

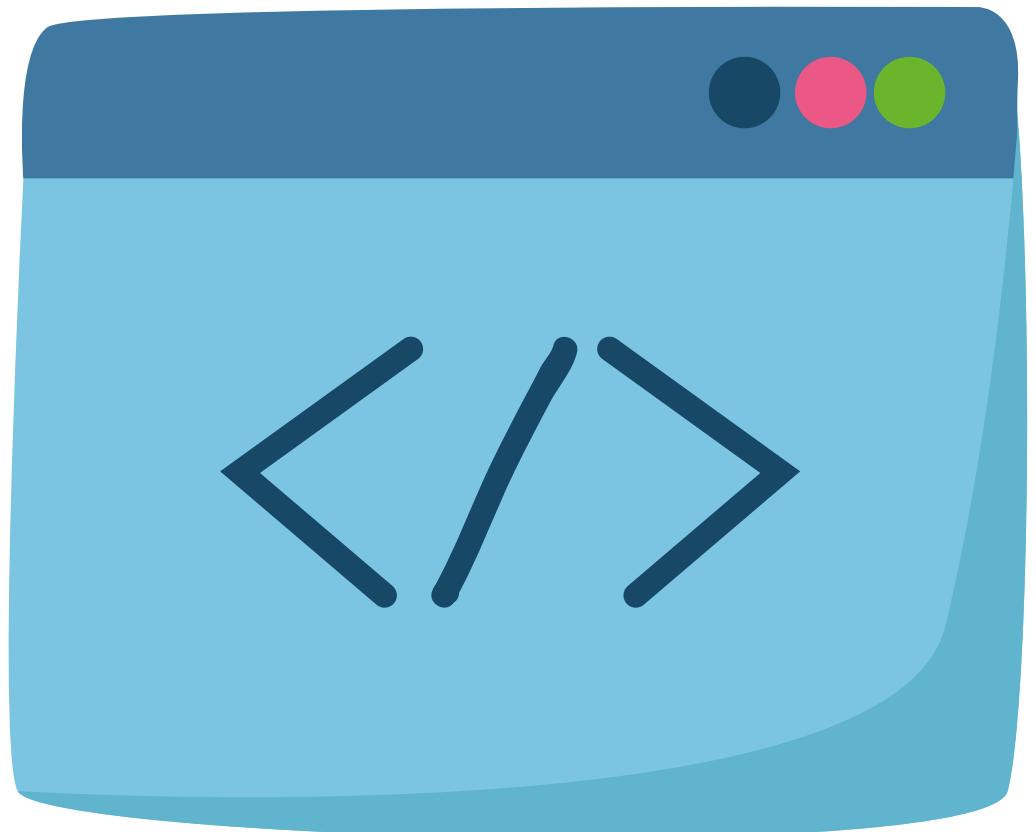
ALGORITHMIQUE

Introduction à l'algorithme

LA LOGIQUE DU CODE

Objectifs du cours

- Obtenir les notions de base en algorithmique
- Comprendre les types de données
- Comprendre la notion de sous-programmes
- Apprendre le nommage des variables
- Écrire vos propres algorithmes



QU'EST-CE QU'UN ALGORITHME ?

