Algorithme : les compétences à acquérir

Notions générales :
Quand on écrit a=1 cela signifie que la variable a prend la valeur 1 : on dit que l'on affecte 1 à la variable a.
Si on écrit a = a+1 cela signifie que l'on prend ce qu'il y a dans la variable a, qu'on lui ajoute 1 et que l'on remet le résultat dans la variable a….
Il faut voir les variables comme des "tiroirs" où l'on range des données.

1) Instructions conditionnelles

Dans de nombreux algorithmes on est amené à réaliser des instructions différentes suivant les cas où on se trouve.

ex : résoudre une équation du second degré du type $ax^2+bx+c=0$

algorithme:
entrer a #ceci signifie que l'utilisateur entre un nombre qui est stocké dans la variable a
entrer b
entrer c
si a = 0
    afficher "ce n'est pas une équation du second degré"
sinon
    d=a²-4*a*c
    si d >0
        afficher $\frac{-b+\sqrt{d}}{2a}$, $\frac{-b-\sqrt{d}}{2a}$
    sinon si d =0
        afficher $\frac{-b}{2a}$
    sinon
        afficher "pas de sol"
    FinSi
FinSi

Note : quand on arrive à un "si", le programme regarde si la condition est vérifiée, et alors il exécute les instructions qui sont après le "si" et jusqu'au "sinon" ou à "FinSi".
Si la condition n'est pas vérifiée, il n'exécute pas ces instructions….et va aux suivantes.
Et le mécanisme est analogue au "Sinon si".

2) Savoir écrire un algorithme qui calcule les termes successifs d'une suite définie par récurrence

ex : $U_0 = 3$ et $U_{n+1} = 3U_n - 5$

Calcul de U_k où k est entré par l'utilisateur.

algorithme :

```
u=3  #on initialise la suite
entrer k
pour i allant de 1 à k   #c'est une boucle "pour"
        u=3 * u – 5
FinPour
afficher u
```

Note : dans cette boucle "pour", au premier passage, i vaudra 1, et toutes les instructions entre "Pour" et FinPour" sont exécutées, puis on retourne à "Pour" et i est automatiquement incrémenté de 1 (c'est-à-dire que i est augmenté de 1). On exécute alors à nouveau toutes les instructions de la boucle. On répète ceci jusqu'à ce que i= 185 (si l'utilisateur a entré k = 185). Ensuite, on exécute les instructions situées après la boucle.

3) Savoir écrire un algorithme qui détecte quand une suite dépasse un seuil

ex : on reprend la suite précédente et on veut savoir à partir de quelle valeur k le terme U_k dépasse 1000.

algorithme :

```
u=3  #on initialise la suite
k=0  #on initialise l'index (ou rang) de la suite (il s'agit de U0, k vaut 0)
tant que u < 1000    #c'est une boucle "tant que"
        u = 3 * u – 5
        k= k + 1     #on incrémente (augmente) i de 1 pour avoir l'index
FinTantQue
afficher k
```

Note : Quand on arrive dans une boucle "TantQue", le programme vérifie que la condition est vérifiée (ici que u est plus petit que 1000). Si c'est le cas, toutes les instructions sont exécutées jusqu'à "FinTantQue". Ensuite, on retourne au début de la boucle et on recommence. Si la condition n'est pas vérifiée, on passe la boucle et on exécute les instructions situées après la boucle.

A savoir : on utilise une boucle "pour" quand on sait combien de boucles on veut faire (dans l'exemple du 2) on sait que l'on veut calculer 185 termes, toujours si l'utilisateur a entré k=185), et une boucle "TantQue" quand on ne sait pas combien de fois on veut passer dans la boucle comme dans l'exemple de ce paragraphe.

4) Savoir déterminer le maximum d'une suite finie de nombres.

ex : soit V la suite définie par $V_1 = 2$ et $V_{n+1} = V_n + n - 10$.

Ecrire un algorithme qui permet de déterminer le maximum de cette suite quand n est entre 1 et 200, et donner l'index correspondant.

algorithme :

```
V=2
```

```
i=1   #1 est l'index du premier terme
max=2   #on initialise le max à la valeur du premier terme
pour k allant de 1 à 199    #le premier terme que l'on calcule est V2 avec k=1
        V= V +k – 10 #attention quand on calcule V2 = V1+1−10
                si V > max
                max = V  #si le terme que l'on vient de calculer est plus grand
                        #que le max obtenu jusque-là, c'est le nouveau max
                i=k+1  #on enregistre l'index du terme qui correspond max
FinSi
FinPour
afficher "le max est : ", max , " . Il a été obtenu au rang  ", i
```

Note : dans un affichage, ce qui est entre guillemets est une chaîne de caractères qui est affichée telle quelle, le reste, correspond à des variables dont on affiche la valeur. On sépare les différents objets à afficher par des virgules.
Dans notre exemple si le maximum est 237 et a été obtenu au rang 13 (pour V_{13}) on obtiendra l'affichage :
le max est : 237 . Il a été obtenu au rang 13

5) Savoir calculer la somme des termes d'une suite.

ex : On veut calculer $S_{1000} = U_0 + U_1 + U_2 + ... + U_{1000}$ où $U_n = \frac{3n-1}{n+2}$.

algorithme :

```
S=0  #on va calculer S petit à petit, en partant de 0
Pour i allant de 0 à 1000 #on va ajouter d'abord U0, puis U1 etc…..
        S = S + (3i−1)/(i+2)
Fin pour
Afficher S.
```

Note : dans ce type d'algorithmes, il est important d'initialiser correctement S au début, avant la boucle….

6) Simulation et compteur

De nombreux algorithmes simulent des situations réelles en utilisant le hasard : pour simuler un dé, on choisit un nombre au hasard entre 1 et 6, par exemple. C'est un outil indispensable pour programmer la grande majorité des jeux.

ex : on veut simuler une population dans laquelle 3% des personnes sont malades.
On veut prendre un échantillon de 2000 personnes et compter le nombre de malades.

algorithme :

```
c=0  #on initialise le compteur de malades à 0
pour i allant de 1 à 2000
```

```
        n= nombre aléatoire entre 1 et 100
        si n ≤ 3 #pour simuler 3%, si n est inférieur à 3, on est dans les malades
                c = c + 1  #on incrémente le nombre de malades de 1
        Finsi
FinPour
afficher c
```