Initiation à la programmation
Exercices

I/ Ecrire, exécuter et interpréter un programme

Pour les exercices 1 à 5, répondre aux questions suivantes :
1) Ecrire, sur AlgoBox, le programme présenté.
2) Exécuter le programme puis donner la valeur (en fin d'exécution du programme) de chaque variable déclarée.
On pourra compléter le code pour afficher les valeurs des variables en fin d'exécution.

Exercice 1

```
VARIABLES
  x EST_DU_TYPE NOMBRE
  y EST_DU_TYPE NOMBRE
  z EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  x PREND_LA_VALEUR 2
  y PREND_LA_VALEUR 3
  z PREND_LA_VALEUR x+y
FIN_ALGORITHME
```

Exercice 2

```
VARIABLES
  x EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  x PREND_LA_VALEUR 2
  x PREND_LA_VALEUR x+1
  x PREND_LA_VALEUR 4*x
FIN_ALGORITHME
```

Exercice 3

```
VARIABLES
  a EST_DU_TYPE NOMBRE
  b EST_DU_TYPE NOMBRE
  c EST_DU_TYPE NOMBRE
  n EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  a PREND_LA_VALEUR 17
  n PREND_LA_VALEUR 2
  b PREND_LA_VALEUR a+a
  c PREND_LA_VALEUR pow(a,2)
  b PREND_LA_VALEUR a*c
  c PREND_LA_VALEUR a/b
  b PREND_LA_VALEUR a-c
FIN_ALGORITHME
```

Exercice 4

```
VARIABLES
  a EST_DU_TYPE NOMBRE
  b EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  a PREND_LA_VALEUR 1
  b PREND_LA_VALEUR 3
  SI (a>0) ALORS
    DEBUT_SI
      a PREND_LA_VALEUR a+1
    FIN_SI
  SI (b > 4) ALORS
      DEBUT_SI
        b PREND_LA_VALEUR b-1
      FIN_SI
FIN_ALGORITHME
```

Exercice 5

```
VARIABLES
  a EST_DU_TYPE NOMBRE
  b EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  a PREND_LA_VALEUR 1
  b PREND_LA_VALEUR 1
  TANT_QUE (b<=24) FAIRE
    DEBUT_TANT_QUE
    a PREND_LA_VALEUR b+a
    b PREND_LA_VALEUR a+b
    FIN_TANT_QUE
FIN_ALGORITHME
```

II/ Exercices sur les variables

Exercice 6

Ecrire un algorithme sur AlgoBox qui demande à l'utilisateur d'affecter une variable *p* et dont le résultat en exécution est comme décrit à droite.

```
***Algorithme lancé***
Entrer p : 17
p=20
***Algorithme terminé***
```

Exercice 7

Voici un programme de calcul :

- Choisir un nombre
- Lui ajouter 4
- Multiplier la somme obtenue par le nombre choisi
- Ajouter 4
- Ecrire le résultat obtenu.

1) Ecrire un algorithme sur AlgoBox qui traduit le programme de calcul ci-dessus.
2) Donner le résultat obtenu après exécution du programme de calcul si le nombre choisi au départ est égal à 4, à 10 et à 100.

Exercice 8

1) Ecrire un algorithme (sur AlgoBox) qui calcule le périmètre d'un cercle et l'aire d'un disque dont on donne le rayon.
2) Tester votre programme pour différentes valeurs de rayon.
3) Pour quelle valeur du rayon, la valeur du périmètre du cercle est égale à la valeur de l'aire de son disque ?

Exercice 9

1) Ecrire un algorithme (sur AlgoBox) qui échange le contenu de deux variables *a* et *b.*
2) Tester votre programme pour différentes valeurs de *a* et de *b*.

Exercice 10

Dans un plan muni d'un repère orthonormé, on considère deux points $A(x_A ; y_A)$ et $B(x_B ; y_B)$. Les coordonnées du point M milieu du segment [AB] sont données par :

$$M\left(\frac{x_A + x_B}{2} ; \frac{y_A + y_B}{2}\right)$$

1) Ecrire un algorithme (sur AlgoBox) qui calcule les coordonnées du milieu d'un segment [AB].
2) On donne A(1;3) et B(-2 ;-1). Quelles sont les coordonnées du point M milieu du segment [AB] ? Le vérifier par exécution de votre programme et sur GeoGebra.

