Bibliothèque d’entier PASCAL

Arbaoui Houssam Eddine

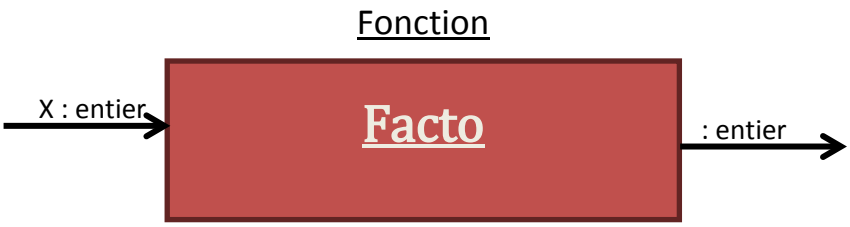
Rôle : Verifie Si X est premier

Analyse : on considère X comme premier, si X<=1 alors X n'est pas premier, si X>1, On vérifie pour chaque i compris entre 2 et X div 2 si il divise X, si on trouve un seul diviseur alors X n'est pas premier sinon il est premier

Algorithme :

```
Fonction Premier(X: entier) : booléen
Var i : entier ; inter : booléen
Debut
Si X<=1 alors inter← faux sinon inter← Vrai
I ← 2
Tant que (i <= X div 2) et inter alors
  Debut
  Si X mod i = 0 alors inter ← faux
  I ← i+1
  Fin
Premier ← inter
Fin
```

☺



Rôle : Calcule $X!$

Analyse : On initialise Facto à 1, on fait varier i de 2 à X et dans chaque itération on multiplie Facto par i

Algorithme :

```
Fonction Facto (X: entier) : entier
Var inter, i : entier
Debut
Inter ← 1
Pour i allant de 2 à X faire inter ← inter * i
Facto ← inter
fin
```

☺

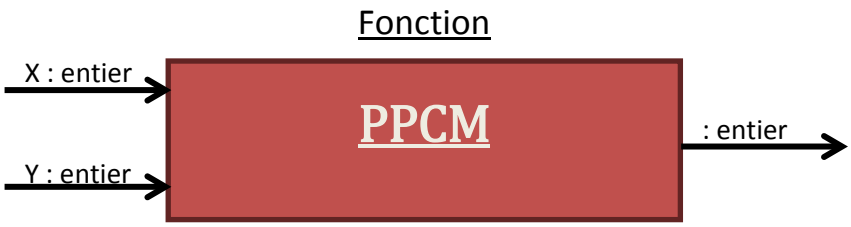
Rôle : Renvoi le plus grand commun diviseur de X, Y

Analyse : On utilise l'algorithme d'Euclid. Soit r = X mod Y, tant que Y ne divise pas X (r≠0), X prend la valeur de Y et Y prend la valeur du reste, à la fin PGCD = Y ; (on travaille avec les valeurs absolues de X et Y)

Algorithme :

```
Fonction PGCD (X, Y: entier) : entier
Var R : entier
Debut
X← abs(X) ; Y ← abs(Y)
Si (X * y = 0) alors Si X>=Y PGCD ← X sinon PGCD ← Y
Sinon
  Debut
  R  ← X mod Y
  Tant que R>0 faire
    Debut
    X ← Y
    Y ← R
    R ← X mod Y
    Fin
  PGCD ← Y
  fin
fi
```

☺

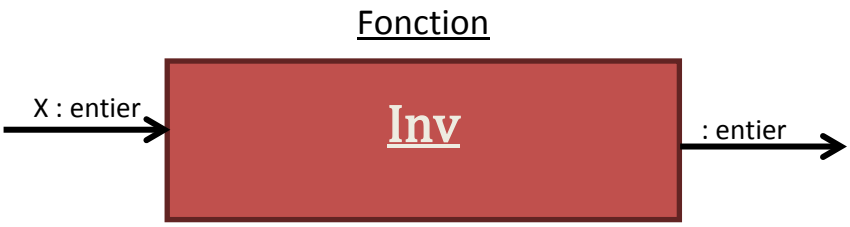


Rôle : Renvoi le plus petit commun multiple X, Y

Analyse : On utilise la loi : $PPCM = \frac{X*Y}{PGCD\,(X,Y)}$ (tout en travaillant avec les valeurs absolues de X et Y)

