

Algorithmique

A. Qu'est-ce que l'informatique

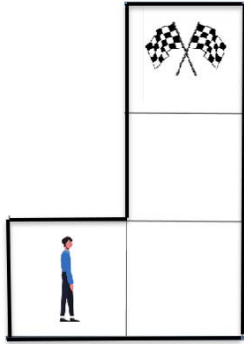
- | |
|---|
| • Le système : est un ensemble d'éléments interagissant entre eux selon certains principes ou règles. |
| • Le réseau : est un ensemble d'équipements reliés entre eux pour échanger des informations. |
| • L'informatique industrielle : est une branche de l'informatique appliquée qui couvre l'ensemble des techniques de conception, d'analyse et de programmation de systèmes informatiques à vocation industrielle. |
| • La robotique : est l'ensemble des techniques permettant la conception et la réalisation de machines automatiques ou de robots. |
| • La sécurité : est l'état d'esprit d'une personne qui se sent tranquille et confiante. Pour l'individu ou un groupe, c'est le sentiment (bien ou mal fondé) d'être à l'abri de tout danger et risque. |
| • L'informatique de gestion (ou de logiciel) : est l'ensemble des connaissances, des technologies, et des outils en rapport avec la gestion de données, c'est-à-dire la collecte, la vérification et l'organisation de grandes quantités d'informations. L'informatique de gestion a de nombreuses applications pratiques dans les entreprises : listes de clients, de fournisseurs, de produits, comptabilité, etc. |

B. Qu'est-ce qu'un algorithme ?

- | |
|---|
| • Un algorithme est une suite d'instructions, permettant de résoudre une classe de problèmes. |
| • Un algorithme en informatique sont des objets historiquement dédiés à la résolution de problèmes arithmétiques, comme la multiplication de deux nombres. Ils ont été formalisés bien plus tard avec l'avènement de la logique mathématique et l'émergence des machines qui permettaient de les mettre en œuvre, à savoir les ordinateurs. |
| • Un pseudo-code également appelé LDA (pour Langage de Description d'Algorithmes) est une façon de décrire un algorithme en langage presque naturel, sans référence à un langage de programmation en particulier. |

Exemple :

Dans cet exemple, un personnage aura pour objectif d'atteindre le drapeau d'arrivée, donc il devra se déplacer de case en case pour arriver à atteindre le drapeau. Pour cela, notre personnage devra respecter certaines règles, à savoir qu'il ne pourra se déplacer que d'une case à la fois ou changer de direction. À chaque étape, le personnage ne pourra pas réaliser qu'une seule action, c'est-à-dire se déplacer d'une case vers l'avant ou changer de direction en tournant à gauche ou à droite. Quand il se tourne, il change de sens que d'un cran d'un côté à gauche ou à droite. Il ne peut pas faire un demi-tour.



Solution :

Il existe plusieurs représentations possibles pour arriver au même résultat :

- **Le 1^{er} algorithme** : La représentation textuelle où l'on a fait simplement des phrases pour indiquer ce que notre personnage fait, donc le personnage avance d'une case puis tourne sur sa gauche et avance d'une case.
- **Le 2^e algorithme** : une liste d'action permet également d'arriver au même résultat :
 - **Étape 1** : le personnage avance d'une case.
 - **Étape 2** : le personnage tourne sur sa gauche.
 - **Étape 3** : le personnage avance d'une case
 - **Étape 4** : le personnage avance d'une case
- **Le 3^e algorithme** : des symboles peuvent être des flèches directionnelles pour indiquer ce que devait faire le personnage

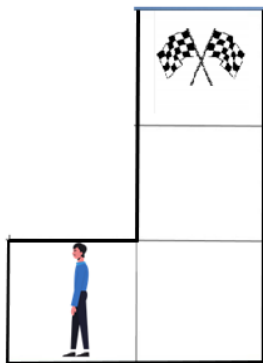


En ce fait, un algorithme est un ensemble d'instructions ou des étapes qui permettent d'arriver à un résultat.

Dans les sections suivantes, on va se concentrer à vous apprendre comment décrire un algorithme avec une syntaxe spécifique. Le pseudo-code qui est le langage utilisé par les informaticiens. Donc on va travailler sur l'aspect logique et l'objectif c'est d'arriver à avoir l'état d'esprit de l'informaticien.

a. Exercice 1

Vous devez déplacer le personnage pour y arriver au drapeau.



Vous ne disposez que de trois instructions suivantes :

- Avancer
- Tourner gauche

- Tourner droite

Solution proposée :

Algorithme exercice 1

VARIABLE

DEBUT

Tourner gauche

Tourner gauche

Avancer

Tourner droit

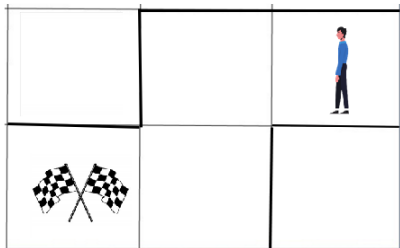
Avancer

Avancer

FIN

b. Exercice 2

Cet exercice a le même principe que le précédent :



Vous ne disposez que des instructions suivantes :

- Avancer
- Tourner gauche
- Tourner droite

Solution :

Algorithme Ex2

VARIABLE

DEBUT

Avancer

Tourne droite

Avancer

Tourne gauche

Avancer

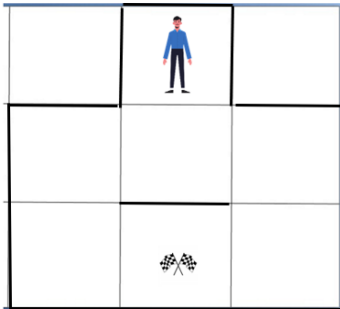
FIN

c. Exercice 3

Vous devez simplement vérifier si l'algorithme décrit permet au personnage d'atteindre le drapeau d'arrivée ou non. Prouve s'il y a une erreur et le cas échéant le corrigé.

1. Avancer
2. Tourner gauche

3. Avancer
4. Tourner droite
5. Tourner droite
6. Avancer



Solution :

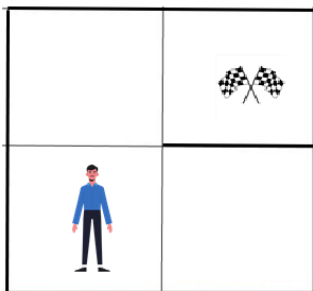
Il y a une erreur et le cas échéant était ;

Entre 4 et 5 Cela aurait dû l'être avancer

d. Exercice 4

Vérifiez si l'algorithme suivant est correct, sinon corrigez-le :

1. Tourner gauche
2. Avancer
3. Tourner gauche
4. Tourner gauche
5. Avancer
6. Tourner droite
7. Avancer
8. Tourner droite
9. Avancer



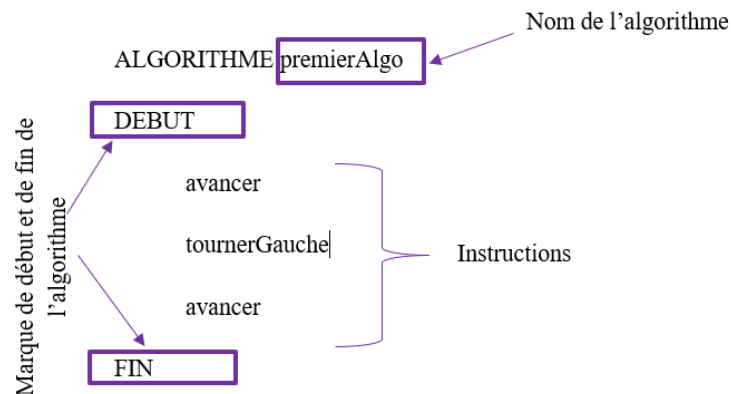
Solution

l'algorithme suivant est correct

C. Le pseudo-code

Pour réaliser nos algorithmes, nous utiliserons une syntaxe qui permettra d'être compris par tous les informaticiens. Cette syntaxe est parfois légèrement différente suivant les formateurs ou la langue, mais l'important à retenir la logique. Ces algorithmes ne peuvent pas, dans tous les cas, être compris par l'ordinateur. Il faudra ensuite les retranscrire en programme grâce à un langage spécifique.