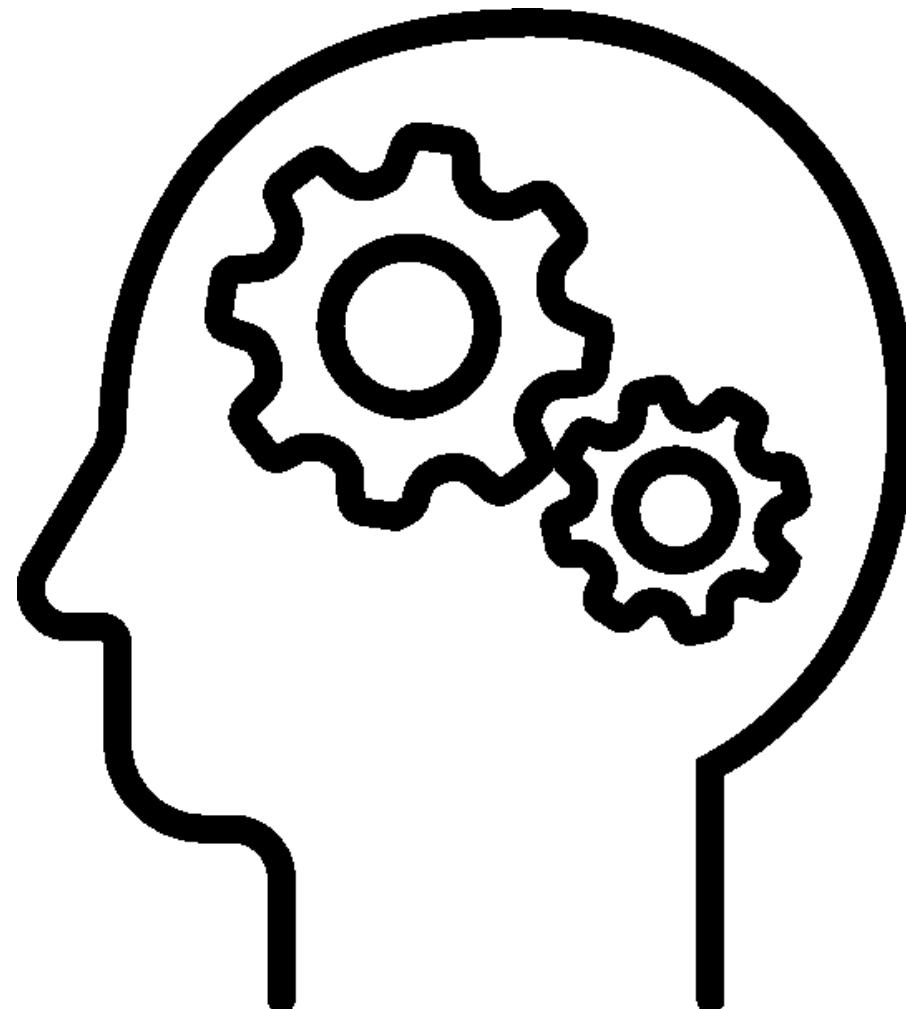
Mot dérivé du mathématicien persan Al-khwarizmi.

Un programme qui prend des données en entrée, effectue un traitement et fournit des données en sortie.

Un programme : une série d'instruction qui peuvent s'exécuter en séquence ou en parallèle en implémentant un algorithme



UN COURS D'ALGO, À QUOI ÇA SERT ?



Faire travailler la machine à notre place.

Savoir comment faire travailler la machine :

