Point de cours :

On considère les deux algorithmes suivants :

```
DEBUT
  S ← 0
  Pour i allant de 0 à 9
    Saisir n
    S ← S+n
  Fin Pour
  Afficher S
FIN
```

```
DEBUT
  S ← 0
  Pour i allant de 0 à 9
    Saisir T[i]
    S ← S+T[i]
  Fin Pour
  Afficher S
FIN
```

Ces deux algorithmes ont le même rôle mais ont des exécutions complètement différentes.

Dans le premier cas, il est impossible après l'exécution de l'algorithme de retrouver les valeurs qui ont été saisies. Dans le second cas, les valeurs saisies sont « enregistrées » au fur et à mesure dans un tableau.

CE QU'IL FAUT SAVOIR

Un tableau est un regroupement de variables. Chacune des variables est numérotée, ce numéro s'appelle un indice. Chaque variable est donc caractérisée par le nom du tableau et l'indice de son emplacement dans le tableau.

Déclaration

En pseudo-code, un tableau peut être déclaré comme une variable de la manière suivante :

```
Variables numériques : T(n)
## déclare un tableau contenant n variables de type numérique.
```

Accès aux éléments

Pour accéder aux éléments individuellement on utilise l'indice. Attention, en Python, la numérotation de ces indices commence à 0 et non à 1.

Par convention, nous commencerons la numérotation à 0 également en pseudo-code.

```
Algorithme : affichage
Variables numériques : T(10)
DEBUT
  T ← [5,4,3,6,9,7,1,2,0,8]
  Afficher T[5]
  Afficher T[9]
FIN
```

Cet algorithme initialise le tableau T, constitué de 10 valeurs numériques.

Indice i	0	1	2	3	4	5	6	7	8	9
$T[i]$	5	4	3	6	9	7	1	2	0	8

Il affichera 7 pour $T[5]$, 8 pour $T[9]$ mais une erreur si on appelle $T[10]$ car cette valeur n'existe pas.

Modification des éléments d'un tableau

Il est possible de modifier les éléments individuels d'un tableau :

Algorithme : modification tableau

Variables numériques : $T(10)$
DEBUT
  $T \leftarrow [5, 4, 3, 6, 9, 7, 1, 2, 0, 8]$
  $T[1] \leftarrow 2 \times T[1]$
  $T[7] \leftarrow T[0] + 3 \times T[6]$
  $T[5] \leftarrow 2 \times T[1]$
  Afficher T
FIN

Le tableau affiché est :

[5, 8, 3, 6, 9, 16, 1, 8, 0, 8]

Exemple 1 : Initialisation d'un tableau à valeurs numériques

On souhaite remplir un tableau avec des valeurs numériques. Le nombre de valeurs à saisir est choisi par l'utilisateur.

Algorithme : initialisation tableau

Variables numériques : dim, i, T(dim)
DEBUT
  Saisir dim
  Pour i **allant de** 0 **à** dim−1
    Saisir $T[i]$
  Fin Pour
  Afficher T
FIN

Pour initialiser un tableau en Python il y a plusieurs méthodes :

- **On initialise le tableau avec des 0 et on les modifie ensuite.**

Python : initialisation tableau 1

```
N=int(input("Saisissez la taille du tableau"))
T=[0]*N
for i in range(N):
    T[i]=float(input())
print(T)
```

- **On initialise le tableau vide et on ajoute ensuite une à une les valeurs.**

Python : initialisation tableau 2

```
N=int(input("Saisissez la taille du tableau"))
T=[]
for i in range(N):
    T.append(float(input())) # Ajoute la valeur saisie au clavier à la fin du tableau
print(T)
```

Exemple 2 : Parcours de tableau

On souhaite écrire un algorithme qui recherche la valeur maximale dans un tableau préalablement initialisé par l'utilisateur.

Algorithme : recherche de maximum

```
Variables numériques : dim,i, M,T(dim)
DEBUT
  M ← T[0] # On initialise le maximum à la première case du tableau
  Pour i allant de 0 à dim−1
    Si T[i] > M alors
      M ← T[i]
    Fin Si
  Fin Pour
  Afficher M
FIN
```

Python : recherche de maximum

```
...
# Le tableau a été initialisé au préalable
...
M=T[0]
N=len(T)          # len(T) est la taille du tableau
for i in range(N) :
  if T[i]>M :
    M=T[i]
print(M)
```

Exemple 3 : Recherche d'une valeur dans un tableau

On souhaite écrire un algorithme qui demande à l'utilisateur de choisir une valeur, puis affiche si cette valeur est ou non dans le tableau et, si oui, son indice.

Algorithme : recherche de valeur

```
Variables numériques : i, valeur, N, T(N)
DEBUT
  ...
  # Le tableau T a été initialisé au préalable avec une taille N
  ...
  Saisir valeur
  i ← 0
  Tant que T[i] ≠ valeur ET i < N
      i ← i +1
  Fin Tant que
  Si i = N alors
      Afficher « La valeur », valeur, « n'est pas dans le tableau. »
  Sinon
      Afficher « La valeur », valeur, « est dans le tableau à l'indice », i
  Fin Si
FIN
```

Remarque

Cet algorithme présente un problème de dépassement de dimension. Considérons que le tableau ne contienne pas la valeur recherchée. Lorsque $i = N - 1$, comme on a toujours $i < N$, on entre dans la boucle **Tant que**. La valeur de i est incrémentée et passe à N. Pour savoir si on entre encore dans la boucle **Tant que**, il faut pouvoir accéder à $T[N]$. Or cela n'est pas possible car le tableau contient des cases numérotées de 0 à $N - 1$. Cette erreur peut entraîner des problèmes d'exécution lors du passage sur machine.
Voici deux propositions de correction pour éviter ce problème.