III/ Structure conditionnelle (SI ALORS SINON)

Exercice 11

On appelle *valeur absolue* d'un nombre *x* : $|x| = \begin{cases} x & \text{si } x \geq 0 \\ -x & \text{si } x < 0 \end{cases}$

Ecrire un algorithme qui permet de calculer la valeur absolue d'un nombre x donné par l'utilisateur (**sans utiliser la fonction** *abs(x)*)

Exercice 12

L'utilisateur vous donne un nombre entier quelconque.
S'il est pair, le programme renvoie son nombre divisé par 2
S'il est impair, le programme renvoie son nombre multiplié par 3 puis on rajoute 1 au résultat final.

Ecrire un algorithme (sur AlgoBox) qui répond au problème.

Exercice 13

On donne trois longueurs de même unité, *a, b* et *c.* Ecrire un algorithme qui permet de dire si on peut construire le triangle de longueurs *a,* b et *c.*

Pour aller plus loin : Déterminer si ce triangle est isocèle, équilatéral.
(Pour les quatrièmes, déterminer si ce triangle est rectangle).

IV/ Structure répétitive (TANTQUE/POUR)

Exercice 14

On considère le problème suivant :
– On lance une balle d'une hauteur initiale de 300 cm.
– On suppose qu'à chaque rebond, la balle perd 10% de sa hauteur (la hauteur est donc multipliée par 0.9 à chaque rebond).
– On cherche à savoir le nombre de rebonds nécessaire pour que la hauteur de la balle soit inférieure ou égale à 10 cm.

Compléter l'algorithme suivant pour qu'il réponde au problème.

```
VARIABLES
  nombre_rebonds EST_DU_TYPE NOMBRE
  hauteur EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  nombre_rebonds PREND_LA_VALEUR 0
  hauteur PREND_LA_VALEUR 300
  TANT_QUE (hauteur .......) FAIRE
    DEBUT_TANT_QUE
    nombre_rebonds PREND_LA_VALEUR nombre_rebonds+1
    hauteur PREND_LA_VALEUR .......
    FIN_TANT_QUE
    AFFICHER nombre_rebonds
FIN_ALGORITHME
```

Exercice 15

Ecrire un algorithme qui permet de calculer la somme des 100 premiers entiers naturels.

V/ Programmer des jeux simples

Exercice 16 (Le lièvre et la tortue)

Le jeu du lièvre et de la tortue est le suivant :

- A chaque tour, on lance un dé ;
- si le 6 sort, le lièvre gagne la partie, sinon la tortue avance d'une case ;
- la tortue gagne quand elle avance 6 fois.

1) Ecrire un algorithme (sur AlgoBox) qui modélise le jeu du lièvre et la tortue.
2) Quand on fait joue un grand nombre de fois à ce jeu, qui gagne le plus souvent ?

Exercice 17 (Marche aléatoire d'un robot)

Un robot marche aléatoirement sur un axe gradué régulièrement. Il démarre de l'origine de l'axe, il choisit un sens de déplacement aléatoire (soit vers les positifs, soit vers les négatifs) et il se déplace vers le nombre entier supérieur ou inférieur au nombre entier de départ (selon son sens de déplacement).

Exemple de marche aléatoire :

```
***Algorithme lancé***

Entrer bornes : 10
Déplacements: 0 -1 0 1 2 1 0 1 2 1 0 1
2 3 4 5 6 5 4 3 4 5 4 5 4 3 4 5 4 3 4 5
6 5 4 3 4 3 4 5 6 7 8 7 6 7 6 5 6 7 6 7
8 9 8 7 6 5 6 7 6 7 6 7 6 7 6 5 4 3 2 3
4 3 4 3 2 3 4 3 4 5 6 7 8 9 8 7 6 7 8 9
10
Nombre de déplacements: 92
***Algorithme terminé***
```

1) Ecrire un algorithme qui permet de modéliser le problème décrit ci-dessus.
2) On appelle « changement de signe », les déplacements successifs suivants « -1 0 1 » ou « 1 0 -1 ». Modifier l'algorithme de la question 1 pour qu'il permette de compter le nombre de changements de signe.

Exercice 18 (C'est plus, c'est moins)

Ce jeu se joue à deux joueurs. Un des joueurs pense à un nombre entre 1 et 100, l'autre joueur doit deviner ce nombre par propositions successifs. Après une proposition du second joueur, le premier doit indiquer si le nombre proposé est plus grand (ou plus petit) que le nombre pensé.