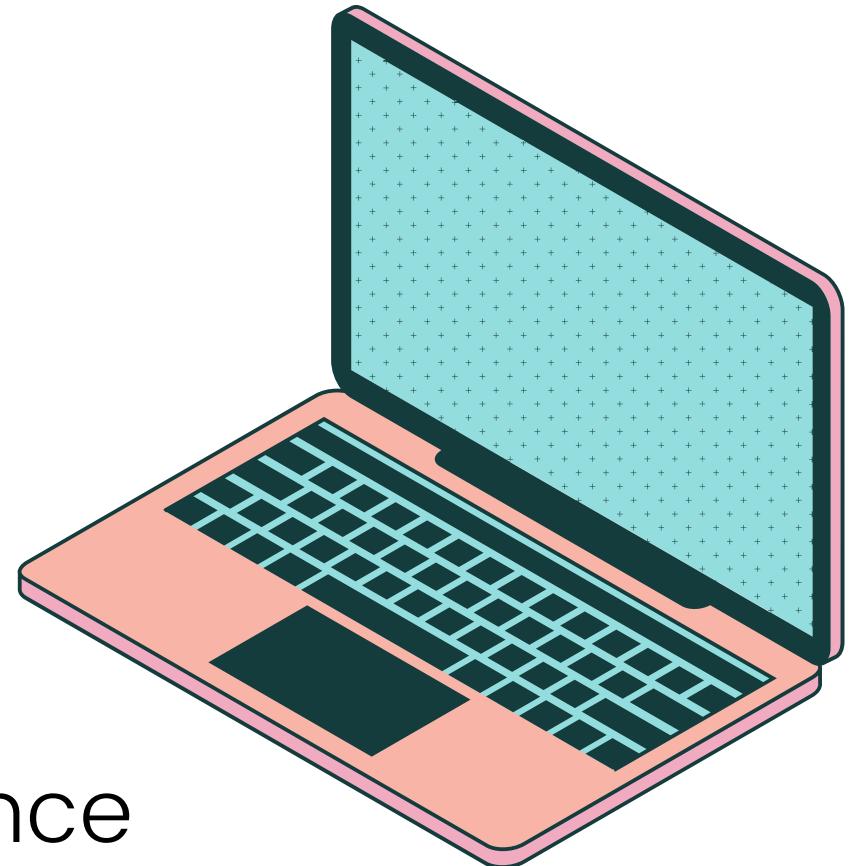
- Savoir expliciter son raisonnement
- Savoir formaliser son raisonnement
- Concevoir et écrire des algorithmes

UN COURS D'ALGO, À QUOI ÇA SERT ?

Vous donner les bases afin de savoir expliquer comment faire un travail sans la moindre ambiguïté, en utilisant un langage simple appelé instructions, en une suite finie d'actions à entreprendre tout en respectant une chronologie imposée.

EXEMPLES D'ALGORITHMES

- Une recette de cuisine
- La notice de montage d'un meuble Ikea
- Additionner deux nombres
- Trouver l'âge d'une personne grâce à sa date de naissance



LES PROBLÈMES FONDAMENTAUX EN ALGORITHMIQUE

La complexité

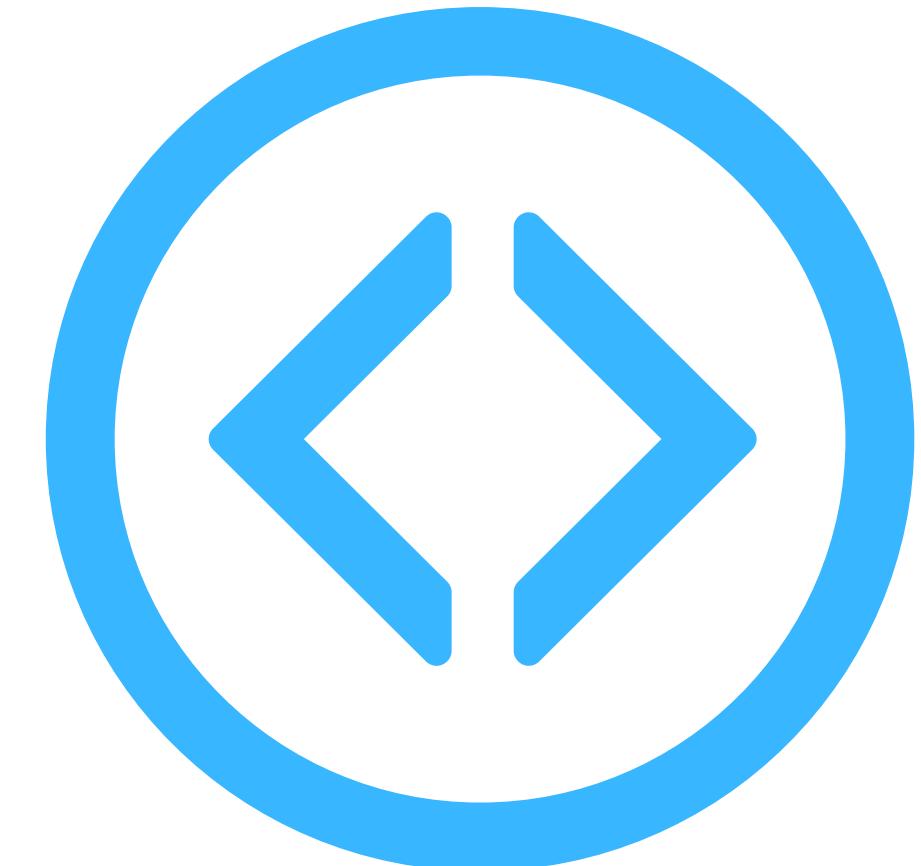
- Temps d'exécution
- Mémoire nécessaire