- **Solution 1 : Utiliser une variable booléenne.**

Algorithme : recherche de valeur solution 1

```
Variables numériques : i, valeur, N, T(N)
DEBUT
  ...
  # Le tableau T a été initialisé au préalable avec une taille N
  ...
  Saisir valeur
  i ← 0
  b ← Faux
  Tant que i < N ET b = Faux
    Si T[i] = valeur alors
      b ← vrai
    Fin Si
    i ← i+1
  Fin Tant que
  Si b = Vrai alors
    Afficher « La valeur », valeur, « est dans le tableau à l'indice », i – 1
  Sinon
    Afficher « La valeur », valeur, « n'est pas dans le tableau. »
  Fin Si
FIN
```

Python solution 1

```
i=0
valeur=int(input())
b=False
while i<len(T) and b==False :
  if T[i]==valeur :
    b=True
  i=i+1
if b==True :
  print("La valeur", valeur, "est dans le tableau à l'indice", i-1)
else :
  print("La valeur", valeur, "n'est pas dans le tableau.")
```

- **Solution 2 : Traiter le cas $i = N - 1$ à part.**

Algorithme : recherche de valeur solution 2

```
Variables numériques : i, valeur, N, T(N)
DEBUT
  ...
  # Le tableau T a été initialisé au préalable avec une taille N
  ...
  Saisir valeur
  i ← 0
  Tant que T[i] ≠ valeur ET i < N – 1
    i ← i + 1
  Fin Tant que
  Si i = N – 1 ET T[i] = valeur alors
    Afficher « La valeur », valeur, « est dans le tableau à l'indice », i
  Sinon
    Si i = N – 1 alors
      Afficher « La valeur », valeur, « n'est pas dans le tableau. »
    Sinon
      Afficher « La valeur », valeur, « est dans le tableau à l'indice », i
    Fin Si
  Fin Si
FIN
```

Python solution 2

```
i=0
valeur=int(input())
while i<len(T)-1 and T[i] !=valeur :
  i=i+1
if i==len(T)-1 and T[i]==valeur :
  print("La valeur", valeur, "est dans le tableau à l'indice", i)
else :
  if i ==len(T)-1 :
    print("La valeur", valeur, "n'est pas dans le tableau")
  else :
    print("La valeur", valeur, "est dans le tableau à l'indice", i)
```

Application 1 : jeu à gratter

On considère un jeu à gratter où l'utilisateur découvre cinq nombres entiers aléatoires entre 1 et 5.

- Si la somme des nombres est paire, l'utilisateur gagne 1 euro.
- Si le plus grand nombre est 2, alors l'utilisateur gagne 2 euros.
- Si le même nombre apparaît 3 fois ou plus, l'utilisateur gagne 5 euros.
- Dans les autres cas, l'utilisateur ne gagne rien.
- Les différents gains se cumulent.

Écrire un algorithme qui simule le jeu à gratter et affiche le détail des gains du joueur.

Application 2 : tarifs de cotisation sportive

Pour s'inscrire en club sportif, le montant de la cotisation dépend de la catégorie :

Catégorie	Senior (21 ans et +)	Junior (18 à 20 ans	Cadet (16 à 17 ans)	Minime (14 à 15 ans)	Benjamin (12 à 13 ans)	Poussin (9 à 11 ans)	Pousset (7 à 8 ans)
Tarif	233 €	203 €	175 €	149 €	125 €	103 €	83 €

Écrire un algorithme qui demande à l'utilisateur sa catégorie puis affiche le montant de sa cotisation.

Indication pour la programmation en Python : on pourra utiliser deux tableaux ; un tableau *categorie* et un tableau *tarif*.

Application 3 : conseil de classe

En vue du conseil de classe, un professeur souhaite effectuer des statistiques sur sa classe de BTS SIO qui compte 30 élèves.

1. Écrire un algorithme qui permet la saisie des 30 notes dans un tableau.
2. Modifier cet algorithme pour qu'il calcule et affiche la moyenne des notes.
3. Modifier l'algorithme précédent pour qu'il renvoie la note maximale et la note minimale de la série de notes.
4. Modifier le programme pour qu'il affiche l'étendue de la série de notes (différence entre la note maximale et la note minimale).
5. Modifier le programme pour qu'il affiche le nombre de notes en dessous de 10/20.

Indication : Pensez à faire un parcours de tableau et à utiliser une variable compteur.

6. Modifier le programme pour qu'il calcule et affiche la médiane de la classe.

Indication : Attention, les indices commencent à 0 !

7. Modifier le programme pour qu'il calcule et affiche la moyenne élaguée (sans les notes maximales et minimales).

Remarque

En Python, il existe de nombreuses autres commandes pour manipuler les tableaux. On pourra tester par exemple les commandes suivantes.

- T=[5,8,7,1,3,6,4,9]
- len(T) # longueur
- sum(T) #somme
- max(T) #maximum
- min(T) #minimum
- T.append(2)
- T[2:5]
- 2 in T
- T[1 :]
- T.remove(3)
- T[:]
- T.index(6)