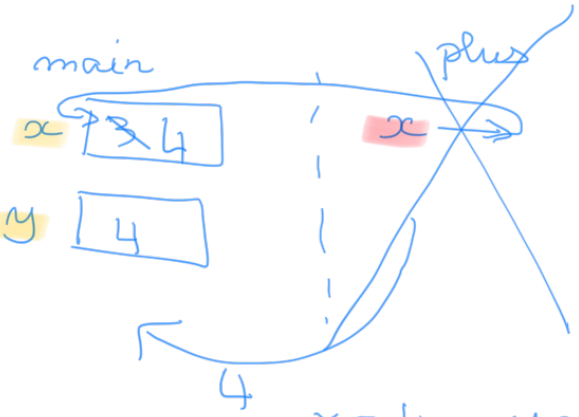
Exemple (à comparer avec l'exemple donné pour la passage par valeur:

```
int plus(int *x)
{
    *x = *x + 1;
    return (*x);
}

int main()
{
    int x, y;
    y = plus(&x) ;
    printf(" x= %d  y=%d ",x, y);
 }
```

x est une adresse




x = 4 y = 4

Equivalence : résultat retourné par la fonction ou résultat passé en paramètre

résultat retourné par la fonction

```
typef f(…)
{
   typef resultat ;
   ……
   resultat = …….. ;
   return resultat ;
}
```

typef : le type de la fonction
int, float ……

equivalent

résultat passé en paramètre

```
void calculef(……., typef *f)
{
   typef resultat ;
   …
   resultat = …;
   *f = resultat ;
   return ;

}
```

Appel :

```
typef valeur ;
```

cas 1 `valeur =f(…) ;`

cas 2 `calculef(…, &valeur) ;`

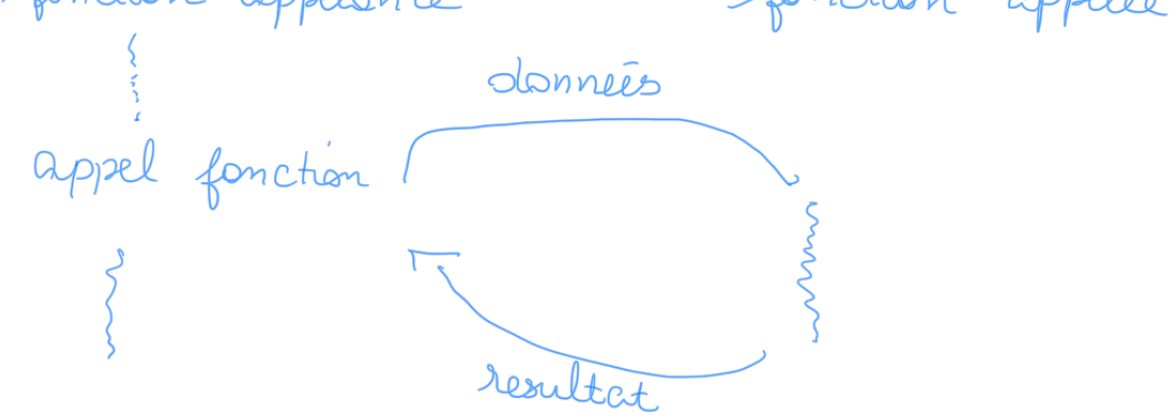
exemple calcul de la factorielle d'un entier positif ou nul

```
long fact (int x)
{ long F; /* resultat */
  int k; /* boucle */
  F = 1;
  for (k=1; k<= x; k++)
      F *= k;
  return F;
}
int main ()
{ long y;
  int x;
  x = 4;
  y = fact(x);
  printf (" %d ! = %ld ", x, y);
  return 0;
}
```

```
void fact (int x, long *y)
{ long F; /*resultat */
  int k ; /* boucle */
  F = 1;
  for (k = 1 ; k <= x ; k++)
      F *= k;
  *y = F;
}
int main()
{ long y;
  int x;
  x = 4;
  fact (x, &y) ;
  printf(" %d ! = %ld ", x, y);
  return 0;
}
```

```
void fact (int x, long *y)
{ int k;
  *y = 1;
  for (k=1; k<=x; k++)
      *y = *y * k ; /* ou  *y *= k ; */
  return;
}
int main()
{ int x;
  long *y;        long 3;
  x = 4;          y=&....
  fact (x, y);    <- non car y n'est pas initialisé
                  &3
```