Algorithme :

```
Fonction PPCM (X, Y: entier) : entier
  Debut
  X← abs(X) ; Y ← abs(Y)
  PPCM ← (X * Y) div PGCD(X,Y)
  fin
☺
```



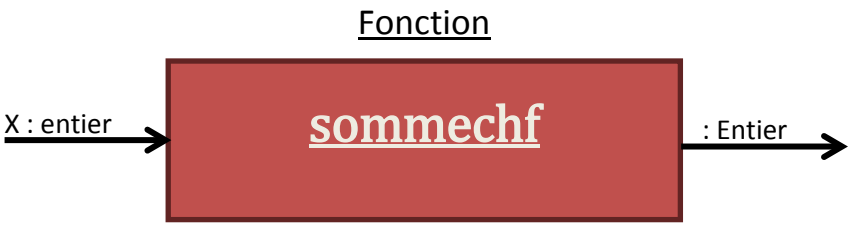
Rôle : Calcule l'inverse (le miroir) de X

Analyse : On fait des divisions successives de X sur 10 et dans chaque itération on Multiplie Inv par 10 et on ajoute a Inv X mod 10 (Inv étant initialisé à 0)

Algorithme :

```
Fonction Inv (X: entier) : entier
Var inter, signe : entier
  Debut
  Si X<>0 alors signe ← abs(X) div X sinon signe ← 1
  X ← abs(X)
  Inter ← 0
  Tant que X>0 faire
    Debut
    Inter ← 10*inter + X mod 10
    X ← X div 10
    Fin
  Inv ←signe* inter
  Fin
```

☺



Rôle : Calcule la somme des chiffres composants X

Analyse : On fait des divisions successives sur X sur 10 jusqu'à que X atteint 0 ; et à chaque itération on ajoute à sommechf X mod 10, (sommechf étant initialisé à 0)

Algorithme :

```
Fonction sommechf(X : entier) : entier
Var inter: entier
  Debut
  Inter ←0
  X ← abs(X)
  Tant que X>0 faire
    Debut
    inter ← inter + X mod 10
    X ← X div 10
    fin
  Sommechf ← inter
  fin
☺
```

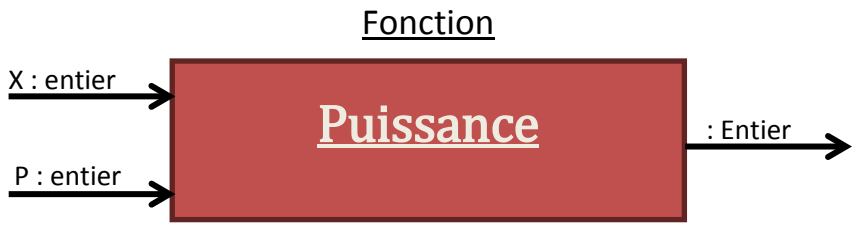
Rôle : Calcule le produit des chiffres composants X

Analyse : On fait des divisions successives sur X sur 10 jusqu'à que X atteint 0 ; et à chaque itération on multiplie produitchf par X mod 10, (produitchf étant initialisé à 1)

Algorithme :

```
Fonction produitchf(X : entier) : entier
Var inter: entier
  Debut
  X ← abs(X)
  Si X = 0 alors Inter ← 0 sinon Inter ← 1
  Tant que X>0 faire
    Debut
    inter ← inter *( X mod 10)
    X ← X div 10
    fin
  produitchf ← inter
  fin
```

☺



Rôle : calcule x^p

Analyse : On fait varier un entier i de 1 à P et à chaque itération on multiplie puissance par X (puissance étant initialisée à 1)

Algorithme :

```
Fonction puissance(X, P : entier) : entier
Var inter, i : entier
Debut
Inter ←1
Pour i allant de 1 à P faire inter ← inter * X
Puissance ← inter
fin
```

