

Practical work series N°2: Recursion

1. Definition :

In programming, the *recursion* is a method that allows a module (procedure or function) to call herself.

It is in there body part (instructions) we find a call to the procedure (function) herself.

Example : a function **fact** allowing of calculate the factorial of an integer **n**

Iterative solution	Recursive solution
<pre>int fact(int n) { int i, p=1 ; for (i=1 ;i< n;i++) p = p * i ; return p ; }</pre>	<pre>int fact(int n) { int p ; if (n == 0) p = 1; else p = n * fact (n-1); return p ; }</pre> <div style="position: absolute; top: 100px; left: 300px; border: 1px solid black; border-radius: 50%; padding: 5px; width: fit-content;">Recursive call</div>

2. Types of recursion :

We distinguished in general two types of recursion :

- Simple recursion
- crossed recursion

2.1. Simple recursion:

It is when a sub-program (function or procedure) is called herself.
It is in the general case of recursion as we there already seen In the previous with the factorial function.

Example : calculation of the sum of **n** first positive integer numbers.

$$\text{Sum}(n) = 1 + 2 + 3 + \dots + (n-1) + n$$

```
#include <iostream>
using namespace std;
int nbr;
int Sum (int n)
{
  int S;
  if (n== 1)
    S = 1 ;
  else
    S = Sum (n-1) + n ;

  return (S);
}
int main ()
{
  cout <<"please enter an integer number:"<< endl;
  cin>> nbr;
  cout << "the sum is: " << Sum(nbr) << endl;
  getchar();
}
```



2.2. Crossed recursion:

We calls crossed recursion when two procedures P1 And P2 are called mutually, i.e. : when **P1** executes, She calls to **P2** , and when **P2** executes, She calls to **P1** .

Example :

A positive integer number n can be either :

- Even → n = 2*k
- Odd → n = 2*k+1

If we considered two functions Even (n) And Odd (n) witch have logical values (boolean), so we will have :

- If Even (n) = true then Odd (n) = false
- If Odd (not) = true then Even (not) = false

```

#include <iostream>
using namespace std;
int nbr;
// Statement of the headers of the functions
int even (int);
int odd (int);
// The implementation of the functions
int even (int n)
{
    int R;
    cout<<" call of function even (n=" << n << ")"<<endl;
    if (n==0)
        R= 1;
    else
        R= odd (n-1);

    return (R);
}
int odd (int n)
{
    int R;
    cout<<" call of function odd (n=" << n << ")"<<endl;
    if (n==0)
        R= 0;
    else
        R= even (n-1);

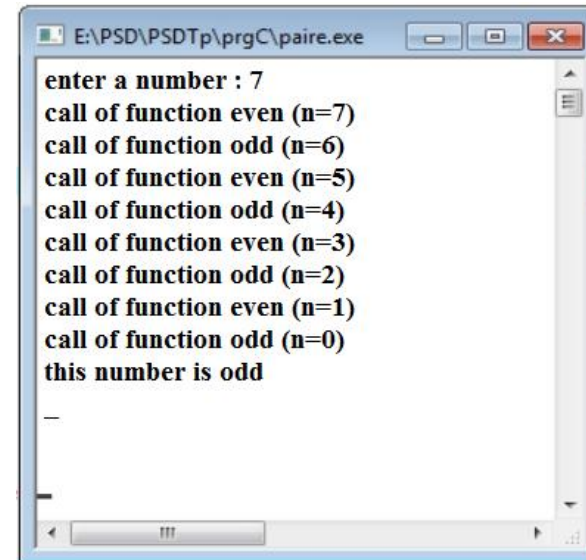
    return (R);
}
// main program
int main ()
{
    cout <<"enter a number: "<< endl;
    cin >>nbr;
    // even(nbr);
    if (even(nbr) ==0)

```

```

    cout <<" This number is even "<<endl;
    else
        cout <<" This number is odd "<<endl;
    getchar();
    return 0;
}

```



TP :

- I. Write a C++ **recursive** procedure that reads elements of a vector V of N elements, and a recursive function that search the Max of these elements
- II. the dichotomy research of an element in an ordered vector is carried out as follows :
 - 1) We split the vector into two parts noticeably equal,
 - 2) We compared the value has seek with the element of medium,
 - 3) If they are not equal, we are interested uniquely with the part containing the elements wanted And we abandons the other part.
 - 4) We start again these 3 steps until we have one element to compare.

Write a **recursive** function to research by dichotomy an element **X** ?