bilan sur les paramètres




paramètres
- données qui n'ont pas vocation à être modifiés
- données qui doivent être modifiées
- resultats : paramètre qui contient un resultat

retour
- resultat retourné par return (1 seul possible)

si souligné en rouge : passage par adresse.

ex : calcule des coefficients d'une droite

$y = ax + b$.

à partir des coordonnées de 2 points (2D)

$P_1 \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$ $P_2 \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$.

$a = \frac{y_2 - y_1}{x_2 - x_1}$ $b = y - ax$

$b = y_1 - a x_1$

hyp : $x_2 \neq x_1$

paramètres

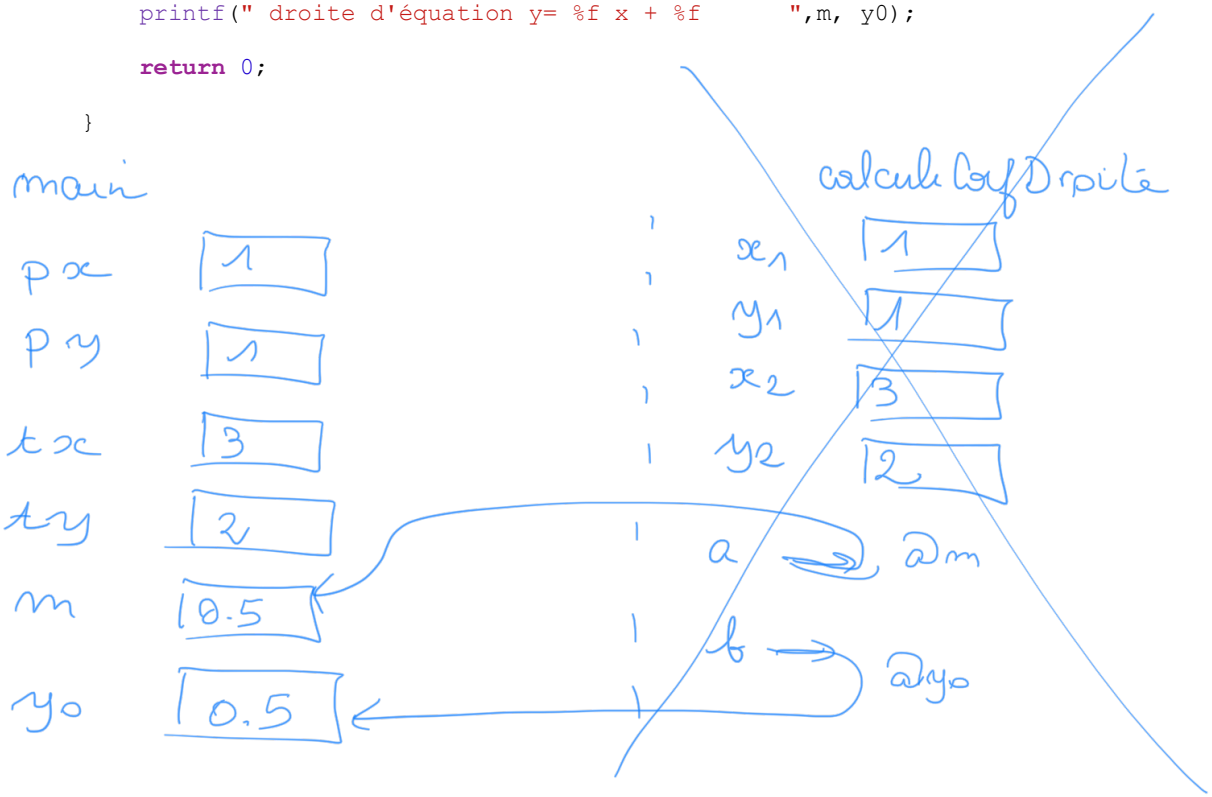
données $x_1\ y_1\ x_2\ y_2$ ) passage par valeur