☺

Rôle : Renvoi le nombre de chiffre composant X

Analyse : On fait des divisions successives sur X sur 10 jusqu'à que X soit Strictement inférieur à 10 ; et à chaque itération on incrémente Nbchf par 1, Nbchf étant initialisé à 1

Algorithme :

```
Fonction Nbchf(X : entier) : entier
Var inter: entier
  Debut
  Inter ←1
  X ← abs(X)
  Tant que X>9 faire
    Debut
    inter ← inter +1
    X ← X div 10
    fin
  Nbchf ← inter
  fin
```

☺

Rôle : Extraire un nombre de K chiffres de X à partir de la position pos

Analyse : On fait une division entière de X sur 10^{pos-1} et on prend le reste de la division entière de X sur 10^k ; $extpos = (X\ div\ 10^{pos-1})\ mod\ 10^k$

Algorithme :

```
Fonction extpos(X, pos, k : entier) : entier
Debut
Extpos ← (X div puissance(10, pos − 1)) mod puissance(10, k)
fin
```

☺

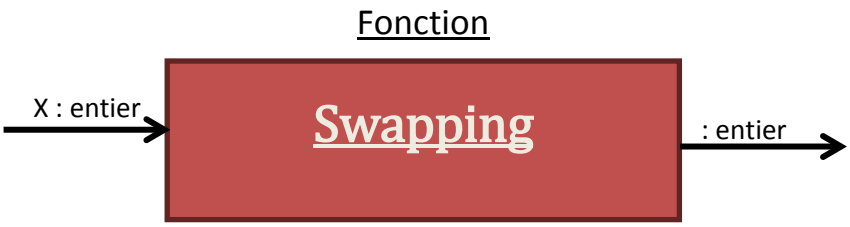
Fréquence d'un chiffre dans un nombre

Rôle : Renvoi le nombre d'apparition d'un chiffre ch dans X

Analyse : On fait des divisions successives de X sur 10 jusqu'à que X atteint 0, et à chaque itérations on vérifie si $X \bmod 10 = ch$, si c'est le cas on incrémente Freq par 1 (Freq étant initialisé à 0)

Algorithme :

```
Fonction Freq (Ch, X : entier) : entier
Var inter : entier
  Debut
  X ← abs (X)
  Inter←0
  Tant que X>0 faire
    Debut
    Si X mod 10 = ch alors inter← inter + 1
    X ← X div 10
    Fin
  Freq ← inter
  Fin
☺
```



Rôle : Calcule l'inverse (le miroir) de X

Analyse : On extrait le poids Fort et le Poids Faible De X, on Calcule l'expression :

Swapping = $(X + \left((Pfaible - Pfort) * (10^{nbc\,hf(X)-1} - 1)\right)$

Algorithme :

```
Fonction Swapping (X: entier) : entier
Var Pfaible, Pfort, Signe : entier
Debut
Si X<>0 alors signe ← abs(X) div X sinon signe ← 1
X ←abs(X)
Pfort ← extpos(X, nbchf(X), 1) ; Pfaible ← X mod 10
Swapping ← Signe*(X + ((Pfaible – Pfort)*(puissance (10,
nbchf(X)-1)-1)))
Fin
```

☺

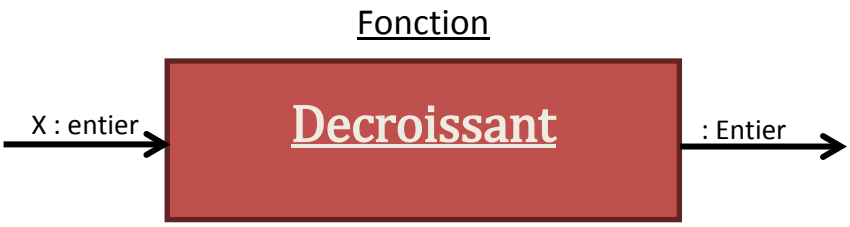
Rôle : Ordonne les chiffres de X du plus petit au plus grand

