Les structures

Structure alternative (instruction conditionnelle)

SI … ALORS … SINON … FSI

Si l'expression logique (la condition) prend la valeur **vrai** (est vérifiée), le premier bloc d'instructions est exécuté; si elle prend la valeur **faux** (non vérifiée), le second bloc est exécuté (s'il est présent, sinon rien ne se fait).

si *<expression logique>*
    alors *instructions*
*[**sinon** instructions]*
fsi

Exercice

Saisir une valeur entière et afficher son double si cette donnée est inférieure à un seuil donné.

Structures alternatives imbriquées

si *<expression logique>*
    alors *instructions*
sinon ***si*** *<expression logique>*
        alors *instructions*
        sinon *instructions*
    fsi
fsi

Exercice

Saisir une valeur entière et afficher :

- "Reçu avec mention Assez bien" si une note est supérieure ou égale à 12,
- "Reçu avec mention Passable" si une note est supérieure à 10 et inférieure à 12,
- "Insuffisant" dans tous les autres cas.

Sélection choix multiples

SELON

S'il y a plus de 2 choix possibles, l'instruction **SELON** permet une facilité d'écriture.
De plus, cette instruction permet de ne pas se soucier de l'ordre de la liste de valeur.
Elle est appropriée pour remplacer efficacement une structure alternative imbriquée avec un identificateur unique, ou une structure alternative séquentielle avec un identificateur unique.

selon *<identificateur>*
(liste de) valeur(s) : instruction(s)
(liste de) valeur(s) : instruction(s)
…
*[**autres** : instruction(s)]*

Exercice

Reproduire avec l'instruction **SELON** l'équivalence de cette structure conditionnelle :

```
variables       abreviation: chaîne de caractères

début
        {préparation du traitement}
        afficher("Civilité (Mme / Mlle / M / Autre) :")
        saisir(abreviation )

        si abreviation = "Mme"
                alors afficher("Madame")
        sinon  si abreviation = "Mlle"
                        alors afficher("Mademoiselle")
                sinon  si abreviation = "M"
                                alors afficher("Monsieur")
                        sinon afficher("Transgenre")
                        fsi
                fsi
        fsi
fin
```

Répétition d'un traitement

Boucle POUR

Sa fonction est de répéter une suite d'instructions un certain nombre de fois. Elle s'utilise dans le cas ou le nombre d'itération est connu.
L'instruction **POUR**:

1. initialise une variable de boucle (le compteur),
2. incrémente cette variable avec une valeur de "pas",
3. vérifie que cette variable ne dépasse pas le borne supérieure.

Attention *:* le traitement ne doit pas modifier la variable de boucle !

pour *<variable>* ← *<valeur initiale>* ***à*** *<valeur finale>* ***[par*** *<pas>]* ***faire***
    traitement {suite d'instructions}
fpour

Exercice

En fonction d'un nombre d'itérations saisi, faire la somme des entiers saisis et afficher le résultat de l'opération.

Boucle TANT QUE … FAIRE

Sa fonction est de répéter une suite d'instructions un certain nombre de fois.
C'est la structure itérative universelle (n'importe quel contrôle d'itération peut se faire par le "tant que") permettant la répétition d'un traitement à nombre d'itération inconnu.
C'est une structure itérative irremplaçable dès que la condition d'itération devient complexe.

amorçage {initialisation de la(des) variable(s) de condition}
tant que *<expression logique (vraie)>* ***faire***
    traitement {suite d'instructions à exécuter si condition vraie}
    relance {ré-affectation de la(des) variable(s) de condition}
ftq

Exercice

Reproduire l'algorithme de la boucle **POUR** avec l'instruction **TANT QUE … FAIRE**.
L'expression logique d'arrêt sera la saisie de la valeur "-1".

Comparaison des boucles POUR et TANT QUE … FAIRE

Implicitement, l'instruction POUR :

- initialise un compteur,
- incrémente le compteur à chaque pas,
- vérifie que le compteur ne dépasse pas la borne supérieure.

Explicitement, l'instruction TANT QUE … FAIRE doit :

- initialiser un compteur (amorçage),
- incrémenter le compteur à chaque pas (relance),
- vérifier que le compteur ne dépasse pas la borne supérieure (test de boucle).

Afin de choisir au mieux la boucle à utiliser, rappelez-vous que :

- pour un nombre d'itération connu à l'avance, il vaut mieux utiliser la boucle POUR (parcours de tableaux, test sur un nombre donné de valeurs, …),
- pour un arrêt de la boucle sur un événement particulier, il vaut mieux utiliser TANT QUE … FAIRE.

Boucle RÉPÉTER ... TANT QUE

Sa fonction est d'exécuter une suite d'instructions au moins un fois et la répéter tant que condition est remplie. Contrairement aux autre boucles (et notamment à la boucle TANT QUE … FAIRE), la boucle RÉPÉTER … TANT QUE lira une première fois la suite d'instructions avant d'exécuter un test.

répéter
    (ré)affectation de la(des) variable(s) de condition
    traitement {suite d'instructions}
tant que *<expression logique (vraie)>*

La boucle RÉPÉTER … TANT QUE est donc typique pour des saisies avec vérification.

Exercice

Saisir des données et s'arrêter dès que leur somme dépasse 500.

Mise en pratique

Exercice 0

Écrire un algorithme qui échange la valeur de deux variables.
Exemple, si a ← 2 et b ← 5, le programme donnera a ← 5 et b ← 2.

Exercice 1

Écrire un algorithme qui demande deux nombres à l'utilisateur et l'informe ensuite si leur produit est négatif ou positif (on laisse de coté le cas où le produit est nul).
Attention toutefois : on ne doit pas calculer le produit des deux nombres.

Exercice 2

Écrire un algorithme qui demande un nombre compris entre 10 et 20, jusqu'à ce que la réponse convienne.
En cas de réponse supérieure à 20, on fera apparaître un message : "Plus petit !" , et inversement, "Plus grand !" si le nombre est inférieur à 10.