

LIFAPI – TD 1 : Algorithmes simples

Objectifs : Manipuler les notions de bases vues en CM 1

Définition de type, variable

Instruction, séquence d'instructions

Gestion des entrées / sorties

Structures de contrôle : condition, boucle, ...

Les instructions seront écrites uniquement **en langage algorithmique**

Séquence / bloc Début ... Fin Affectation $a \leftarrow 5$	Entrées /sorties utilisateur \rightarrow machine Lire (valeur) / Saisir (Valeur) utilisateur \leftarrow machine Afficher (valeur) / Ecrire (Valeur)
Conditionnelle Si condition alors instruction(s) Sinon instruction(s) FinSi	Boucle inconditionnelle Pour i allant de ... à ... par pas de ... faire instruction(s) FinPour
Boucle conditionnelle	
TantQue condition faire instruction(s) FinTantQue	Faire instruction(s) TantQue condition

1. Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ? Les deux versions qui contiennent les mêmes instructions donnent-elles le même résultat ?

```

Début
A, B, C : Entier
  A  $\leftarrow$  3
  B  $\leftarrow$  10
  C  $\leftarrow$  A + B
  B  $\leftarrow$  A * C
  A  $\leftarrow$  C + 4
Fin
  
```

```

Début
A, B, C : Entier
  A  $\leftarrow$  3
  B  $\leftarrow$  10
  C  $\leftarrow$  A + B
  A  $\leftarrow$  C + 4
  B  $\leftarrow$  A * C
Fin
  
```

Après La valeur des variables est :

A \leftarrow 3	A = 3	B = ?	C = ?
B \leftarrow 10	A = 3	B = 10	C = ?
C \leftarrow A + B	A = 3	B = 10	C = 13
B \leftarrow A * C	A = 3	B = 39	C = 13
A \leftarrow C + 4	A = 17	B = 39	C = 13

Si on change l'ordre des instructions (partie droite) on obtiendra le résultat suivant

Après La valeur des variables est :

A \leftarrow 3	A = 3	B = ?	C = ?
B \leftarrow 10	A = 3	B = 10	C = ?
C \leftarrow A + B	A = 3	B = 10	C = 13
A \leftarrow C + 4	A = 17	B = 39	C = 13
B \leftarrow A * C	A = 17	B = 221	C = 13

\Rightarrow L'ordre des instructions est très important ! Le résultat est différent si on permute des instructions.

2. Écrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui calcule et affiche le carré de ce nombre (en mémorisant le résultat puis sans le mémoriser ensuite).
Exemple : valeur saisie : 5 ➔ résultat affiché : 25

```
Début
  valeur, carre : entier
  Afficher('Donnez la valeur dont vous voulez calculer le carré')
  Saisir(valeur)
  carre ← valeur*valeur
  Afficher('le carré de', valeur, 'est', carre)
Fin
```

Bien constater la différence dans les affichages : chaîne de caractères / contenu de variable.

3. Écrire l'algorithme d'un programme qui demande un entier (âge) à l'utilisateur puis affiche "Majeur" si l'âge est supérieur ou égal à 18 et "Mineur" sinon.

```
Début
  age : entier // attention ne pas mettre d'accent dans le nom de la variable
  Afficher("Donnez l'âge de la personne")
  Saisir(age)
  si (age>=18) alors
    afficher ("personne majeure")
  sinon
    afficher("personne mineure" )
  fin si
Fin
```

4. Écrire l'algorithme d'un programme permettant de saisir puis d'afficher une valeur supérieure à 18 ; on recommencera la saisie tant que la valeur proposée ne convient pas. On écrira l'algorithme de **deux** manières différentes.

Première version avec un tant que ... faire

```
Début
  age : entier
  Afficher("Donnez l'âge de la personne")
  Saisir(age)
  Tant que (age<18) faire
    afficher ("la personne est mineure recommencez la saisie")
    saisir(age)
  fin tant que
  afficher (age, " ans est correspond bien à une personne majeure")
Fin
```

Ici on doit faire une première saisie (ou initialiser la variable age avec une valeur qui ne convient pas pour forcer à rentrer dans la boucle). La boucle sera réalisée 0 fois à n fois.

Deuxième version avec un faire Tant que

```
Début
  age : entier
  faire
    Afficher("Donnez l'âge de la personne (>=18)")
    Saisir(age)
  Tant que (age<18)
    afficher (age, " ans est correspond bien à une personne majeure")
  Fin
```

On fera au moins une fois la saisie.

Bien comprendre la différence avec le si qui ne fera qu'une seule fois le traitement.

5. Écrire l'algorithme d'un programme permettant d'afficher la table de multiplication d'un entier saisi par l'utilisateur.

Exemple : valeur saisie : 5 ➔ résultat affiché : 0 5 10 15 20 25 30 35 40 45 50

```
Début
  N, i : Entier
  Afficher('Entrez un nombre')
  Saisir(N)
  Afficher('La table de multiplication de ce nombre est')
  Pour i allant de 0 à 10 par pas de 1 faire
    Afficher(N*i, ' ')
  Fin Pour
Fin
```

Éventuellement, s'il vous reste du temps, leur faire le même algorithme avec un tant que pour leur montrer comment passer de l'un à l'autre.

Exercices pour aller plus loin...

1. Écrire un algorithme qui demande deux nombres entiers à l'utilisateur et l'informe ensuite si le produit est négatif, positif ou nul. Attention, on ne doit pas calculer le produit !

```
Début
  N1, N2 : Entiers
  Afficher ('Entrez nombre 1 :')
  Saisir (N1)
  Afficher ('Entrez nombre 2 :')
  Saisir (N2)
  Si ((N1 < 0 et N2 < 0) ou (N1 > 0 et N2 > 0))
    Alors  Afficher ('Le produit est positif.')
  Sinon  Si ((N1 < 0 et N2 > 0) ou (N1 > 0 et N2 < 0))
    Alors Afficher ('Le produit est négatif.')
    Sinon Afficher ('Le produit est nul.')
  Fin si
Fin si
Fin
```

2. Écrire un algorithme qui demande un nombre entier entre 1 et 31 (bornes incluses) ; on recommencera la saisie tant que la valeur n'est pas dans les bornes imposées.

Exemple : valeur saisie : 43 ➔ résultat affiché : valeur non comprise entre 1 et 31 recommencez... valeur saisie : 15 ➔ résultat affiché : affichage 15 ok !

```
Début
val : entier
Afficher('Donnez une valeur comprise entre 1 et 31')
Saisir (val)
Tant que ((val < 1 ) ou (val > 31)) faire
  Afficher('valeur non comprise entre 1 et 31 recommencez...')
  Saisir(val)
Fin tant que
Afficher('affichage ',val,' ok !')
Fin
```

Autre solution (à faire également pour leur montrer la différence entre les deux constructions)

```
Début
val : entier
Faire
  Afficher('Donnez une valeur comprise entre 1 et 31')
  Saisir (val)
Tant que ((val < 1 ) ou (val > 31))
  Afficher('affichage ',val,' ok !')
Fin
```