Exemple :



Voici notre premier algorithme si l'on reprend le premier exercice que nous avons fait. Les éléments de syntaxe suivants permettent de réaliser du pseudo-code au niveau des règles que nous allons suivre. Nous allons mettre tous les mots clés en majuscules. On utilisera le 'camelCase' pour les noms de nos informations (par exemple le nom de l'algorithme ou encore les instructions, les variables, etc.). Le 'camelCase' est de mettre la première lettre de chaque mot en majuscules sauf pour le premier mot, donc nous avons l'exemple ici sur le nom de l'algorithme. Dans l'exemple, on a mis 'premierAlgo' la première lettre en minuscule et la première lettre en majuscule. La deuxième syntaxe est de réaliser une tabulation (comme dans l'exemple ci-dessus).

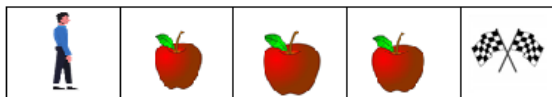
Dans l'exemple, nous avons trois mots clés :

- **Algorithme** permet de commencer notre algorithme. Nous plaçons ce mot au début pour signaler le début de l'algorithme. On inscrit ensuite le nom de l'algorithme que l'on veut (dans notre exemple, on appelait 'premierAlgo').
- **Pour signaler le début et la fin** d'un algorithme, on utilisera les mots clés 'DEBUT' et 'FIN'. Entre ces deux mots clés, vous devrez placer toutes les instructions de notre algorithme.
- **Les instructions** sont ce qu'on retrouve à l'intérieur et qui permet de réaliser les étapes. Chacune de ces instructions et là aussi écrite en 'camelCase'.

D. Les variables

Une variable est un symbole qui contient une valeur, qui peut être modifiée au cours du temps. Autrement dit, c'est un conteneur qui dispose d'une valeur modifiable. Ce symbole ou ce conteneur est représenté par un nom unique, donc comme en mathématiques on vous utilisait une variable dans un calcul, cette valeur de la variable pouvait être différente et donner un résultat différent.

Exemple :



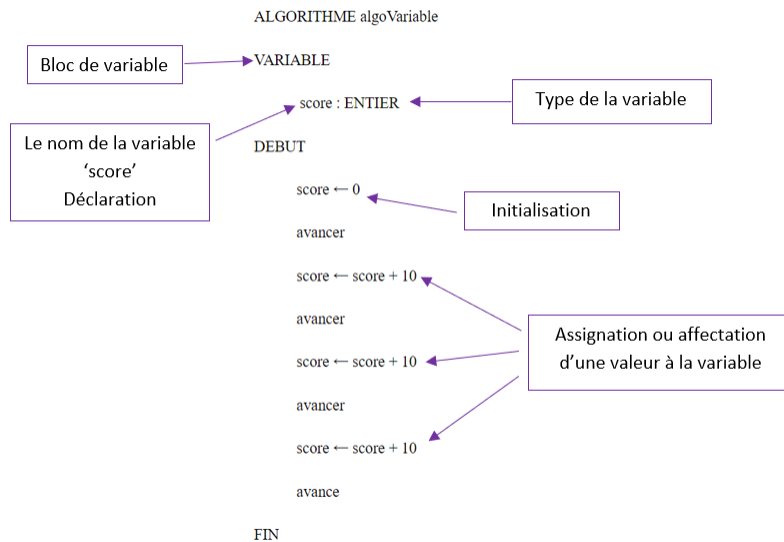
Si par exemple, le personnage touche une pomme, il gagne 10 points. Nous devons conserver cette information c'est-à-dire le score à chaque étape, pour cela nous devons utiliser une variable que nous pourrions appeler par exemple 'score'. Cette variable contiendra donc la valeur de score de notre personnage dans le niveau.

Soit la variable score du personnage, voici son évolution à chaque étape :

1. Étape 0 : score = 0 puisqu'il n'a pas encore mangé de pomme.
2. Étape 1 : score = 10, le personnage se déplace d'un cran et il a mangé une pomme du même coup, son score a pris plus de 10 points

3. Étape 2 : score = 20, le personnage a encore avancé, il a à nouveau mangé une pomme du même coup, il a pris encore plus de 10 points, et donc la valeur de score est égale à 20.
4. Étape 3 : score = 30, le personnage est encore déplacé et a encore mangé une pomme, donc le score a nouveau augmenté de 10 et donc la nouvelle valeur est 30.
5. Étape 4 : score = 30, le personnage se déplaçait et a atteint le drapeau d'arriver du coup le score n'a pas modifié.

On va faire le même truc en pseudo-code. La première chose à faire ça va être d'utiliser les mots clés que nous avons vues précédemment.



Explication :

Le mot ALGORITHME donne le nom de notre algorithme.

'DEBUT' et 'FIN' définissent les blocs d'instructions où nous mettons toutes les actions qui vont se passer pendant l'algorithme.

Le mot VARIABLE définit nos conteneurs qui contiendront les informations importantes de notre algorithme.

Le mot entier indique les nombres signés permettant de faire notamment des calculs.

score : ENTIER, cette ligne permet de déclarer la variable score

score ← 0, cette ligne permet d'initier la variable score. Initialiser une variable est de donner une première valeur à notre variable

score ← score + 10, la variable score reçoit son ancienne valeur plus 10 .

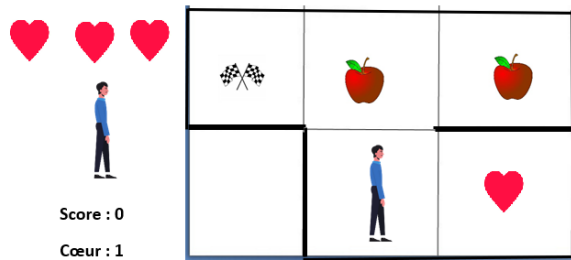
L'assignation correspond à un peu à l'initialisation dans un certain sens. L'assignation est le fait de donner une valeur à une variable. Quand on fait une initialisation, on réalise une assignation, ensuite on va mettre de nouvelles informations dans nos variables. Le symbole de l'assignation est le flèche '←'

Remarque :

- Déclaration indique à l'algorithme que la variable existe
- Initialisation : donner une première valeur à une variable.
- Assignation : donner une valeur à une variable

a. Exercice 5

Vous devez écrire l'algorithme de l'exercice en pseudo-code. Pour cela, vous aurez besoin de deux variables et vous pourrez choisir leurs noms.



