The 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
int fact(int **n**)	int fact(int **n**)
{	{
int i, p=1 ;	int p ;
	if (n = = 0)
for (i=1 ;i< n;i++)	p = 1;
p = p * i ;	else
	p = n * fact (n-1); — Recursive call
return p ;	
}	return p ;
	}

2. Types of recursion :

We distinguished in general two types of recursion :

- Simple recursion
- crossed recursion

2.1. Simple recursion:

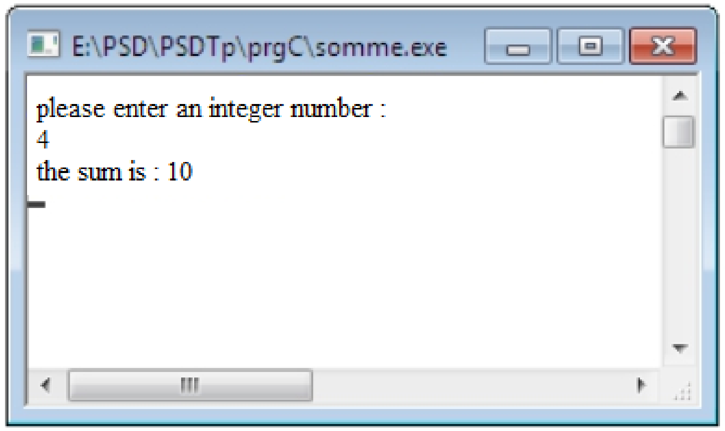
It is when a sub-program (function or procedure) is called herself. It is in the general case of recursion as we there already seen In the previous with the factorial function.

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

Sum(n) = 1 + 2 + 3 + … + (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();
  }
```



```
please enter an integer number :
4
the sum is : 10
```

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 :

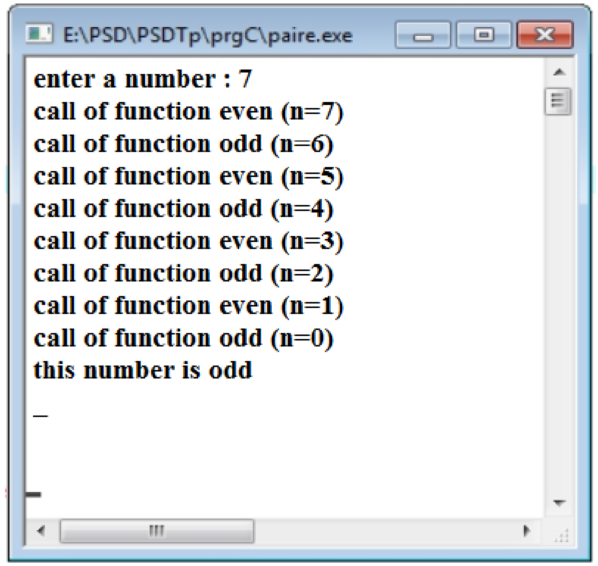
If Even (n) = true then Odd (n) = false

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;
 }
```



```
enter a number : 7
call of function even (n=7)
call of function odd (n=6)
call of function even (n=5)
call of function odd (n=4)
call of function even (n=3)
call of function odd (n=2)
call of function even (n=1)
call of function odd (n=0)
this number is odd

_
```

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** ?

Algorithmic solution :

I. reading vector:

```
Procedure ReadVE (n: integer; Var V []: array of
integers);
i: integer;
Begin
If (n=1) then
Read (V[1]);
else
readVE(n-1, V);
Read (V[n]);
end if
END ;
```

searching for the max element

```
function MaxVE (n: integer; V []: array of
integer;): integer;
max,i: integer;
Begin
If (n=1) then
   max ← V[1];
else
max ←MaxVE(n-1, V);//recursive call
 If (max<V[n]) then
   max ← V[n];
 end if
End if
Return( max );
END ;
```

II. The function returns the rank of the element if he exist Otherwise she returns -1.

```
Function Research (Val:integer; V[]:array of integer;
Iinf,Isup:integer): integer ; // Iinf: index inferior of V and Isup: index superior
Imil : integer; // index of medium
Begin
Imil ⟵ (Iinf+Isup) div 2 ;
if (Iinf>Isup Or V [Imil]= Val) then // condition to stop
If (Iinf>Isup) then
Return(-1); // Val does not exist  In V
else
Return (Imil);
end if
else
If ( Val< V [Imil] ) then
 Return (Research (Val, V, Iinf, Imil-1)); // recursive call
else
 Return (Research (Val, V, Imil+1, Isup)); //recursive call
end if
end if
END ;
```