résultat $a$ $b$ ) passage par adresse.

---

```
long fact (int);  /* prototype entete simplifiee */
int main()
{
   : ecriture du main
}
long fact (int x)
{
   : ecriture de la fonction
}
```

```c
#include <stdio.h>


void  calculeCoefDroite(float x1,float y1, float x2, float y2, float *a, float *b)
// calcul des coefficients d'une droite a et b a partir des coordonnées 2D de 2
//points (x1,y1) (x2,y2). y = ax+b
//
//Les coordonnées des points sont des données de la fonction.
//Ils n'ont pas vocation à être modifiés par la fonction donc passage par valeur

// a et b sont des adresses car ces paramètres vont permettre de transmettre des
//résultats de la fonction appelée vers la fonction appelante.

{ // hypothèse sur les données :  x2 est différent de x1

    *a = (y2-y1)/(x2-x1);
    *b = y1- (*a * x1);
    return ;
}



int main()
{
    float px,py;  //premier point
    float tx,ty; // deuxieme point

    float m,y0;  // resultats coefficients de la droite y= mx+y0

    px= 1;
    py=1;
    tx = 3;
    ty = 2;

    calculeCoefDroite(px,py,tx,ty,&m,&y0);
    printf(" droite d'équation y= %f x + %f      ",m, y0);

    return 0;

}
```



```
double puissance (double , int);
:
int main()
{ double y;

  y = puissance (3.54, 2);
  :
  }
```

```
double puissance (double x, int n)
{ :
  : x, n,
  :
} : return.
```

