A.1 Variables et affectations

Les variables en algorithmique

— Les variables algorithmiques peuvent servir à stocker des données de différents types, mais nous nous contenterons ici d'utiliser des variables du type `NOMBRE`.
— La valeur d'une variable peut changer au fil des instructions de l'algorithme.
— Les opérations sur les variables s'effectuent ligne après ligne et les unes après les autres.
— Quand l'ordinateur exécute une ligne du type `mavariable PREND_LA_VALEUR un calcul`, il effectue d'abord le calcul et stocke ensuite le résultat dans `mavariable`.

► Activité n°1
On considère l'algorithme suivant :

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: y EST_DU_TYPE NOMBRE
4: z EST_DU_TYPE NOMBRE
5: DEBUT_ALGORITHME
6:   x PREND_LA_VALEUR 2
7:   y PREND_LA_VALEUR 3
8:   z PREND_LA_VALEUR x+y
9: FIN_ALGORITHME
```

Après exécution de l'algorithme :

— La variable x contient la valeur : ☐
— La variable y contient la valeur : ☐
— La variable z contient la valeur : ☐

► Activité n°2
On considère l'algorithme suivant :

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: DEBUT_ALGORITHME
4:   x PREND_LA_VALEUR 2
5:   x PREND_LA_VALEUR x+1
6: FIN_ALGORITHME
```

Après exécution de l'algorithme :

— La variable x contient la valeur : ☐

► Activité n°3
Ajoutons la ligne « `x PREND_LA_VALEUR 4*x` » à la fin du code précédent. Ce qui donne :

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: DEBUT_ALGORITHME
4:   x PREND_LA_VALEUR 2
5:   x PREND_LA_VALEUR x+1
6:   x PREND_LA_VALEUR 4*x
7: FIN_ALGORITHME
```

Après exécution de l'algorithme :

— La variable x contient la valeur : ☐

► Activité n°4
On considère l'algorithme suivant :

```
1: VARIABLES
2: A EST_DU_TYPE NOMBRE
3: B EST_DU_TYPE NOMBRE
4: C EST_DU_TYPE NOMBRE
5: DEBUT_ALGORITHME
6:    A PREND_LA_VALEUR 5
7:    B PREND_LA_VALEUR 3
8:    C PREND_LA_VALEUR A+B
9:    B PREND_LA_VALEUR B+A
10:   A PREND_LA_VALEUR C
11: FIN_ALGORITHME
```

Après exécution de l'algorithme :

— La variable A contient la valeur : ☐

— La variable B contient la valeur : ☐

— La variable C contient la valeur : ☐

► Activité n°5
On considère l'algorithme suivant :

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: y EST_DU_TYPE NOMBRE
4: z EST_DU_TYPE NOMBRE
5: DEBUT_ALGORITHME
6:    LIRE x
7:    y PREND_LA_VALEUR x-2
8:    z PREND_LA_VALEUR -3*y-4
9:    AFFICHER z
10: FIN_ALGORITHME
```

On cherche maintenant à obtenir un algorithme équivalent sans utiliser la variable y. Compléter la ligne 6 dans l'algorithme ci-dessous pour qu'il réponde au problème.

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: z EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:    LIRE x
6:    z PREND_LA_VALEUR .....................
7:    AFFICHER z
8: FIN_ALGORITHME
```

A.2 Instructions conditionnelles

SI...ALORS...SINON

Comme nous l'avons vu ci-dessus, un algorithme permet d'exécuter une liste d'instructions les unes à la suite des autres. Mais on peut aussi "dire" à un algorithme de n'exécuter des instructions que si une certaine condition est remplie. Cela se fait grâce à la commande `SI...ALORS` :

```
SI...ALORS
  DEBUT_SI
  ...
  FIN_SI
```

Il est aussi possible d'indiquer en plus à l'algorithme de traiter le cas où la condition n'est pas vérifiée. On obtient alors la structure suivante :

```
SI...ALORS
  DEBUT_SI
  ...
  FIN_SI
  SINON
    DEBUT_SINON
    ...
    FIN_SINON
```

► Activité n°6

On cherche à créer un algorithme qui demande un nombre à l'utilisateur et qui affiche la racine carrée de ce nombre s'il est positif. Compléter la ligne 6 dans l'algorithme ci-dessous pour qu'il réponde au problème.

```
 1: VARIABLES
 2: x EST_DU_TYPE NOMBRE
 3: racine EST_DU_TYPE NOMBRE
 4: DEBUT_ALGORITHME
 5:   LIRE x
 6:   SI (.........) ALORS
 7:     DEBUT_SI
 8:     racine PREND_LA_VALEUR sqrt(x)
 9:     AFFICHER racine
10:     FIN_SI
11: FIN_ALGORITHME
```

► Activité n°7

On cherche à créer un algorithme qui demande à l'utilisateur d'entrer deux nombres (stockés dans les variables `x` et `y`) et qui affiche le plus grand des deux. Compléter les ligne 9 et 13 dans l'algorithme ci-dessous pour qu'il réponde au problème.

```
 1: VARIABLES
 2: x EST_DU_TYPE NOMBRE
 3: y EST_DU_TYPE NOMBRE
 4: DEBUT_ALGORITHME
 5:   LIRE x
 6:   LIRE y
 7:   SI (x>y) ALORS
 8:     DEBUT_SI
 9:     AFFICHER .......
10:     FIN_SI
11:   SINON
12:     DEBUT_SINON
13:     AFFICHER ......
14:     FIN_SINON
15: FIN_ALGORITHME
```

► Activité n°8
On considère l'algorithme suivant :

```
1: VARIABLES
2: A EST_DU_TYPE NOMBRE
3: B EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:   A PREND_LA_VALEUR 1
6:   B PREND_LA_VALEUR 3
7:   SI (A>0) ALORS
8:     DEBUT_SI
9:     A PREND_LA_VALEUR A+1
10:    FIN_SI
11:  SI (B>4) ALORS
12:    DEBUT_SI
13:    B PREND_LA_VALEUR B-1
14:    FIN_SI
15: FIN_ALGORITHME
```

Après exécution de l'algorithme :

— La variable A contient la valeur : ☐

— La variable B contient la valeur : ☐

► Activité n°9
On cherche à concevoir un algorithme correspondant au problème suivant :

— on demande à l'utilisateur d'entrer un nombre (qui sera représenté par la variable x)

— si le nombre entré est différent de 1, l'algorithme doit stocker dans une variable y la valeur de 1/(x-1) et afficher la valeur de y (note : la condition x différent de 1 s'exprime avec le code x!=1). On ne demande pas de traiter le cas contraire.

Compléter l'algorithme ci-dessous pour qu'il réponde au problème.

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: y EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:   LIRE ......
6:   SI (........) ALORS
7:     DEBUT_SI
8:     ....... PREND_LA_VALEUR .......
9:     AFFICHER .......
10:    FIN_SI
11: FIN_ALGORITHME
```