1) L'ordinateur pense à un nombre (entre 1 et 100) et l'utilisateur doit deviner le nombre pensé. Ecrire un algorithme sur Algobox qui permet cela.
2) L'utilisateur pense à un nombre (entre 1 et 100) et l'ordinateur doit deviner le nombre pensé. Ecrire un algorithme sur Algobox qui permet cela.

Exercice 19 (Le jeu de l'oie)

Le **jeu de l'oie** est un jeu de société de parcours où l'on déplace des pions en fonction des résultats de deux dés. Traditionnellement, le jeu de l'oie comprend 63 cases disposées en spirale enroulée vers l'intérieur. Le but est d'arriver le premier à la dernière case (Wikipedia).

Dans cet exercice, nous allons considérer un jeu de l'oie sans piège (c'est-à-dire qu'aucune case ne permet d'avancer ou reculer dans le jeu).

1) Ecrire un algorithme qui permet de simuler une partie de jeu de l'oie à deux joueurs :
   a) en indiquant le numéro du joueur gagnant ;
   b) en indiquant, à chaque coup de dé, la position sur le plateau du joueur lançant le dé puis à la fin de la partie, le numéro du joueur gagnant.

2) Le jeu de l'oie *précis* est une variante du jeu de l'oie *classique.* Le joueur gagnant est celui qui est **EXACTEMENT** à la case 63. Exemple : pour gagner le jeu dès la case 56, il faut faire un 7 avec les deux dés. Si on fait un 9, on se retrouve à la case 61.

Ecrire un algorithme qui permet de simuler une partie de jeu de l'oie *précis* à deux joueurs en indiquant, à chaque coup de dé, la position sur la plateau du joueur lançant le dé puis à la fin de la partie, le numéro du joueur gagnant.

Exercice 20 (Un jeu de rôle façon texte)
Vous êtes un guerrier qui combat des créatures dans une arène (on ne modélisera pas l'arène en 3D).
Le combattant et la créature ont des caractéristiques (variables)

	Combattant	Créature ennemie
Santé	*sante_c*	*sante_e*
Expérience	*exp_c*	*exp_e*
Magie	*magie_c*	

Voici les valeurs d'initialisation des variables présentés ci-dessus

```
sante_c PREND_LA_VALEUR 100
exp_c PREND_LA_VALEUR 10
magie_c PREND_LA_VALEUR 100
sante_e PREND_LA_VALEUR sante_c - floor(ALGOBOX_ALEA_ENT(0,20)*sante_c/100)
exp_e PREND_LA_VALEUR exp_c - floor(ALGOBOX_ALEA_ENT(0,50)*exp_c/100)
```

Les caractéristiques de la créature seront réinitialisés au début de chaque combat.

Face de dé	Coût magie
1	0 pt, le dé prendra aléatoirement la valeur d'une des trois faces (1, 2 ou 3)
2	
3	
4	5 pts
5	15 pts
6	30 pts

Le combat entre le combattant et la créature ennemie se passe de la manière suivante :
Le combattant peut choisir la face de dé (*de_c)* mais attention les faces de valeur élevée coûtent des points de magie (voir tableau ci-contre).

La créature ennemie aura comme valeur de dé (*de_e*) : *floor((ALGOBOX_ALEA_ENT(1,4))/2)+1*

Celui qui a la plus petite valeur de dé perdra *5*diff_de* (où *diff_de* a pour valeur *abs(de_c-de_e))* de points de santé.
Le combat continue jusqu'à tant que les points de santé de l'un des deux protagonistes soit nul ou négatif.

Si le combattant gagne le combat, il se régénere en points de santé, d'expérience et de magie comme suivant :

```
exp_c PREND_LA_VALEUR (1+ALGOBOX_ALEA_ENT(1,5)/10)*exp_e
sante_c PREND_LA_VALEUR ALGOBOX_ALEA_ENT(80,100)/100*100
magie_c PREND_LA_VALEUR magie_c + ALGOBOX_ALEA_ENT(8,10)/10*(magie_c_init-magie_c)
```

où magie_c_init est la valeur de la variable magie_c au début du combat

Si le combattant perd le combat, il meurt et le score de la partie correspond à *exp_c*.

Ecrire un algorithme qui permet de programmer le RPG dont les règles sont décrites ci-dessus.