Analyse : On fait varier C de 1 à 9 et à chaque itération on fait diviser successivement X sur 10 jusqu'à qu'il atteint 0 et à chaque fois on compare X mod 10 avec C si il y a égalité alors on multiplie Croissant par 10 et on ajoute à croissant C, (Croissant étant initialisé à 0)

Algorithme :

```
Fonction Croissant (X: entier) : entier
Var inter, usex, C : entier
  Debut
  Inter ← 0
  Pour C allant de 1 à 9 faire
    Debut
    Usex ← X
    Tant que usex > 0 faire
      Debut
      If usex mod 10 = C alors inter ← 10*inter + C
      Usex ← Usex div 10
      Fin
    Fin
  Croissant ← inter
  Fin
```

☺



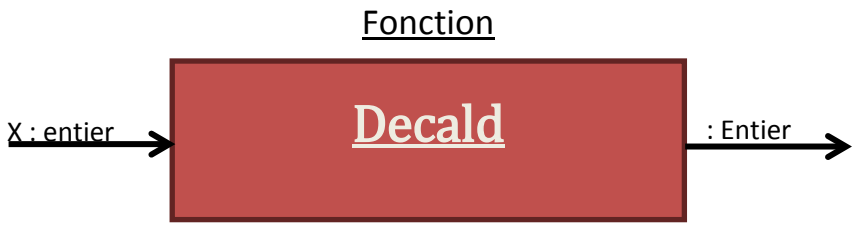
Rôle : Ordonne les chiffres de X du plus Grand au plus petit

Analyse : On fait varier C de 9 à 0 et à chaque itération on fait diviser successivement X sur 10 jusqu'à qu'il atteint 0 et à chaque fois on compare X mod 10 avec C si il y a égalité alors on multiplie decroissant par 10 et on ajoute à decroissant C, (Decroissant étant initialisé à 0)

Algorithme :

```
Fonction Decroissant (X: entier) : entier
Var inter, usex, C : entier
  Debut
  Inter ← 0
  Pour C allant de 9 à 0 faire
    Debut
    Usex ← X
    Tant que usex > 0 faire
      Debut
      If usex mod 10 = C alors inter ← 10*inter + C
      Usex ← Usex div 10
      Fin
    Fin
  Decroissant ← inter
  Fin
```

☺



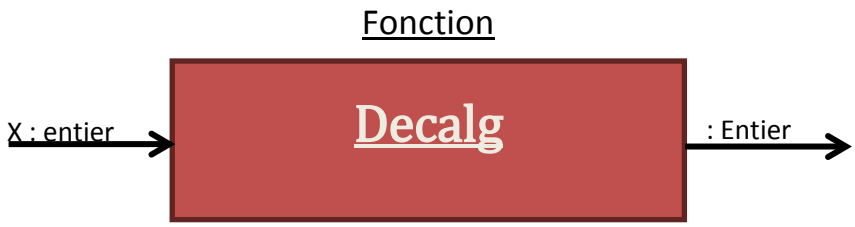
Rôle : Décale un nombre 1 seul fois de gauche à droite Circulairement (→)

Analyse : On fait la division entière de X sur 10 en ajoutant à Decald le poids faible de X multiplié par $10^{nbc\,hf\,(X)-1}$

Algorithme :

```
Fonction Decald (X: entier) : entier
  Debut
  Decald ← X div 10 + (X mod 10) * puissance (10, nbchf(X)-1)
  Fin
☺
```

Décalage circulaire à gauche



Rôle : Décale un nombre 1 seul fois de Droite à gauche Circulairement (←)

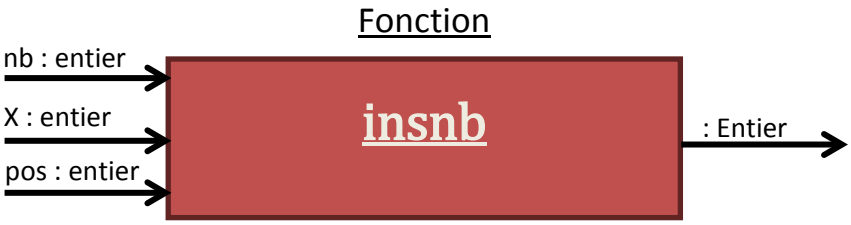
Analyse : On fait la division entière de X sur 10 en ajoutant à Decald le poids faible de X multiplié par $10^{nbc\ hf\ (X)-1}$

