and functions.
- Unlike the algorithmic language, the C++ language **generalizes the structure of the function** across both types of subroutines.
- To do this, we will start by studying the functions in C++ language, then look at the structure of the procedures.

2. Functions:

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

```
function_return_type function_name ( type_1 variable1 , type_2 variable2 .... typeN variableN )  
{  
< Declaration of function variables >  
<instructions1>;  
<instructions2>;  
...  
return (result); /* To return the result */  
}
```

Example :

An algorithm that reads three non-zero positive numbers A, B, and C and calculates and displays the following sum: **A! +B! +C!**

Algorithmic Language	C++ language
<pre> Algorithm Example A ,B,C , Sum: integers //----- /* Definition of factorial function */ Function Factorial (N: integer): integer ; Result , fact, i : integer; Begin fact ← 1; For i = N to 1 step = -1 do fact ← fact * i ; End for ; Result ← fact ; Return (Result); END ; //----- /* Main program */ Begin Read (A ,B,C); Sum ← Factorial(A) + Factorial(B) + Factorial(C); Write(Sum); END. </pre>	<pre> #include <iostream> using namespace std; //----- /* Definition of factorial function */ int Factorial (int N) { int Result , fact, i ; fact =1; for (i =N; i >=1; i = i-1) { fact = fact * i ; } Result = fact; return (Result); } //----- /* Main program <i>main</i> () */ int main () { int A,B,C, Sum; cin >>A>>B>>C; Sum = Factorial(A) + Factorial(B) + Factorial(C); cout << Sum; getchar (); return 0; } </pre>

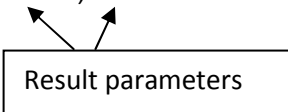
Practical exercise 1:

Write a C++ program that reads three non-zero positive numbers A, B, and C and calculates and displays the following sum: $A! + B^c$

3. Procedures:

A procedure in C++ is represented by a function with **void** return type (**Nothing**). The output parameters of procedures are passed by **variable** (address) using the **'&'** **operator** and are defined using the **'*'** **operator**.

Example 1: A program that displays the sum and product of two integers A and B.

Algorithmic Language	C++ language
<pre> Algorithm Example1 A ,B,C ,Sum,Product: integer; //----- /* Definition of the Calculation procedure */ procedure calculation (X1 , X2: integer; var S, P: integer); Begin S ← X1 + X2; P ← X1 * X2; END ; //----- /* Main program */ Begin Read (A ,B); calculation (A, B, Sum ,Product); // procedure call Write(Sum, Product); END. </pre>	<pre> #include <iostream> _ using namespace std; //----- /* Definition of the Calculation procedure */ void calculation (int X1, int X2, int * S, int * P) { * S = X1 + X2; * P=X1*X2; } //----- /* Main program <i>main</i> () */ int main () { int A,B, Sum, Product; cin >>A>>B>>C; calculation (A,B, & Sum, & Product); // procedure call cout << Sum << Product ; getchar (); return 0; } </pre> <div style="text-align: right; margin-top: 20px;">  </div>

Practical exercise 2:

Write a C++ program that permutes two integer variables A and B.

4. Reusing subroutines

The main program and functions and procedures can be put into separate files, allowing multiple programs to use the same subroutine.

For example, we can put the main program in the “ **PW.cpp** ” file and the factorial program in the “ **PW_functions.cpp** ” file as follows:

The "PW.cpp" file	The “PW_functions.cpp” file
<pre> #include <iostream> _ #include "PW_functions.cpp" using namespace std; /* Program main main() */ int main () { int A,B,C, Sum; cin >>A>>B>>C; Sum = Factorial (A) + Factorial (B) + Factorial (C); cout << Sum; getchar (); return 0; } </pre> <div data-bbox="461 480 1144 541" style="border: 1px solid black; padding: 2px; margin-left: 100px;"> <p>To indicate where is the Factorial function</p> </div>	<pre> /* Definition of factorial function */ int Factorial (int N) { int Result , fact, i ; fact =1; for (i =N; i >=1; i = i-1) { fact = fact * i ; } Result = fact; return (Result); } </pre>

Noticed :

You must put the “ **PW.cpp** ” and “ **PW_functions.cpp** ” files in the same directory before compiling the “PW.cpp” program.

Practical exercise 3:

Repeat the last two practical exercises by putting the functions **Factorial** , **power**, and **permute**, in three deferent files and the main program in other file.