Cours n° 2 : Les instructions de base dans un algorithme

2.1 Introduction

Les instructions dans un algorithme sont les ordres que le programmeur donne à la machine et dont l'exécution permet d'obtenir les résultats voulus. Dans ce qui suit, nous allons découvrir les trois instructions de base de la programmation.

Prenons un exemple de problème géométrique : le calcul de la surface d'un rectangle. Nous allons réfléchir sur les différentes étapes que la machine doit faire pour pouvoir donner la surface de n'importe quel rectangle.

Notons qu'ici le programmeur n'a aucune idée sur les données chiffrées du problème, c'est-à-dire les valeurs numériques de la longueur et de la largeur du rectangle dont la machine calculera la surface. Ces valeurs se trouvent chez l'utilisateur qui lui, viendra demander l'exécution du programme, donner ces valeurs et attendre comme résultat la surface de son rectangle.

2.2 La demande des données

Le programmeur devra ordonner en 1er lieu à la machine de demander à l'utilisateur d'introduire ses deux valeurs. Il devra également désigner deux variables qui pourront les contenir au niveau de la mémoire centrale. Nous les appellerons « longueur » et « largeur » ou même en abréviation « long » et « larg ». Donc les 2 premières instructions à faire exécuter par la machine sont bien :

demander (long) ; demander (larg) ;

Notons que le verbe « demander » est écrit à l'infinitif mais il faut le comprendre à la forme impérative. « demander (long) ; » sous-entends que le programmeur dit à la machine : **demande** à l'utilisateur la valeur de long.

2.3 Le calcul du résultat

Une fois la machine possède les deux valeurs « long » et « larg », elle peut maintenant procéder au calcul de la surface, pour laquelle nous désignons une autre variable « surface » ou même en abrégé « surf ». Nous mettrons alors juste après la 2ème instruction la formule

surf = long * larg ;

Notons ici que le symbole de multiplication est bien le « * » et non pas le « x » habituel.

2.3 L'affichage du résultat

Nous nous approchons alors de la fin de notre algorithme, car nous arrivons au moment où la machine doit afficher sur écran (et donc pour l'utilisateur) la valeur de la surface calculée dans l'étape précédente. Ça sera la dernière instruction de l'algorithme :

afficher (surf) ;

Nous pouvons reconstituer maintenant notre algorithme en déclarant toutes les variables utilisées et en lui donnant un identificateur (un nom) qui doit être différent des identificateurs choisis pour les variables. Ainsi nous ne pourrons pas nommer notre algorithme « surf », car cet identificateur avec cette orthographe a été déjà attribué à la variable résultat.

```
Algo surface ;
Var long, larg, surf : réel ;
Début
    demander (long) ;
    demander (larg) ;
    surf = long * larg ;
    afficher (surf) ;
Fin.
```

2.4 L'instruction de lecture

Observons maintenant la 1ère instruction de l'algorithme « ***demander (long) ;*** ». En réalité la demande de la machine se traduit par ***une simple attente*** que l'utilisateur introduise par le biais de son clavier une valeur numérique qui s'affiche sur son écran. Il valide sa saisie en appuyant sur la touche « entrée », et c'est à ce moment que la machine récupère la valeur saisie et la place dans la case mémoire réservée pour la variable « long ». Une fois la valeur saisie est placée dans sa case, on dit que **la demande a été exécutée**.

Cette demande revient alors en réalité à faire une ***lecture*** de ce que l'utilisateur écrit sur son clavier. Et à partir de ce moment, l'instruction « ***demander (long) ;*** » deviendra :

lire (long) ;

Remarques :

- Evidemment l'instruction « lire » est utilisée uniquement avec des variables. Une lecture d'une constante est catégoriquement rejetée par la machine.
- L'instruction lire est toujours suivie par des parenthèses obligatoires, à l'intérieur desquelles nous mettons la variable ou la liste de variables à lire. Dans ce dernier cas, les variables sont séparées par des virgules. Les exemples exp1 et exp2 donnent le même résultat.
- Il faudra faire attention à ne pas mettre des variables résultats dans l'instruction « lire ». Seules les variables qui représentent des données qui sont chez l'utilisateur doivent être placées dans l'instruction de lecture.

```
Exp1 :  lire (long); lire (larg);

Exp2 :  lire (long, larg) ;
```

2.5 L'instruction d'affectation

Observons maintenant la formule de calcul que nous avons mise dans l'algorithme : « ***surf = long * larg ;*** », le symbole « **=** » est en réalité un symbole de comparaison. Il confirme tout simplement que la partie à gauche de ce symbole est égale à la partie droite du même symbole. Donc cette formule ne porte pas en réalité une notion d'ordre à la machine. C'est pour cette raison que nous allons remplacer le symbole « **=** » par le symbole « **←** », donc une flèche qui va de droite à gauche et qui signifie un ordre de **calculer** l'expression à droite du symbole et d'**affecter** son résultat dans la variable mentionnée à gauche du symbole.

surf ← long * larg ;

L'instruction d'affectation n'accepte jamais une constante à sa gauche, pour la simple raison que la valeur de celle-ci est déjà mentionnée au niveau de la déclaration et elle ne peut pas changer pendant l'exécution du programme.