Algorithme :

```
Fonction Decalg (X: entier) : entier
Debut
Decalg ← 10*(X mod puissance (10, nbchf(X)-1)) + X div
puissance (10, nbchf(X)-1)
Fin
```

☺

Insertion d'un nombre dans un nombre



Rôle : Insère un nombre nb dans X à partir de la position pos

Analyse : On prend tous les chiffres composant X avant la position pos ($X\ mod\ 10^{pos-1}$), puis on ajoute nb (le nombre à insérer à la position pos($nb\ *10^{pos-1}$), à la fin il nous reste une seul partie à ajouter c'est le $X\ div\ 10^{pos-1}$, pour l'ajouter au nombre insnb on le multiplie par $10^{nbc\,hf(nb)+pos-1}$, On obtient cette expression :

$$insnb = X\ mod\ 10^{pos-1} + nb * 10^{pos-1} + (X\ div\ 10^{pos-1}) * 10^{pos+nbc\,hf(nb)-1}$$

Algorithme :

```
Fonction insnb (nb, X, pos : entier) : entier
Var k : entier
Debut
K← puissance (10, pos -1)
Insnb ← (X mod k)+   (nb*k)+(X div k)*puissance(10,pos+nbchf(nb)-1)
Fin
```

☺

Retire tous les chiffre = ch dans X

Rôle : Renvoi un nombre qui contient que les chiffres différents de ch (ordonné de la même manière)

Analyse : On fait des divisions successives de X sur 10 et dans chaque itération on vérifie si X mod 10 ≠ ch si c'est le cas alors on ajoute à delchf k*(X mod 10) (k étant un entier initialisé à 1 et multiplié par 10 sous cette conditions),(delchf étant initialisé à 0)

Algorithme :

```
Fonction delchf (chf, X : entier) : entier
Var K, inter : entier
  Debut
  Inter ← 0 ; k ← 1
  Tant que X>0 faire
    Debut
    Si X mod 10 <> chf  alors
      Debut
      Inter← inter+ K*(X mod 10)
      K←K*10
      Fin
    X ← X div 10
    Fin
  Delchf ← inter
  Fin
```

☺

Retire de X à partir de la position pos k chiffres

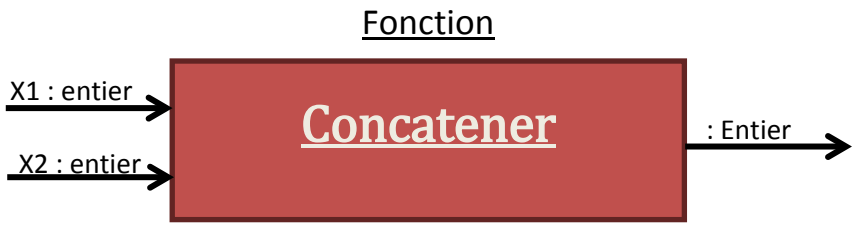
Rôle : Retire de X à partir de la position pos k chiffres

Analyse : On prend tous les chiffres composant X avant la position pos ($X\ mod\ 10^{pos-1}$) puis on ajoute à Delpos la partie qui se situe après le nombre à retirer ($X\ div\ 10^{pos+k-1}$) en la multipliant par 10^{pos-1}

Algorithme :

```
Fonction Delpos (X, pos, k: entier) : entier
Var p : entier
Debut
P← puissance (10, pos-1)
Delpos ← (X mod p) + (X div puissance (10, pos+k-1))*P
Fin
```