Solution

Algorithme exercice 5

Variable

Score :0;
cœur : 1;

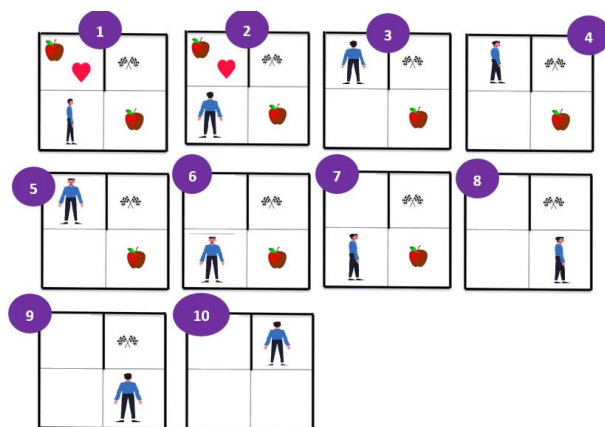
debute

Score<- 0
Cœur <- 1
Avancer
Cœur <- cœur +1
Tourner gauche
Tourner gauche
Avancer
Tourner droit
Avancer
Score <- score+10
Tourner a doit
Avancer
Score <- score +10
Tourner gauche
Tourner gauche
Avancer
Avancer

Fin

b. Exercice 6

Vous devez trouver les erreurs dans l'algorithme (pseudo-code) ci-dessous. Nous vous invitons à chercher l'erreur et de la corriger.



ALGORITHME algoVariable2

VARIABLE

score : ENTIER
cœur : ENTIER

DEBUT

score \leftarrow 0
cœur \leftarrow 1
tournerGauche
avancer
score \leftarrow score + 10
cœur \leftarrow cœur + 1
tournerGauche
tournerGauche
avancer
tournerDroite

avancer
tournerGauche
avancer

FIN

Solution :

Incorrect

Corriger (

Algorithme algovariable2

Vardiable

Cœur : entier

Score : entier

Debut

Score \leftarrow 0

Cœur \leftarrow 1

Tourner gauche

Avancer

Score \leftarrow score + 10

Cœur \leftarrow cœur + 1

Tourner droit

Tourner droit

Avancer

Tourner gauche

Avancer

Score \leftarrow score + 10

Tourner droit

Avancer

Fin

)

E. Les types

Nous avons déjà utilisé le type entier c'est-à-dire les nombres. Un type est une information qui est donnée à une variable comme nous l'avons fait dans les sections précédentes avec la variable score. Imaginons que nous avons deux personnages 'karim' et 'Nabila', ces deux personnages disposent tous les deux d'un nom, d'un âge et d'un sexe. Ces trois types d'informations peuvent

être stockés dans des variables, mais comme vous pouvez le remarquer ce ne sont pas tous des nombres, il va donc falloir définir un nouveau type pour stocker ces informations.



Karim
19
Garçon



Nabila
23
Fille

Nous pouvons définir trois variables :

- Nom avec la valeur 'Nabila' pour Nabila et la valeur 'Karim' pour Karim
- Âge avec la valeur 23 pour Nabila et 19 pour Karim.
- Sexe avec la valeur 'fille' pour Nabila et 'garçon' pour Karim

Nous avons utilisé :

- Une chaîne de caractères pour les variables 'nom' et 'sexe'

Un entier pour la variable 'age'

En réalité, utiliser une chaîne de caractère pour conserver l'information du sexe n'est pas optimal. En effet, il existe deux possibilités fille ou garçon. Ces deux valeurs pourraient être représentés également par des chiffres par exemple 0 et 1 c'est-à-dire du binaire ou représentée par un autre type qui s'appelle le type **booléen**, et peut avoir deux valeurs la valeur vraie et la valeur faux. Dans cet exemple, on pourrait considérer que garçon est la valeur 'Vrai' pour le sexe et que 'Faux' est la valeur fille pour le sexe.

Il existe donc trois types communs utilisés :

- Les entiers, qui correspondent à des nombres signés.
- Les chaînes de caractères qui correspondent à des mots ou des textes.
- Les booléens qui ne peuvent prendre que la valeur 'Vrai' ou 'Faux'.

Exemple :

Variables :

Nom
age
sexe



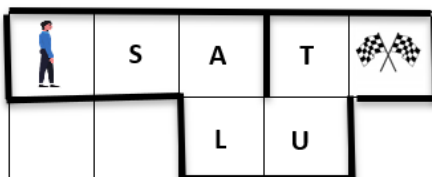
Karim
19
Garçon



Nabila
23
Fille

a. Exercice 7

Réalisez l'algorithme permettant à Karim de récupérer toutes les lettres 'SALUT' et de finir le niveau.



Solution :

Algorithme EX7

VARIABLE

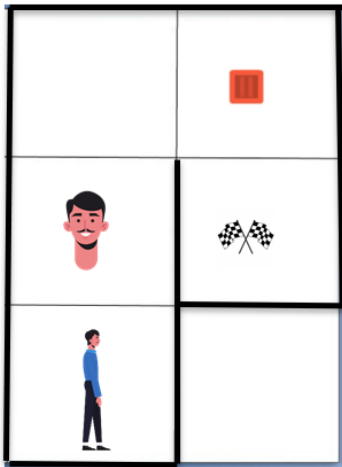
DEBUT

Avancer
Avancer
Tourne droite
Avancer
Tourne gauche
Avancer
Tourne droite
Avancer
Tourne gauche
Avancer
Ecrire "SALUT" ;

FIN

b. Exercice 8

Réalisez l'algorithme permettant à Karim d'atteindre l'arrivée. Lorsque Karim rencontre le grand visage, il devient plus gros, et il ne peut avoir que deux taille une taille grande et une autre petite. Si Karim touche le truc rouge, il rétrécit.



Solution :

Algorithme exercice 8

Variable

Taille : Boolean ;

Début

Taille : faux ;

Tourner gauche

Avancer

Tourner droite

Avancer

Taille : faux ;

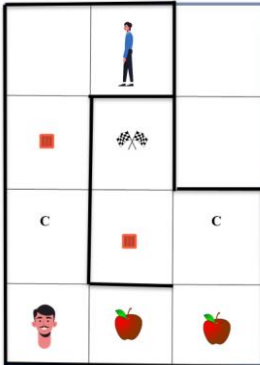
Tourner a doit

Avancer

Fin

c. Exercice 9

Réalisez l'algorithme afin que Karim atteigne le drapeau.



Solution :

Algorithme EX9

VARIABLE

Taille : Boolean ;

Score : Antier ;

DEBUT

Taille <- faux

Score <- 0

Avancer

Tourner gauche

Avancer

Taille <- faux

Avancer

Avancer

Taille <- Vrais

Tourner gauche

Avancer

Score = score + 10 ;

Avancer

Score = score + 10 ;

Tourne gauche

Avancer

Tourne gauche

Avancer

Taille <- faux

Tourne droite

Avancer

FIN

F. Les tests

Le test est un sujet simple à la base et que nous utilisons dans la vie de tous les jours. Nous pouvons prendre un exemple, si une voiture a de licence alors elle peut avancer sinon il s'arrête. On peut aussi imaginer si la voiture n'a plus de licence alors il va falloir aller à la station de service pour faire le plein. Tous ces exemples sont des tests que vous réalisez tous les jours et si vous avez remarqué à chacune des phrases on a commencé par le mot clé 'si'. Donc c'est un mot que vous permettez de réaliser des tests. Donc si l'on a une condition alors on fait une action spécifique.

En informatique, ça sera exactement les mêmes choses, et donc en algorithmique certaines actions se produisent en fonction de l'état d'un élément. Pour tester l'état d'un élément, nous allons utiliser le mot clé 'si'.

On peut traduire cela avec la syntaxe de l'algorithmique Si. L'instruction 'Si' en pseudo-code donne :

```
si condition
alors
    bloc d'instructions
sinon
    bloc d'instructions
fin
```

- Une **condition** est une expression logique donnant le résultat 'Vrai' ou 'Faux'. C'est une expression booléenne.
- L'expression **Alors bloc d'instruction** est utilisée dans le cas où le résultat du test sur la condition serait 'Vrai' alors on réalise les actions dans le bloc 'alors'.
- L'expression **sinon bloc d'instruction** est utilisée dans le cas où le résultat du test sur la condition serait 'Faux' alors on réalise les actions dans le bloc 'Sinon'.

