Cours 3 : l'algorithmique et programmation avec sous-programme

Transmission des paramètres et *segment de pile*.

Exemple : (que se passe-t-il à l'exécution ? *segment de pile?*)

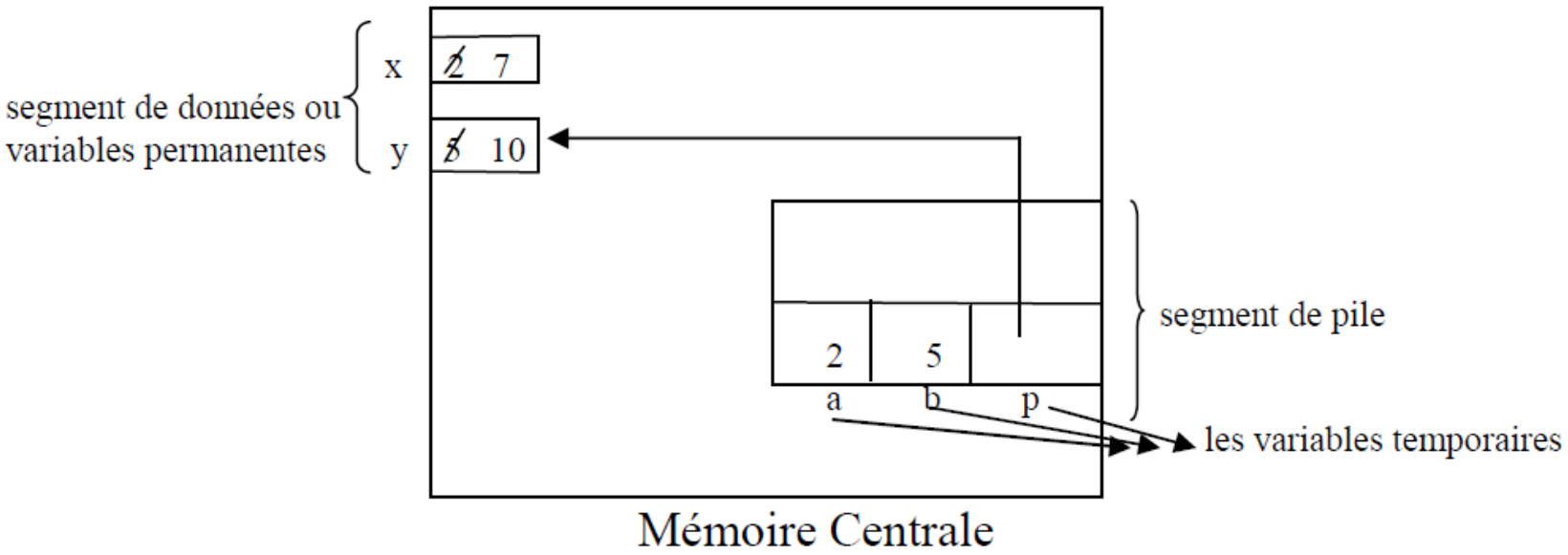
```
 int addmul (int a, int b, int *p)
{    *p = a*b;
     Return   a+b;
}
main()
{  int x = 2, y = 5;
   printf("debut programme principal :\n x = %d \t y = %d\n",x,y);
   x= addmul(x,y,&y) ;
   printf("fin programme principal le produit de x+y = %d  et  x*y =
%d",x,y);
}
```

Cours 3 : La programmation avec sous-programme

Les variables permanentes, temporaires , le contexte d'exécution et *segment de pile.*

Que se passe-t-il à l'appel ? contexte d'exécution et *segment de pile?*

```
 int addmul (int a, int b, int *p)
{    *p = a*b;  Return   a+b;
}
main()
{  int x = 2, y = 5;  /*variables permanentes */
   printf("debut programme principal :\n x = %d \t y = %d\n",x,y);
   x= addmul(x,y,&y) ; /*les variables temporaires seront réservées  à cet appel*/
   printf("fin programme principal le produit de x+y = %d  et  x*y = %d",x,y);
}
```



Cours 3 : La programmation avec sous-programme

Les objets manipulés par les sous programmes variables globales et permanentes

Les Variables globales en C sont celles déclarées dans un programme C à la suite des directives de compilation ces variables sont accessibles par toutes les fonctions définies dans ce programme.

```
#include <stdio.h>
int p=0;   /* variable globale*/
void prod(int a, int n);
 main()
 {int x=3;
  printf("p= %d\n",p);
  prod(x,2);
  printf("p= %d\n" , p);
  prod(x,3);
  printf("p= %d" , p);
 }
 void prod( int a, int n )
    {  static int k;  /*variable locale mais statique */
            printf("k = %d ", k); k++;
       p=a*n;
    }
```

Cours 3 : La programmation avec sous-programme

Les objets manipulés par les sous programmes variables globales et permanentes

Note : le langage C permet de définir des **variables statiques aux variables locales**, avec un mot clé **static** une variable locale jouera un rôle de variable permanente accessible par la fonction où elle est définis, elle est initialisée à zéro avant le premier appel et gardera sa valeur du précédent appel à chaque nouvel appel de la fonction qui éventuellement la midifiera.

```
void prod( int a, int n )
   { static int k; /*variable locale mais statique */
         printf("k = %d ", k); k++;
     p=a*n;
   }
```

Cours 3 : La programmation avec sous-programme

Les sous programmes assurent le concept récursif.