2.6 L'instruction d'écriture

L'instruction **« afficher (surf) ; »** de notre algorithme écrira simplement le résultat sur écran pour permettre à l'utilisateur de le lire. C'est une instruction d'écriture et à partir de ce moment elle devient **« écrire (long) ; »**

```
Exp 1 :  écrire (surface);
         écrire (périmètre);
Exp 2 :  écrire (surface, périmètre) ;
```

Maintenant nous pouvons donner un algorithme complet qui calcule la surface d'un rectangle.

```
Algo surface ;
Var long , larg, surf : réel ;
Début
    Lire (long) ;
    Lire (larg) ;
    Surf ← long * larg ;
    Ecrire (surf) ;
Fin.
```

Ci-dessous le tableau des équivalences de ces nouvelles instructions en pascal.

Algorithme	**Pascal**
Lire	Read
Ecrire	Write
←	:=

Et le programme pascal correspondant à l'algorithme « surface » sera :

```
Program surface ;
Var long , larg, surf : real ;
Begin
    Read (long) ;
    Read (larg) ;
    Surf := long * larg ;
    Write (surf) ;
End.
```

Application.

Nous allons maintenant considérer un 2ème exemple qui nous permet de voir la différence entre le traitement des variables et celui des constantes. Nous voulons faire 2 algorithmes qui calculent la moyenne d'un étudiant qui a 3 notes de 3 modules différents ayant des coefficients différents. Dans le 1er nous considérons les coefficients comme étant des constantes (valeurs aux choix), et dans le 2ème nous les considérons comme variables.

```
Algo Moyenne1 ;
Const C1 = 3 ; C2 = 1 ; C3 = 2 ;
Var N1, N2, N3, M : réel ;
Début
    Lire (N1, N2, N3) ;
    M ←(N1*C1 + N2*C2 + N3*C3) / (C1+C2+C3) ;
    Ecrire (M) ;
Fin.
```

```
Algo Moyenne2 ;
Var N1, N2, N3, M : réel ;
     C1, C2, C3 : entier ;
Début
    Lire (N1, N2, N3) ;
    Lire (C1, C2, C3) ;
    M ← (N1*C1 + N2*C2 + N3*C3) / (C1+C2+C3) ;
    Ecrire (M) ;
Fin.
```

2.7 Les messages

Les messages sont des phrases en langage naturel que le programme affiche à l'utilisateur pour lui apporter des clarifications. Nous pouvons les placer avant une instruction de lecture pour informer l'utilisateur sur les valeurs attendues par le programme, et avant l'affichage d'un résultat pour expliquer ce que la valeur résultat représente.

Nous reprenons l'algorithme précédent en intégrant des messages pour l'utilisateur :

```
Algo Moyenne2 ;
Var N1, N2, N3, M : réel ;
      C1, C2, C3 : entier ;
Début
    Ecrire('Donnez la note du 1er module') ;
    Lire (N1) ;
    Ecrire('Donnez la note du 2ème module') ;
    Lire (N2) ;
    Ecrire('Donnez la note du 3ème module') ;
    Lire (N3) ;
    Ecrire('Donnez le coefficient du 1er module') ;
    Lire (C1) ;
    Ecrire('Donnez le coefficient du 2ème module') ;
    Lire (C2) ;
    Ecrire('Donnez le coefficient du 3ème module') ;
    Lire (C3) ;
    M ← (N1*C1 + N2*C2 + N3*C3) / (C1+C2+C3) ;
    Ecrire ('La moyenne de cet étudiant = ') ;
    Ecrire (M) ;
Fin.
```

2.8 Les fonctions DIV et MOD

La fonction **DIV** permet de faire une division « entière », c'est-à-dire d'avoir un quotient de type entier. Il est évident que la division « entière » provoque presque toujours des restes. Cette fonction est reconnue par le pascal et nous pouvons l'utiliser à chaque fois que nous nous intéressons à avoir juste la partie entière du quotient résultat d'une division. Nous pouvons l'appeler dans une instruction de cette manière :

A ← B ***DIV*** *C* ***;***

La fonction **MOD** est reconnue aussi par le pascal et donne quant à elle le reste de la division entière. Nous pouvons l'appeler dans une instruction de la manière suivante :

A ← B ***MOD*** *C* ***;***

Exemples de division :

13 / 2 = 6,5
13 DIV 2 = 6
13 MOD 2 = 1

Les 2 fonctions DIV et MOD travaillent uniquement et par évidence avec les variables de type entier.

<u>Exemple d'utilisation</u> : On veut un algorithme qui, pour un nombre donné en secondes, nous affiche le nombre d'heures, de minutes et de secondes incluses dans le nombre initial.

```
Algo Temps ;
Var Heure, Minute, Seconde, Nombre, Reste : entier ;
Début
    Ecrire ('Donnez votre nombre initial') ;
    Lire (Nombre) ;
    Heure ← Nombre Div 3600;
    Reste ← Nombre Mod 3600 ;
    Minute ← Reste Div 60 ;
    Seconde ← Reste Mod 60 ;
    Ecrire ('Nombre dheures = ',Heure) ;
    Ecrire ('Nombre de minutes = ',Minute) ;
    Ecrire ('Nombre de secondes = ',Seconde) ;
Fin.
```