Exemple :

Si Karim ramasse les pommes alors il va afficher le message "Youpi"



Exemple 2 :

Que se passe-t-il sur l'algorithme suivant, avec le test ? Vous avez l'algorithme à gauche et la représentation graphique à droite.

Score : 0



```
si score > 15
alors
    AFFICHER "j'ai mangé plus de 1 pomme"
sinon
    AFFICHER "j'ai mangé ma première pomme"
fin
```

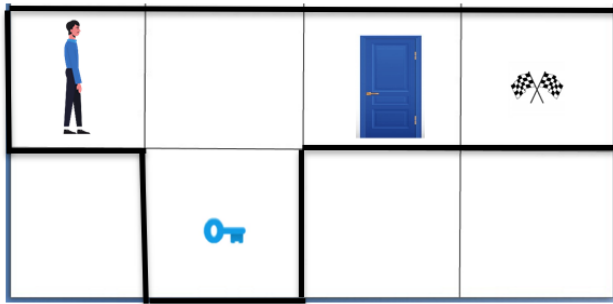
La représentation n'est pas complète, l'objectif est de poser la question : qu'est-ce qu'il va passer au niveau de la représentation graphique et à quel moment un message spécifique va s'afficher ?

Solution :

to do ...

a. Exercice 10

Écrivez l'algorithme permettant d'atteindre le drapeau. Pour passer la porte bleue, Karim a besoin de la clef.



Solution :

Algorithme EX10

Variable

Key : Boolean ;

DEBUT

Key <- faux ;

Avancer

Tourne droite

Avancer

Key <- vrais ;

Tourne doit

Tourne doit

Avancer

Tourne doit

Avancer

If (key == vrais) {

Avancer

} else{

Écrire "vous avez besoin de la clé pour ouvrir la porte !" ;

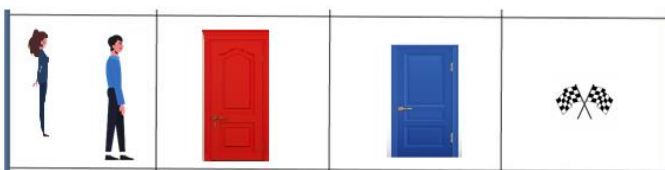
}

FIN

b. Exercice 11

Soit les hypothèses et l'algorithme suivant :

- Seulement les personnages ayant plus de 20 ans (inclus) peuvent passer la porte rouge.
- Seulement les personnages ayant plus de 28 ans (inclus) peuvent passer la porte bleue.



Karim
30
Vrai



Nabila
25
Faux

Solution

Algorithme exercice 11

Variables

Age : Antier

Début

Ecrire "Age de personne " ;

Lire (Age) ;

Age <- Age de personnage

Si (Age > 20)

Avancer

Sinon

Break ;

Si (Age > 28)

Avancer

Sinon

Break ;

Avancer

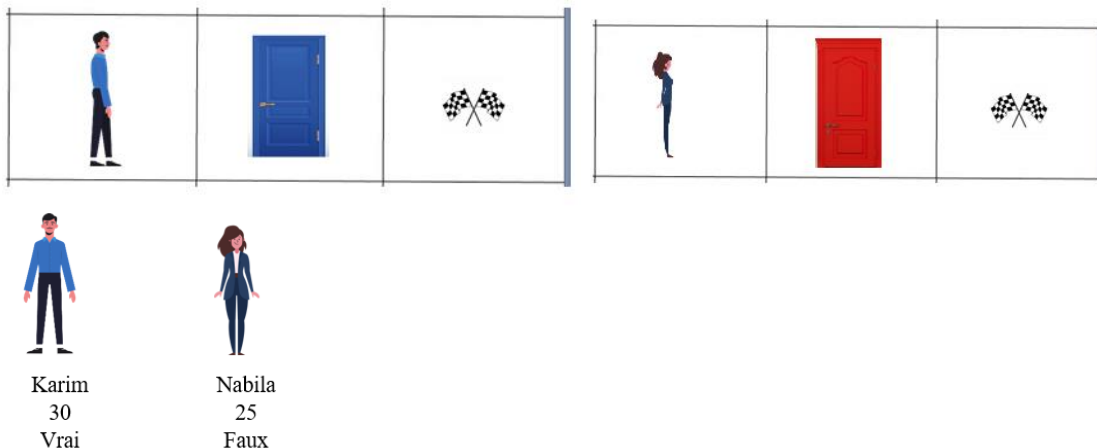
Fin

G. Les opérations logiques ET et OU

Il est possible d'imbriquer des blocs Si et alors sinon et l'on peut également rajouter plusieurs conditions plusieurs expressions booléennes dans un même test. Pour cela, on va avoir deux cas possibles, le cas où l'on voudra deux conditions soit vraies et le cas où l'on voudra que les deux conditions soient possiblement vraies c'est-à-dire au moins l'une des deux qui soit vraies.

Exemple 1 :

Pour passer la porte bleue, il faut avoir plus de 20 ans ET être un garçon.



Karim remplit la condition, car il a 30 ans et que c'est un garçon. Par contre, Nabila a plus de 20 ans, mais c'est une fille. Et vu qu'elle n'a pas un garçon, elle ne pourra pas passer.

Ici on a utilisé 'ET' qui nous a permis de combiner les deux conditions et doivent être rempli pour valider le test.

Voici l'algorithme en pseudo-code :

```

ALGORITHME algoKarim
VARIABLE
    age : ENTIER
    sexe : BOOLEEN
DEBUT
    age ← ageDuPersonnage
    sexe ← sexeDuPersonnage
    Si age > 20 ET sexe = Vrai
    alors
        avancer
    sinon
        AFFICHER "je ne peux pas passer"
        arretDeAlgo
    FINSI

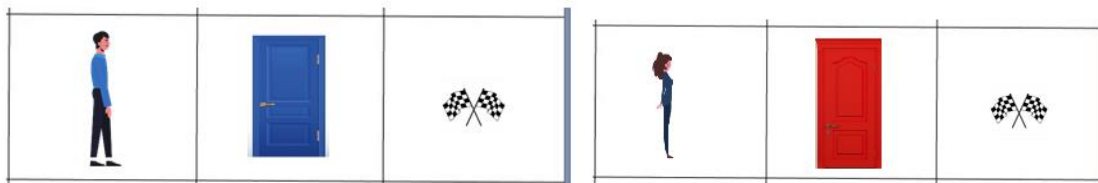
```

FIN

ET est un opérateur logique permettant de combiner les tests. Pour que le résultat du test soit vrai, il faut que les deux conditions soient vraies.

Exemple 2 :

Pour passer la porte bleue, il faut avoir plus de 28 ans OU être une fille.



Karim peut passer, car il remplit la première condition même si qu'il est garçon. Pour Nabila peut aussi passer, car elle remplit la deuxième condition.


```

ALGORITHME algoKarim
VARIABLE
    age : ENTIER
    sexe : BOOLEEN
DEBUT
    age ← ageDuPersonnage
    sexe ← sexeDuPersonnage
    SI age > 28 OU sexe = Faux
    alors
        avancer
    SINON
        AFFICHER "je peux pas passer"
        arretDeAlgo
    FINSI
FIN

```

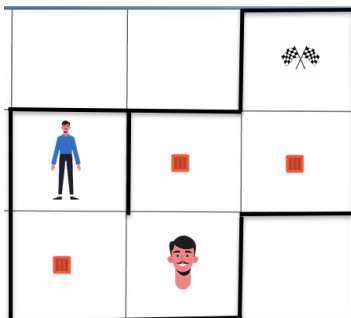
OU est un opérateur logique permettant de combiner des tests. Pour que le résultat du test soit vrai, il faut que l'une des deux conditions soit "Vrai".

a. Exercice 12

Réalisez l'algorithme permettant à Karim d'atteindre l'arrivée.