La calculabilité

- Des tâches sans algorithme ?
- Déjà existant ?

La correction

- Réponse au besoin



EXEMPLE DE LANGAGE ALGORITHMIQUE

```
1 ALGORITHME ElèveAuCarré
2 {Cet algorithme calcul le carré du nombre que lui fournit l'utilisateur}
3
4 VARIABLES unNombre, sonCarré : entiers
5
6 DEBUT
7
8     afficher("Quel nombre voulez-vous éllever au carré ?")
9     saisir(unNombre)
10
11     sonCarré <- unNombre x unNombre
12
13     afficher("Le carré de ", unNombre)
14     afficher("c'est ", sonCarré)
15 FIN|
```

LES ÉTAPES D'UN ALGORITHME

Préparation au traitement des données

Traitement du problème pas à pas en décomposant

Édition du résultat

COMPOSITION D'UN ALGORITHME

```
1 ALGORITHME NomAlgorithme  
2 {commentaire de l'algorithme}  
3  
4 DEBUT  
5     ACTIONS  
6 FIN
```

```
1 ALGORITHME DireBonjour  
2 {Dire bonjour en anglais}  
3  
4 DEBUT  
5     afficher("Hello world!")  
6 FIN
```

Mot clé ALGORITHME
{COMMENTAIRE}
DÉBUT

INSTRUCTIONS
FIN
VARIABLES

DÉCLARATION DES DONNÉES

Déclarer vos variables avant le mot-clé DÉBUT

Utiliser le mot-clé VARIABLE pour déclarer vos variables

Déclaration d'une variable :

- VARIABLE <nom de la variable> : type de la variable

La variable permet de réserver de l'espace mémoire pour stocker des données.

DÉCLARATION DES DONNÉES

Déclarer plusieurs variable de même type :

```
VARIABLE    nombreUn, nombreDeux : entiers
```

Déclarer plusieurs variables avec des types différents :

```
VARIABLE    nombreUn, nombreDeux : entiers  
nom, prenom : chaînes de caractères
```

DÉCLARATION DES DONNÉES

Les variables qui ne changent jamais de valeurs sont appelées CONSTANTE.

Il est possible de déclarer ce type de données en écrivant :

CONSTANTE <nom de la constante> : type <- valeur ou expression

Par exemple :