La récursivité est un concept qui consiste à définir un mode de calcul d'une manière déclarative (par définition) et qui masquera la boucle explicite.

Exemples :

1)

$0!=1$

$n!=n*(n-1)!$

algorithme récursif de calcul de la factorielle

2)

$x^0=1$

$x^n= x*x^{(n-1)}$

algorithme récursif de calcul de la puissance

3)Pgcd , fibbonacci , palindrome et ....

Cours 3 : La programmation avec sous-programme

Les sous programmes assurent le concept récursif.

La récursivité est un concept qui consiste à définir un mode de calcul d'une manière déclarative (par définition) et qui masquera la boucle explicite.

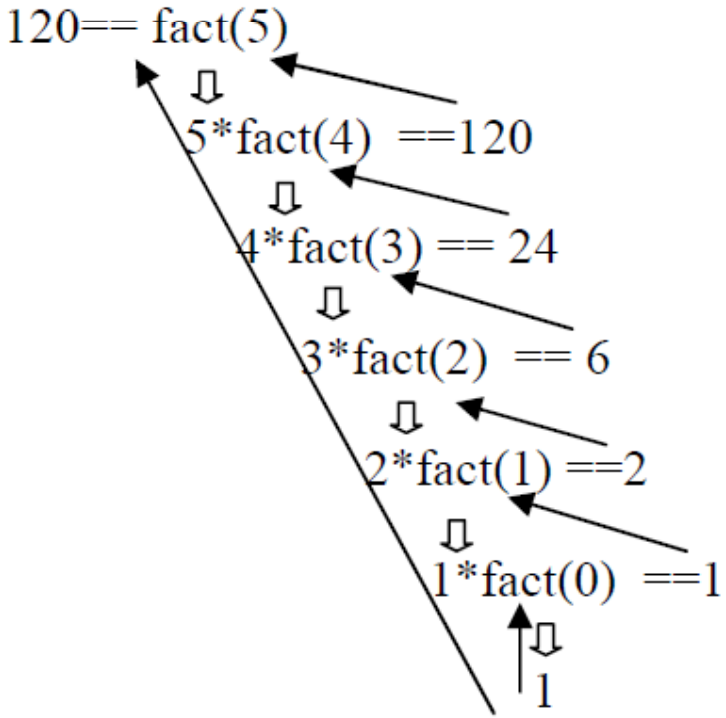
Exemples :

```
#include <stdio.h>

int fact( int n )  /* on suppose   n une valeur  positive ou nulle*/
{
   if (n==0) return 1;
   else return  n*fact(n-1);
}

main()
{
  printf(″%d ″,fact(5) );
}
```

Cours 3 : La programmation avec sous-programme

Les sous programmes assurent le concept récursif.

```
#include <stdio.h>
int fact( int n )  /* on suppose   n une valeur  positive ou nulle*/
{   if (n==0) return 1;
    else return  n*fact(n-1);
}
main()
{ printf("%d ",fact(5) );  /*      affichera  120 obtenu comme schématisé */
}
```



Cours 3 : La programmation avec sous-programme

Les sous programmes assurent le concept récursif.

Le cas particulier est impératif pour l'arrêt de la boucle implicite induite par les appels récursifs.

```
#include <stdio.h>

int fact( int n )
{
   if (n==0) return 1;

   else  return  n*fact(n-1);
}

main()
{
  printf("%d",fact(5) );
}
```

le cas particulier sans appel récursif

l'appel récursif avec valeur qui mènera au cas particulier

Cours 3 : La programmation avec sous-programme

Les sous programmes assurent le concept récursif.

Définition : La résolution par récurrence consiste à définir la forme du concept d'un problème par la même forme mais en en plus petit.

Exemple 1 : écrire une fonction en C qui permet de vérifier si un mot est palindrome, sachant qu'un mot est palindrome :
A- s'il est vide ou
B- si le premier caractère est égale au dernier caractère et le mot sans ces deux caractères est palindrome.

Exemple 2 : écrire une fonction en C qui permet de calculer la somme des éléments d'un tableau de taille n :
A-Somme(v,n) = 0 si n=0
B-Somme(v,n) = v[n]+Somme(v,n-1)

Cours 3 : La programmation avec sous-programme
Les sous programmes assurent le concept récursif.

Exercice 1 : La vérification si un tableau est ordonné. Donner une définition récursive. Ecrire une fonction récursive qui vérifie si un tableau d'entiers est trié.

Exercice 2 : Recherche dichotomique d'une valeur val si elle se trouve dans un tableau d'entiers ordonné.