Lorsque Karim rencontre Karim de grande taille, il devient plus gros, il affiche 'Youpi'.

- Si Karim touche une boîte rouge, il rétrécit et il affiche le message 'Aie'.
 - Si Karim est déjà petit (donc qu'il a déjà touché une boîte) alors il crie 'Ouille' à la place.



Solution :

Algorithme EX12

Variable

Taille : Boolean ;

Br : Boolean ;

DEBUT

Taille <- faux ;

Br <- faux ;

Avancer

Br <- Vrais ;

```

Ecrire "Aié XC" ;
Tourne gauche ;
Avancer
Taille <- vrais ;
Br <- faux ;
Ecrire "Youpiiii XD" ;
Tourne gauche ;
Avancer
Br <- Vrais ;
Ecrire "Aié XC" ;
Tourne droite ;
Avancer
If (Br==Vrais) {
    Ecrire "Ouille XC !!!!!!!" ;
}
Sinon
    Ecrire "Aié XC" ;
Tourne gauche ;
Avancer

```

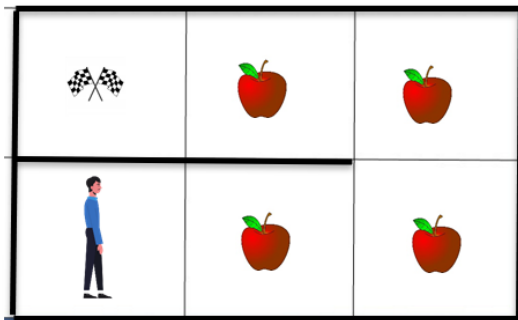
FIN

b. Exercice 13

Réalisez l'algorithme permettant à Karim d'atteindre l'arrivée.

- Si Karim touche une pomme, il affiche son score avec le message suivant :
 - Mon score est x points.

De plus, si le score de Karim est supérieur à 25 alors Karim grandit.



Solution

Algorithme exercice 13

Variable

Score : Antier

Taille : Boolean

Début

Taille <- faux

Score <- 0

Avancer

Score : 10

```

Ecrire ("Mon score est", score, "points")
Avancer
Score : score + 10
Ecrire ("Mon score est", score, "points")
Tourner gauche
Avancer
Score : score + 10
Ecrire ("Mon score est", score, "points")
Si (score > 25 alors
  Taille <- vrais
  Avancer
  Score : score + 10
  Ecrire ("Mon score est", score, "points")
  Avancer

```

Fin

H. L'instruction Selon

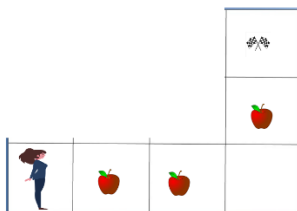
L'instruction 'Selon' permet de réaliser certaines actions en fonction d'une variable (comme le permet l'instruction 'Si').

L'instruction 'Selon' permet de tester plusieurs valeurs de manière plus efficace qu'un Si, et de réduire le nombre de Si imbriqués.

L'instruction 'Selon' ne permet pas de tout tester. Il n'est pas très adapté pour tester des plages de valeur (signe < ou >), exemple entre 5 et 15.

Exemple :

Imaginons que Nabila dispose d'une 'barre' de 3 carrés, et que celle-ci est vide au démarrage de l'algorithme.



Le pseudo-code :

```

SELON faim
  cas 0 : AFFICHER "j'ai faim"
  cas 1 : AFFICHER "j'ai encore faim"
  cas 2 : AFFICHER "je n'ai pas bien mangé"
  cas 3 : AFFICHER "j'ai bien mangé"
  autrement : Afficher "ce cas n'est pas inclus"
FINSELON

```

- **faim** est une variable.
- **Cas 0** est la valeur testée.
- **autrement** : si la valeur testée n'est pas prise en compte dans les 'cas' alors c'est la partie 'autrement' qui sera traitée.

Question : complétez le pseudo-code ci-dessus.

a. Exercice 14

Réalisez un algorithme permettant d'afficher le mois de l'année en lettres à partir d'un chiffre (compris entre 1 et 12).

1 → Janvier

2 → Février

...

12 → Décembre

Utilisez l'instruction 'Saisir' pour démarrer une saisie utilisateur, qui va permettre d'assigner une valeur à la variable 'mois'.

Exemple :

SAISIR x permet de saisir une valeur pour la variable x (si l'utilisateur saisit 11 alors la variable x = 11).

Solution

to do ...

b. Exercice 15

Réalisez un algorithme permettant d'afficher le jour de la semaine en lettres à partir d'un chiffre saisi au clavier.

1 → Lundi

2 → Mardi

...

7 → Dimanche

Utilisez l'instruction 'Saisir' pour démarrer une saisie utilisateur, qui va permettre d'assigner une valeur à la variable 'jour'.

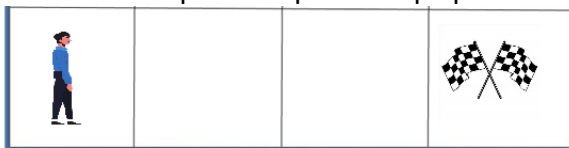
To do

I. Les boucles

Lorsque l'on réalise plusieurs fois la même action à la suite, il est regrettable de devoir réécrire x fois les mêmes lignes d'instructions. Les boucles vont vous permettre d'éviter les répétitions, mais il faudra que l'algorithme respecte certaines conditions. De plus, dupliquer du code est très limitant et impose d'avoir des algorithmes complexes.

Exemple :

Voici un exemple de répétition qui permet à Karim d'atteindre le drapeau



Algorithme sans boucle : avancer 3 fois Karim

ALGORITHME algoEx

DEBUT

 avancer

 avancer

 avancer

FIN

Pour améliorer l'algorithme précédent, il faudrait pouvoir indiquer que Karim réalise l'action 'avancer' jusqu'au drapeau. Il existe deux façons de le faire :

Réaliser une boucle à partir au moment où l'on sait le nombre de fois pour réaliser une action donnée (exemple : Karim doit avancer 3 fois). À partir de moment où l'on sait cette condition-là qu'il est fixe, on pourra utiliser la boucle '**POUR**' qui va permettre d'effectuer la même action x fois.

Si jamais on ne connaîtra pas le nombre de fois où Karim doit avancer, dans ce cas-là on va dire que Karim doit avancer tant qu'il n'a pas atteint l'arrivée. Donc on utilisera la boucle '**TANT QUE**' (ou '**FAIRE**' / '**TANT QUE**') pour réaliser une action tant qu'une condition n'est pas atteinte.

1. La boucle **POUR**

La boucle **POUR** permet de répéter des actions en fonction de nombre de fois prédéfinies à l'avance.

Syntaxe :

```
POUR compteur ALLANT DE debut A fin PAR PAS DE x  
FAIRE  
    <Instructions>  
FINPOUR
```

DEBUT

avancer

avancer

avancer

FIN



ALGORITHME algoEx

DEBUT

POUR compteur ALLANT DE 1 A 3 PAR PAS DE 1 FAIRE

avancer

FINPOUR

FIN

Le compteur commence à la valeur 1 et s'arrête à la valeur 3. Il augmente de 1 en 1.