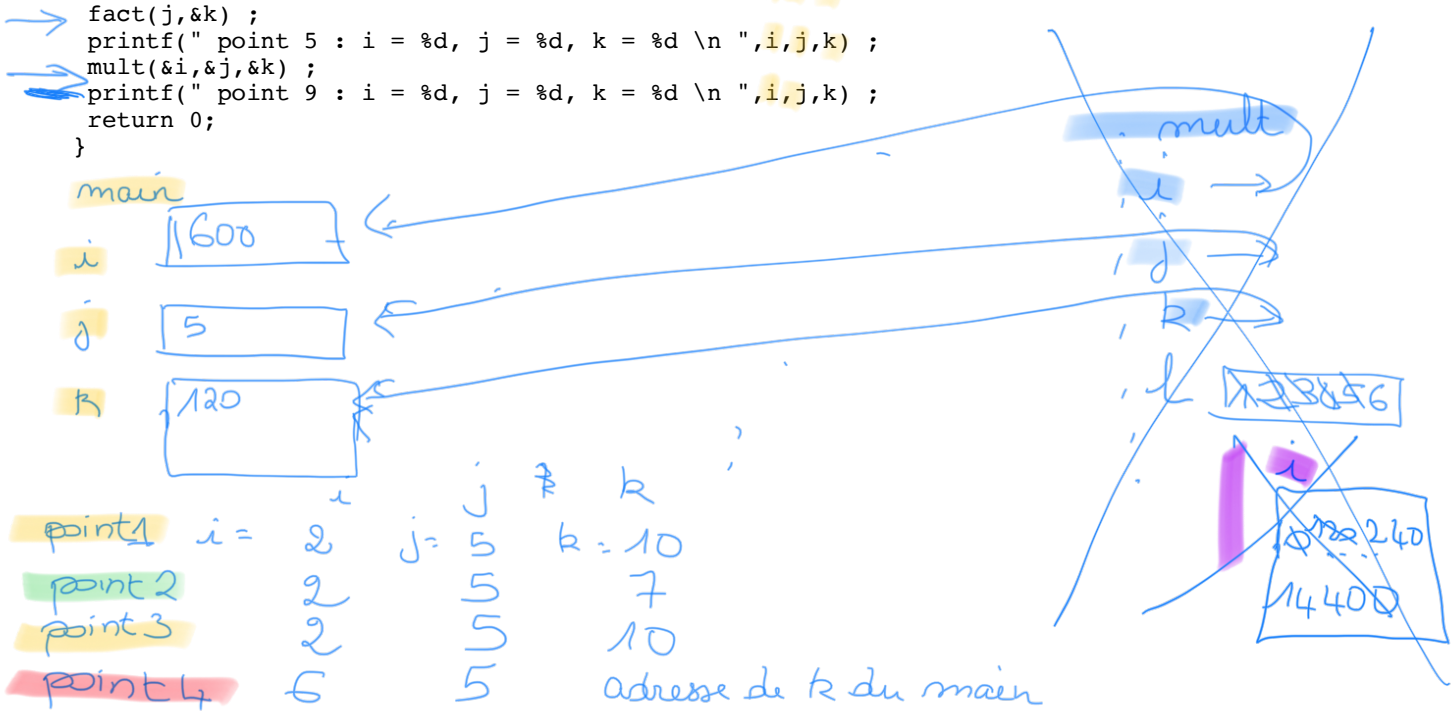
réel

| | |
|---|---|
| float | $10^{-6}$ |
| double | $10^{-15}$ |

Indiquer la valeur des variables i, j, k aux différents *printf* du programme.

```
#include <stdio.h>
void plus(int i,int j,int k)
{
        k = i + j ;
        printf(" point 2 : i = %d, j = %d, k = %d \n ",i,j,k) ;
}
void fact(int j, int *k)
{
        int i ;
        for (i=1, *k = 1  ; i<=j  ; i++)
                *k  = *k * i ;
        printf(" point 4 : i = %d, j = %d, k = %d \n ",i,j,k) ;
}
void mult( int *i,int *j,int *k)
{
        int l ;
        for ( l=1 , *i = 0 ; l<=*j ; l++)
                *i = *i + *k ;
        printf(" point 6 : i = %d, j = %d, k = %d \n ",*i,*j,*k) ;
        { int i = 0 ;
                for( l = 1 ; l <= *k ; l++)
                        i = i + *k ;
                printf(" point 7 : i = %d, j = %d, k = %d \n ",i,*j,*k);
        }
        printf(" point 8 : i = %d, j = %d, k = %d \n ",*i,*j,*k) ;
}

int main()
{
 int i,j,k;
 i = 2;
 j = 5;
 k = 10;

 printf(" point 1 : i = %d, j = %d, k = %d \n ",i,j,k) ;
 plus(i,j,k) ;
 printf(" point 3 : i = %d, j = %d, k = %d \n ",i,j,k) ;
 fact(j,&k) ;
 printf(" point 5 : i = %d, j = %d, k = %d \n ",i,j,k) ;
 mult(&i,&j,&k) ;
 printf(" point 9 : i = %d, j = %d, k = %d \n ",i,j,k) ;
 return 0;
}
```

k et non *k




| | i | j | k |
|---|---|---|---|
| point 1 | 2 | 5 | 10 |
| point 2 | 2 | 5 | 7 |
| point 3 | 2 | 5 | 10 |
| point 4 | 6 | 5 | adresse de k du main |
| point 5 | 2 | 5 | 120 |
| point 6 | 600 | 5 | 120 |
| point 7 | 14400 | 5 | 120 |
| point 8 | 600 | 5 | 120 |
| point 9 | 600 | 5 | 120 |