I. Parcours de boucles

Pour répéter plusieurs fois la même opération (exemple : parcourir une suite de nombres), on peut utiliser des boucles. Ainsi, on ne décrit qu'une seule fois la série d'instructions qui se répéteront, et on précise la condition de fin de parcours de la boucle.

A. Boucle for (pour)

Exercice 1 : On considère l'algorithme suivant.

```
Début
Entiers i, factorielle = 1 ;
Pour i allant de 1 à 4 faire
   factorielle = factorielle * i ;
FinPour
Afficher factorielle ;
Fin
```

1. À combien de parcours de boucle cet algorithme va-t-il procéder ?
2. Quelle valeur prend la variable `factorielle` en fin d'algorithme ?
3. Préciser l'ensemble du calcul permettant d'arriver à ce résultat.

Exercice 2 : Écrire un algorithme en langage naturel permettant de calculer la somme des n premiers entiers.

B. Boucle while (tant que)

Exercice 3 : On considère l'algorithme suivant.

```
Début
Entier i = 1 ;
Entier produit = i ;
Tant que produit < 100 faire
   produit = produit * i ;
   i = i + 1 ;
FinTantQue
Afficher produit ;
Fin
```

1. Détailler les valeurs de `i` et de `produit` à chaque fin de parcours de boucle.
2. À combien de parcours de boucle cet algorithme a-t-il procédé ?

Remarque : Les boucles while et do...while

Exercice 4 : Combien de parcours de boucles l'algorithme suivant fait-il ?

```
Début
Entier i = 5 ;
Entier produit = i ;
Faire
   produit = produit * i ;
   i = i + 1 ;
   Tant que produit < 10 ;
FinTantQue
Afficher produit ;
Fin
```

Conclusion : la boucle *do...while* assure le parcours de la boucle au moins une fois. Pour la lisibilité du code, il vaut mieux lui préférer la boucle *while*.

II. Structures conditionnelles

Il est également possible de concevoir des morceaux d'algorithmes qui ne seront appliqués que sous certaines conditions.

A. Structure conditionnelle If... then...else (si... alors...)

Exercice 5 : On considère l'algorithme suivant.

```
Début
Entier i ;
Afficher « Entrer un nombre entier »
Saisir i ;
Si i modulo 2 est égal à 0 faire
   Afficher « Ce nombre est pair »
Sinon faire
   Afficher « Ce nombre est impair »
FinSi
Fin
```

1. Quelle partie de l'algorithme est appliquée si la valeur saisie par l'utilisateur vaut 9 ?
2. Quelle partie de l'algorithme est appliquée si la valeur saisie par l'utilisateur vaut 6 ?

B. Structure conditionnelle switch

Exercice 6 : On considère l'algorithme suivant.

```
Début
Entier i ;
Afficher « Entrez votre classement aux jeux olympiques »
Saisir i ;
Switch (i)
   Cas i vaut 1 faire
      Afficher « Vous avez la médaille d'or »
   Cas i vaut 2 faire
      Afficher « Vous avez la médaille d'argent »
   Cas i vaut 3 faire
      Afficher « Vous avez la médaille de bronze »
   Cas par défaut
      Afficher « Vous n'êtes pas médaillé »
FinSwitch
Fin
```

1. Comment fonctionne cet algorithme ?
2. Quel gain apporte-t-il par rapport à une structure conditionnelle if/then/else ?

III. Conception d'algorithmes

Exercice 7 : Écrire en langage naturel un algorithme qui détermine le plus grand de deux entiers saisis.

Exercice 8 :

1. Écrire en langage naturel un algorithme qui détermine et affiche le plus petit entier n tel que 2^n est supérieur à 1000.
2. Modifier cet algorithme pour que la valeur limite (précédemment égale à 1000) soit saisie par l'utilisateur.

Exercice 9 : Écrire en langage naturel un algorithme qui demande à l'utilisateur de saisir un nombre entier n inférieur à 10 et qui affiche un carré de x de côté n. Par exemple, pour n = 3, le résultat affiché sera :

x x x
x x x
x x x

Exercice 10 : Écrire en langage naturel un algorithme qui demande à l'utilisateur de saisir un nombre entier n compris entre 0 et 9, et qui affiche ce nombre en toutes lettres. Par exemple, si n = 3, le résultat affiché sera *trois*.
Si le nombre saisi n'est pas compris entre 0 et 9, le résultat affiché sera *Erreur de saisie*.