Remarque :

L'incrémentation est l'opération qui consiste à augmenter une variable d'un nombre donné (+1, +2, +3 ...).

a) Exercice 16

- Aidez Nabila à atteindre le drapeau.

Algo EX16

Variable

Début

Avancer

Tourner droite

POUR compteur ALLANT DE 1 A 2 PAR PAS DE 1 FAIRE

Avancer

FINPOUR

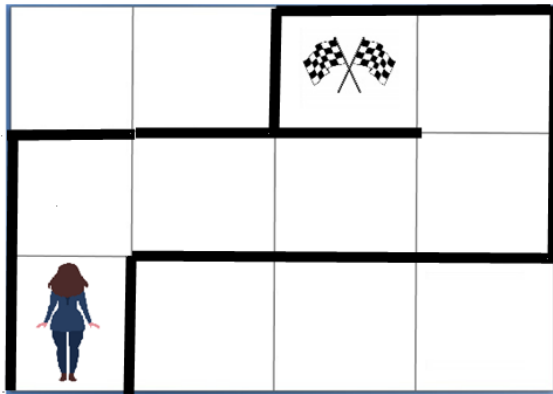
Tourne gauche

Avancer

Tourne gauche

Avancer

FIN



b) Exercice 17

Aidez Nabila à atteindre le drapeau

Algorithme EX17

Variable

Début

Avancer

Tourne droite

POUR compteur ALLANT DE 1A3 PAR PAS DE 1 FAIRE

Avancer

FINPOUR

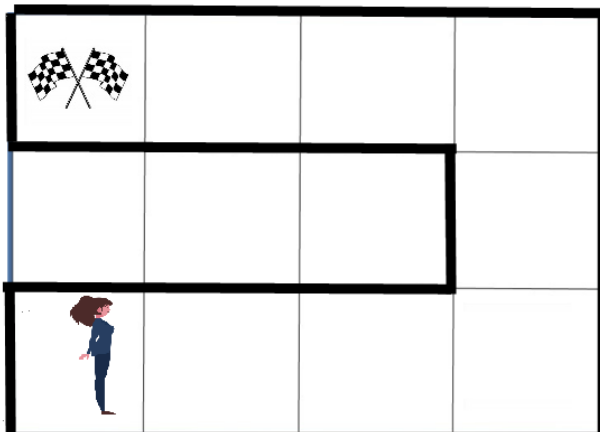
Tourne gauche

Avancer

Tourne gauche

Avancer

Fin



c) Exercice 19

Écrivez un algorithme qui permet d'afficher la factorielle (symbole mathématique : !) d'un nombre saisi au clavier.

Le résultat attendu doit être le suivant :

Le chiffre saisi est 5. $5! = 1*2*3*4*5$

Algorithme : factorial

Variable

x,c,s : entiers

Debut

Ecrire (" donner un number: ")

Lire (x)

S <- 1

Pour c=x A c=1 pas -1

s <- s*c

Fin Pour

Ecrire ("le valeur factoriel de ce nombre est:",s)

Fin

2. La boucle TANT QUE

La boucle TANT QUE permet de faire la même chose que la boucle POUR, mais n'aura pas la condition d'arrêt qui sera fixer, on ne sera pas effectivement combien de fois on devra boucler.

Syntaxe :

```
TANT QUE condition FAIRE
    <Instructions>
FINTANTQUE
```



ALGORITHME algoEx

DEBUT

avancer

avancer

avancer

FIN



ALGORITHME algoEx

DEBUT

TANT QUE naPasAtteintDrapeau FAIRE

avancer

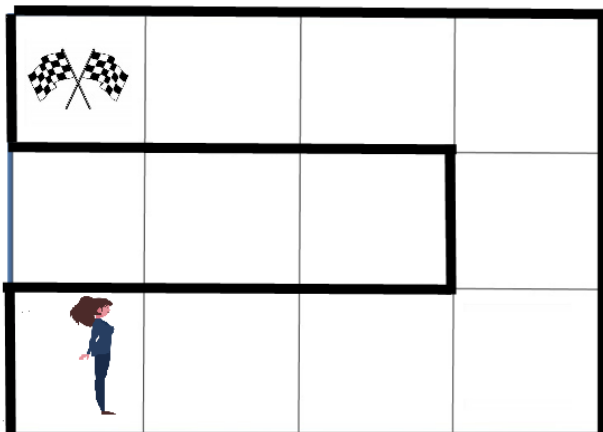
FINTANTQUE

FIN

Karim s'arrêtera quand il aura atteint la condition d'arrêt.

Exercice

Reprenons l'algorithme suivant, et réécrivons-le avec la boucle TANT QUE :



Algorithme EX

Variable

DEBUT

TANT QUE nePasTrouvermure FAIRE

Avancer

Fintantque

Tourne gauche

Avancer

Avancer

Tourne gauche

TANT QUE nePasattindreDraoeux FAIRE

Avancer

Fintantque

FIN

a) Exercice 20

Écrivez un algorithme qui permet de trouver la première factorielle qui dépasse le chiffre 1200. Dans cet exercice, on ne sait pas à l'avance combien de fois il faudra boucler d'où l'obligation d'utiliser un TANT QUE pour trouver le résultat.

Solution

to do ...

b) Exercice 21

Écrivez un algorithme qui permet à un utilisateur de saisir un chiffre. Si le chiffre saisi est inférieur à 10, l'algorithme redemande à l'utilisateur de saisir un nouveau chiffre.

Solution

Algorithme EX21

Variable

A : Antier ;

Debut

Ecrire "Donne le nombre SVP! :";

Lire (A);


```

si(A<10){
    Ecrire"Number must be infirieur a 10 !!!"
    Ecrire "Donne le nombre SVP! :";
}

```

Fin

3. La boucle FAIRE TANT QUE

Les boucles **FAIRE TANT QUE** et **TANT QUE** sont identiques, à une seule différence près : le **TEST** de la condition est réalisé en premier pour le **TANT QUE** et après les instructions pour le **FAIRE TANT QUE**.

Symbole

```

FAIRE
    instructions
TANT QUE condition

```

Exemple :



```

ALGORITHME algoTantQue
DEBUT
    FAIRE
        avancer
    TANT QUE pasAtteintDrapeau
FIN

```

Remarque

On utilise la boucle 'FAIRE TANT QUE' seulement si l'on est certain que la première itération ne causera pas une erreur.

a) Exercice 22

Refaire le même exercice 21, mais cette fois utilisez 'FAIRE TANT QUE'.

Solution

Algorithme EX22

Variable

A : Antier ;

Début

```

Ecrire "Donne le nombre SVP ! :";
Lire (A);
Tank que(A<10)Faire{
    Ecrire"Number must be infirieur a 10 !!!"
    Ecrire "Donne le nombre SVP! :";
}
Fin Tank que

```

Fin

J. Les fonctions

1. Qu'est-ce qu'une fonction ?

Une fonction est un outil permettant de créer un bloc d'instructions réutilisable à plusieurs endroits. En réalité, nous l'avons utilisé déjà lorsque nous fusons écrire avancer, tourner gauche ... même on n'a pas dit que ce sont des fonctions, en réalité sont été.

Les fonctions permettent de regrouper des morceaux d'algorithme. Ce sont des mini-algorithmes ayant pour objectif de réaliser des actions précises. Elles peuvent ensuite être utilisées à plusieurs reprises, et à différents moments dans l'algorithme principal.

Exemple :

Créez une fonction qui va réaliser les actions suivantes, si Nabila touche une pomme, elle va gagner 10 points, va gagner un point de vie et va afficher 'Youpi'.