☺



Rôle : Concatène X1 et X2

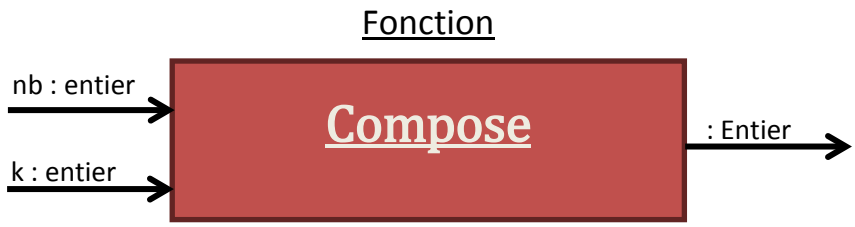
Analyse :

Pour obtenir le nombre (X1X2), il suffit de multiplier X1 par 10^p tel que P représente le nombre de chiffre composant X2 et on l'ajoute à X2 ; P= (nbchf(X2))

Algorithme :

```
Fonction concaténer (X1, X2: entier) : entier
Debut
Concatener ← X1 * puissance (10, nbchf(X2)) + X2;
Fin
```

☺



Rôle : Composer un nombre qui est la répétition de nb k fois

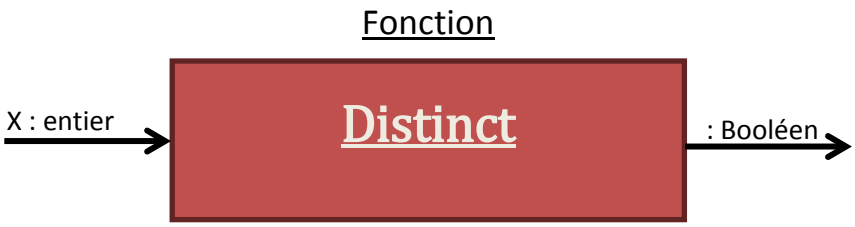
Analyse : Compose est une somme de terme d'une suite géométrique de raison P et son premier terme est nb, P = 10^m tel que m représente le nombre de chiffres composant nb, (m = nbchf(nb)), on a l'expression qui permet de calculer cette somme

$$Compose = nb * \frac{p^k - 1}{p - 1} / p = 10^{nbc\,hf\,(nb)}$$

Algorithme :

```
Fonction compose (nb, k: entier) : entier
Var p : entier
Debut
P ← puissance (10, nbchf (nb));
Compose← nb * ((puissance (p, k)-1) div (p-1));
Fin
```

☺



Rôle : Verifie si le nombre X est composé de chiffres distincts

Analyse : par définition un nombre composé de chiffres distincts ne contient pas plus d'une fois le même chiffre, alors on calcule la fréquence d'apparition de chaque chiffres dans X (en variant ch de 0 à 9 et en utilisant la fonction Freq), si la fréquence de ch dans X est strictement supérieur de 1 alors ce nombre ne contient pas des chiffres distincts,

Algorithme :

```
Fonction distinct (X: entier) : booléen
Var ch : entier ; intier : booléen
  Debut
  Inter←Vrai
  Ch ← 0
  Tant que (ch<10) et inter faire
    Debut
    Si Freq (ch, X)>1 alors inter ← faux
    Ch ← ch +1
    fin
  Distinct ← inter
  Fin
```

☺

Transformer X de la base décimal a une base dans [2,10]

Rôle : Transforme X de la base décimale à une base compris b entre 2 et 10

Analyse : On fait des divisions successives de X sur b jusqu'à que X atteint 0, et dans chaque itération on ajoute (X mod b)*p à Dectob (Dectob étant initialisé à 0 et p étant un entier initialisé à 1 et multiplié par 10 chaque itération)

Algorithme :

```
Fonction Dectob (b, X : entier) : entier
Var inter, p, signe : entier
  Debut
  Inter ← 0 ; P ← 1 ;
  Si X<>0 alors signe ← abs(X) div X sinon signe ← 1
  X ← abs(X)
  Tant que X>0 faire
    Debut
    Inter ← p*(X mod b) + inter
    P ← 10*P
    X ← X div b
    Fin
  Dectob ←signe *  inter
  fin
```

☺

Rôle : Renvoi le plus grand chiffre d'un nombre donné X

Analyse : On initialise Maxch à 0 et on fait des divisions successives de X sur 10 jusqu'à que X atteint 0 et à chaque itération on compare le reste de la division (X mod 10) avec Maxch si le reste est supérieur de Maxch alors Maxch prend la valeur de ce reste.