Voici l'algorithme de notre fonction "manger" :

- Gagner un point de vie.
- Gagner +10 points.
- Afficher "Youpi".

```
FONCTION manger()  
VARIABLE  
    pointVie : ENTIER  
    score : ENTIER  
DEBUT  
    pointVie ← 0  
    score ← 0  
    pointVie ← pointVie + 1  
    score ← score + 10  
    AFFICHER "Youpi"  
FIN
```

L'algorithme principal "monAlgorithme" :

- Avancer jusqu'au drapeau.
- S'elle rencontrerait une pomme alors "manger".

```

ALGORITHME monAlgorithme
DEBUT
    avancer
    manger()
    avancer
    avancer
    manger()
    avancer
FIN

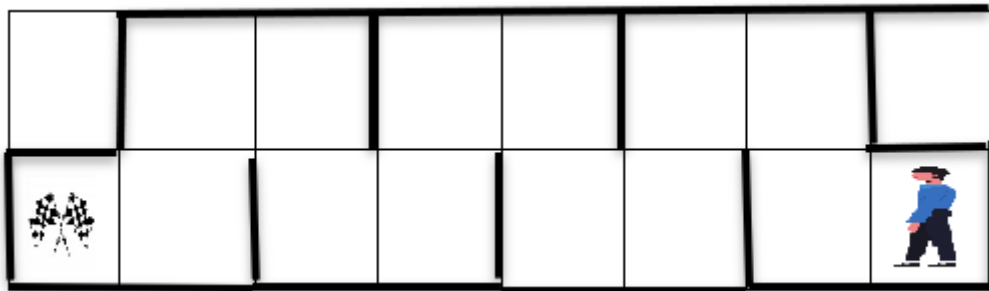
```

Question :

Optimisez votre algorithme ci-dessus.

a) Exercice 23

Faire traverser le niveau à Karim. Identifiez les séquences et réalisez les fonctions. **to do ...**



2. Les fonctions avec un paramètre

Une fonction est un algorithme comme une autre qui permet de réaliser une tâche spécifique, celle-ci peut s'attendre à recevoir une information afin qu'elle réalise une action spécifique.

Exemple

Nous voulons réaliser une fonction qui permet d'ajouter 7 à un nombre qui lui a été donné comme information en entrée. Elle affiche ensuite le résultat.

```

FONCTION ajouterNombre(nombre)
VARIABLE
    resultat : ENTIER
DEBUT
    resultat ← 7 + nombre
    AFFICHER "resultat = ", resultat
FIN

```

À chaque fois que nous appellerons la fonction "ajouterNombre" dans un programme, nous devons lui donner un nombre de types entier et elle affichera le résultat de l'addition.

```

ALGORITHME monAlgorithme
VARIABLE
    nombre1 : ENTIER
    nombre2 : ENTIER
DEBUT
    nombre1 ← 3
    ajouterNombre(nombre1)
    SAISIR nombre2
    ajouterNombre(nombre2)
FIN

```

Remarque

L'avantage de l'utilisation d'une fonction est de rendre un morceau de code réutilisable par plusieurs algorithmes ou à plusieurs endroits dans un algorithme.

a) Exercice 24

Réalisez une fonction qui va afficher le message "Bonjour" suivi d'un nom (par exemple, Bonjour Web4Jobs). Réalisez un exemple d'algorithme appelant.

Solution

Algorithme EX24

Variable

A : réel

Début

Ecrire "Entre votre nom SVP! "

Lire (A)

Ecrire " Bonjour "+A

Fin

b) Exercice 25

Réalisez une fonction qui va afficher si un nombre passé en paramètre est pair ou impair. Pour tester si un nombre est pair, vous pouvez utiliser le modulo : mod (exemple : $4 \bmod 2 = 0$).

Solution

to do ...

3. Les fonctions avec plusieurs paramètres

Il est possible de transmettre plusieurs informations directement à une fonction, et donc avoir plusieurs paramètres. Il faudra par conséquent faire attention à l'ordre des informations passées.

Exemple

Réalisez une fonction permettant d'afficher 'Bonjour' suivit par le nom et le prénom de quelqu'un.

Solution

to do ...

a) Exercice 26

Réalisez une fonction permettant de calculer la moyenne de 5 nombres.

Solution

to do ...

b) Exercice 27

Réalisez une fonction prenant deux informations : l'âge (un entier) et le sexe (un booléen).

- Si l'âge est inférieur à 17 alors la fonction affichera : "Trop jeune".
- Si l'âge est supérieur à 30 alors la fonction affichera : "Trop vieux".
- Si le sexe est 'Vrai' alors la fonction affichera : "Femme".
- Si le sexe est 'Faux' alors la fonction affichera : "Homme".

Solution

to do ...

4. Retour de fonction

Une fonction pourra réaliser un traitement spécifique et renvoyant ensuite une information à l'algorithme appelant, ça vous permet également de vous indiquer qu'une fonction qui ne retourne aucune information est appelée "Procédure". Alors une fonction qui retourne une information est en générale appelée "Fonction".

Exemple :

Réalisez une fonction permettant de retourner la somme de trois valeurs.

Solution

to do ...

a) Exercice 28

Réalisez une fonction permettant d'indiquer si un nombre est divisé par 3. L'algorithme principal se chargera d'afficher le message correspondant.

Solution

to do ...

b) Exercice 29

Réalisez une fonction qui permet de retourner "Bonjour" suivi du nom et du prénom de quelqu'un.

Solution

to do ...

c) Exercice 30

Vous devez créer un algorithme qui va permettre à l'utilisateur de saisir un nombre. Après sa saisie l'utilisateur pourra choisir de calculer la factorielle ou la somme des n premiers nombres en fonction de celui saisi.

Vous devriez créer trois fonctions :

- Une qui affiche le menu avec les différentes possibilités.
- Une qui calcule et retourne la factorielle d'un nombre saisi au clavier.
- Une qui calcule et retourne la somme des n premiers nombres saisis au clavier.

L'algorithme ne s'arrête pas lorsque l'utilisateur le demande.

Solution

to do ...

K. Les tableaux

Un tableau est un conteneur pouvant disposer de plusieurs valeurs. C'est un type complexe utilisable pour nos variables.

Exemple

Le tableau suivant contient cinq valeurs :

Le premier élément commence à l'indice 0 et le dernier à l'indice 4

Karim	Ahmad	Nabila	Omar	Khadija
[0]	[1]	[2]	[3]	[4]

Le symbole des tableaux est **[]**

```
ALGORITHME algoTableau
VARIABLES
    table[c] : TABLEAU CHAINE DE CARACTERES
DEBUT
    table[0] ← "Karim"
    table[1] ← "Ahmad"
    table[2] ← "Nabila"
    table[3] ← "Omar"
    table[4] ← "Khadija"
FIN
```

Table[c] est la déclaration d'un tableau contenant des valeurs de type chaîne de caractères.

Nous ne pouvons pas afficher directement le tableau complet, il faut afficher chacune des valeurs qu'il contient.

On peut afficher l'élément à une position donnée.

```
ALGORITHME algoTableau
VARIABLES
    table[c] : TABLEAU CHAINE DE CARACTERES
DEBUT
    table[0] ← "Karim"
    table[1] ← "Ahmad"
    table[2] ← "Nabila"
    table[3] ← "Omar"
    table[4] ← "Khadija"

    AFFICHER table[1]
    AFFICHER table[3]
FIN
```

Nous pouvons aussi afficher tous les éléments et pour cela nous devons boucler sur le tableau et afficher les valeurs une à une.

```
POUR compteur ALLANT DE 0 A taille(table)-1 PAR PAS DE 1 FAIRE
    AFFICHER table[compteur]
FINPOUR
```

taille(table) renvoie le nombre de valeurs que le tableau contient.