Algorithme :

```
Fonction Maxch (X: entier) : entier
Var inter : entier
  Debut
  Inter←0 ; X← abs(X)
  Tant que (X>0) faire
    Debut
    Si (X mod 10)> inter alors inter ← X mod 10
    X ← X div 10
    Fin
  Maxch ← inter
  Fin
☺
```

Rôle : Renvoi le plus petit chiffre d'un nombre donné X

Analyse : On initialise Minch à 9 et on fait des divisions successives de X sur 10 jusqu'à que X atteint 0 et à chaque itération on compare le reste de la division (X mod 10) avec Minch si le reste est inferieur de Minch alors Minch prend la valeur de ce reste.

Algorithme :

```
Fonction Minch (X: entier) : entier
Var inter : entier
Debut
Inter←9 ; X← abs(X)
Repeter
    Si (X mod 10) < inter alors inter ← X mod 10
    X ← X div 10
    Jusqu'à X = 0
Minch ← inter
Fin
```

☺

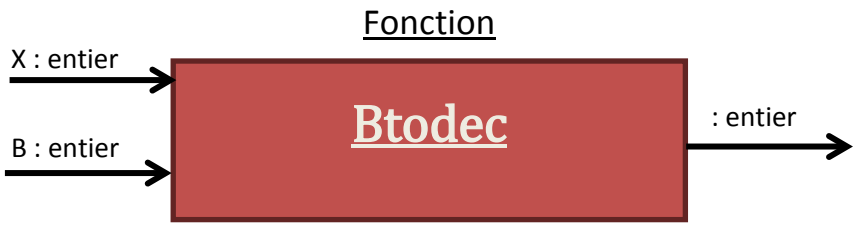
Rôle : Verifie si deux nombres Sont composé de mêmes chiffres

Analyse : On fait varier Ch de 0 à 9 et à chaque itération on compare la fréquence d'apparition de ch dans X1 et X2, s'il y a différence alors X1 et X2 ne sont pas composés de mêmes chiffres (Verif prend la valeur FAUX) , si ch atteint 10 et Verif reste Vrai alors X1 et X2 sont composés de mêmes chiffres

Algorithme :

```
Fonction Verif (X1, X2 : entier) : Booléen
Var ch : entier ; inter : booléen
  Debut
  X1 ← abs(X1) ;  X2←abs(X2)
  Inter ← Vrai ; Ch ← 0
  Tant que (Ch<10) et inter faire
    Debut
    Inter ← (Freq(Ch,X1)=Freq(Ch,X2))
    Ch ← Ch + 1
    Fin
  Verif ← inter
  Fin
```

☺



Rôle : renvoi le développement polynomiale (transforme en décimal) d'un nombre donné X écrit dans la base B

Analyse : Initialement on vérifie si le plus grand chiffre de X est strictement inférieur à b et que la base B est comprise entre 2 et 10 ; si c'est le cas, on fait des divisions successives de X sur 10 jusqu'à ce que X atteint 0 et dans chaque itération on ajoute à BtoDec (initialement nul) (X mod 10 * p) (p étant initialisé à 1 et multiplié par b à chaque itération)

Algorithme :

```
Fonction Btodec (X, b : entier) : entier
Var inter, P, Signe : entier
  Debut
  Inter ← 0 ; P ← 1 ;
  Si X<>0 alors signe ← abs(X) div X sinon signe ← 1
  X ← abs(X)
  Si (Maxch(X) < b) et (b <= 10) et (B>=2) alors
  Tant que X>0 faire
    Debut
    Inter ← p*(X mod 10) + inter
    P ← b*P
    X ← X div 10
    Fin
  Sinon inter ← 0
  BtoDec ←signe *  inter
  fin
☺
```