Remarque

Dans le cas d'un tableau contenant des valeurs entières on inscrira [n] et [c] pour le type chaîne de caractères.

a. Exercice 31

1. Réalisez l'algorithme suivant :

2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---

2. Essayez d'optimiser le pseudo-code

Solution

to do ...

b. Exercice 32

Réalisez un algorithme contenant 6 notes qui permet de calculer leur moyenne. Toutes les notes seront définies aléatoirement et seront contenues dans un tableau.

1. Sans fonction.

2. Avec une fonction qui fait le calcul de la moyenne d'un tableau passé en paramètres.

Solution

to do ...

c. Exercice 33

Créez une fonction qui affiche tous les éléments d'un tableau passé en paramètre. Vous pourrez ensuite utiliser cette fonction dans un exemple d'algorithme que vous réaliserez.

to do ...

d. Exercice 34

Créez une fonction qui permet d'indiquer si un tableau passé en paramètre ne contient que des valeurs paires ou impaires. La fonction retourne la valeur 'VRAI' ou 'FAUX'.

to do ...

L. Algorithme de Tri

Les tableaux permettent à nos variables d'avoir plusieurs valeurs qui sont positionnées dans des cases, de l'indice 0 à $\text{taille}(\text{table}) - 1$. Les tableaux ne peuvent pas avoir de trou dans leurs indices.

5	15	10	2	17
---	----	----	---	----

Si nous voulons trier notre tableau, par exemple pour avoir des valeurs croissantes, on va devoir utiliser des algorithmes spécifiques : les algorithmes de tri.

Il existe plusieurs méthodes pour trier un tableau, chacun présente des avantages par rapport aux autres. Ces différentes méthodes permettent de trier plus ou moins rapidement en fonction du tableau initial à trier.

Parmi les algorithmes existants, il y a notamment :

- Le tri d'insertion.
- Le tri de sélection.
- Le tri à bulle.

1. Le tri par insertion

On va utiliser le tableau suivant qui va falloir trier :

3	6	5	1	4	2
---	---	---	---	---	---

Le tri par insertion permet lors du parcours du tableau de placer chaque nouvel élément à la bonne position.

Voici le rendu par étape sur notre tableau :

Étape	Tableau					
0	3	6	5	1	4	2
1	3	6	5	1	4	2
2	3	5	6	1	4	2
3	1	3	5	6	4	2
4	1	3	4	5	6	2
5	1	2	3	4	5	6

Donc on va décaler les éléments d'un cran vers la droite pour faire la place à l'élément qu'on est en train de parcourir (table[i]).

Le pseudo-code est le suivant :

```

FONCTION triInsertion(table : ENTIER)
VARIABLES
    valeur : ENTIER
    compteur : ENTIER
DEBUT
    POUR i ALLANT DE 1 A taille(table)-1 PAR PAS 1 FAIRE
        valeur ← table[i]
        compteur ← i
        TANT QUE (compteur > 0 ET table[compteur-1] > valeur) FAIRE
            table[compteur] ← table [compteur - 1]
            compteur ← compteur - 1
        FINTANTQUE
        table[compteur] ← valeur
    FINPOUR
FIN

```

Explication :

Déroulement – **Étape 1**

Avant	3	6	5	1	4	2
	3	6	5	1	4	2

Après

= table[1] = 6

compteur = 1

table[compteur - 1] = table[1-1] = table[0] = 3 → 3 < 6 alors on ne fait rien

table[1] = 6

Déroulement – **Étape 2**

3	6	5	1	4	2
---	---	---	---	---	---

Avant

Après

3	5	6	1	4	2
---	---	---	---	---	---

valeur = table[2] = 5

compteur = 2

table[compteur - 1] = table[2 - 1] = table[1] = 6 → 6 > 5 alors

✓ on écrase le 5 en mettant le 6

✓ compteur = compteur - 1

table[compteur - 1] = table[1 - 1] = table[0] = 3 → 3 < 5 alors on ne fait rien

table[1] = 5

2. Le tri par sélection

On va utiliser le même tableau de la section précédente :

3	6	5	1	4	2
---	---	---	---	---	---

Le tri par sélection permet à chaque étape de placer la plus petite valeur restante du tableau à sa bonne position.

Voici le rendu par étape sur notre tableau :

Étape	Tableau					
0	3	6	5	1	4	2
1	1	6	5	3	4	2
2	1	2	5	3	4	6
3	1	2	3	5	4	6
4	1	2	3	4	5	6
5	1	2	3	4	5	6

On va chercher à chaque tour de boucle le plus petit élément restant dans le tableau.

```

FONCTION triSelection(table : ENTIER)
VARIABLES
    min : ENTIER
    tmp : ENTIER
DEBUT
    POUR i ALLANT DE 0 A taille(table) - 2 PAR PAS DE 1 FAIRE
        min ← i
        POUR j ALLANT DE i+1 A taille(table) - 1 PAR PAS DE 1 FAIRE
            Si table[j] < table[min]
                Alors min ← j
            FINSI
        FINPOUR
        Si min ≠ i Alors
            tmp = table[i]
            table[i] = table[min]
            table[min] = tmp
        FINSI
    FINPOUR
FIN

```

Explication :

Déroulement – Étape 1

Avant

3	6	5	1	4	2
---	---	---	---	---	---

1	6	5	3	4	2
---	---	---	---	---	---

Après

0 / table[0] = 3

table[j] < table[min] = table[1] < table[0] → 6 > 3 Alors on ne fait rien

table[j] < table[min] = table[2] < table[0] → 5 > 3 Alors on ne fait rien

table[j] < table[min] = table[3] < table[0] → 1 < 3 Alors min ← j = 3

table[j] < table[min] = table[4] < table[3] → 4 > 1 Alors on ne fait rien

table[j] < table[min] = table[5] < table[3] → 2 > 1 Alors on ne fait rien

min ≠ i → 3 ≠ 0 alors

tmp = table[i] = table[0] = 3

table[i] = table[min] = table[3] = 1

table[min] = tmp = 3

Déroulement – Étape 2

Avant

1	6	5	3	4	2
---	---	---	---	---	---

1	2	5	3	4	6
---	---	---	---	---	---

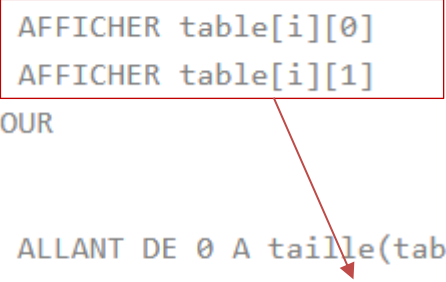
Après

1 / table[1] = 6

Pour afficher tous les éléments d'un tableau en faisant :

```
ALGORITHME tableMulti
VARIABLES
    table[n][n] : TABLEAU ENTIER
DEBUT
    table ← [[4,17],[11,12],[5,3]]
    POUR i ALLANT DE 0 A taille(table)-1 PAR PAS DE 1 FAIRE
        AFFICHER table[i][0]
        AFFICHER table[i][1]
    FINPOUR
FIN

POUR j ALLANT DE 0 A taille(table[i])-1 PAR PAS DE 1 FAIRE
    AFFICHER table[i][j]
FINPOUR
```



a. Exercice 35

Vous devez réaliser un tableau contenant un tableau de prénom et nom. Votre algorithme devra afficher le nom et le prénom de chaque personne séparée par un espace. Voici les personnes à insérer :

- Bouchra Nadim
- Karim El alami
- Nabila Najem

Solution

- to do ...