CONSTANTE max: entier <- 10

maxFoisDeux : entier <- max x 2

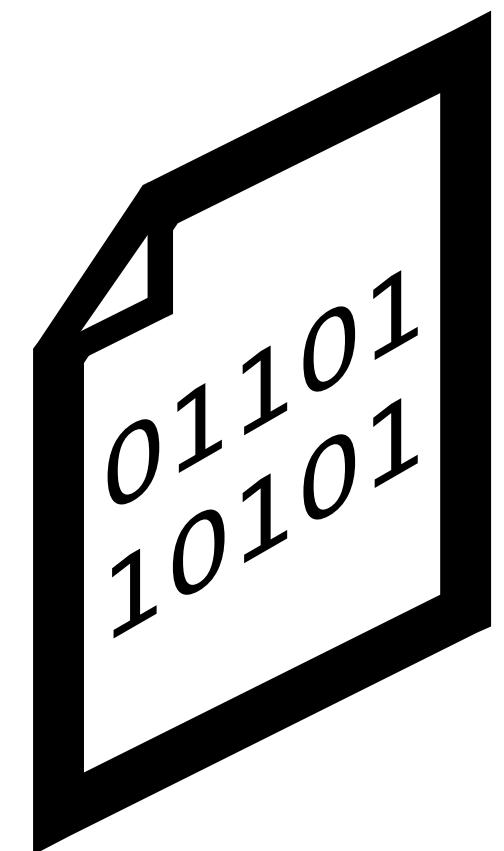
LES TYPES DE DONNÉES

Caractériser une variable en indiquant la nature de son contenu.

Un type défini ne peut plus changer, sauf exception de certains langages.

4 types de données :

- Entier : données numérique positive ou négative
- Réel : nombres à virgules
- Booléen : VRAI ou FAUX
- Chaîne de caractères : Données alphanumériques



RÈGLES DE NOMMAGE

Règles à respecter pour le nommage des variables :

1. Commencer par une lettre ou un underscore _
2. Peut contenir des chiffres, lettres, underscores, pas de tirets ou points
3. Pas de mots clés (POUR, TANT QUE, ...)
4. Sensible à la casse : maVar différent de mavar
5. Si plusieurs mots : majuscule sur première lettre de chaque mot sauf premier mot. Ex : monAdresse plutôt que monadresse
6. Le nom doit décrire la variable
7. Prononçable
8. Aussi court que possible, mais aussi long que nécessaire

RÈGLES DE NOMMAGE

La "portée" ou le "scope" :

- Selon l'endroit où vous allez utiliser votre variable son nommage sera différent
- Plus la variable a de chance d'être utilisée à divers endroits dans votre programme (sa portée), plus le nommage de celle-ci sera important.

Choisissez un nom de variable qui soit le plus explicite possible

RÈGLES DE BONNES PRATIQUES

Élément	Convention	Exemple
Classe / interface	Noms / groupes nominaux Eviter autant que possible les mots parasites (manager, service...)	Recrutement, ConsultantRecruteur, ExperiencesCandidat
Variable de classe (field)	Noms / groupes nominaux	nom, adresse, preferencesDeMission
Variable locale (variable)	Noms / groupes nominaux Idéalement dérivé du nom de la classe	consultantsSurParis, clientActuelDuConsultant
Booléen	Prédicat d'état : verbe être + complément	estAccepte, estRefuse, estEnMission
Fonctions / méthode	Verbes à l'impératif Si retournant un booléen : Prédicat	annulerEntretien(), enregistrer(), estCoopte()
Enumération / type fermé	Adjectif ou nom	EN_COURS, ACCEPTE
Événement	Verbe au participe passé	integrationTerminee, cooptationValidee, missionChoisie

LECTURE ET ÉCRITURE DE DONNÉES

Permettre à votre utilisateur d'entrer des informations (écriture)

- saisir(<nom de la donnée>)

Permettre de visualiser les données en mémoire (lecture)

- afficher(<nom de la donnée>)

Exemple :

saisir(unNombre)

afficher("Le nombre est : ", unNombre)

LA PHASE D'ANALYSE

Extraire de l'énoncé du problème les éléments de modélisation.

Souligner en différentes couleurs afin d'identifier :

- Le but du programme (traitement à réaliser)
- Les données qui sont prises en entrées
- Les résultats en sortie

EXEMPLE D'ÉNONCÉ D'UN PROBLÈME

On souhaite calculer et afficher, à partir d'un prix hors taxe saisi, la TVA ainsi que le prix TTC.

Le montant TTC dépend :

- Du prix HT
- Du taux de TVA qui est de 20%

EXEMPLE D'ÉNONCÉ D'UN PROBLÈME

On souhaite **calculer et afficher**, à partir d'un prix hors taxe saisi, la TVA ainsi que le prix **TTC**.

Le montant TTC dépend :

- Du prix HT
- Du taux de TVA qui est de 20%

ALGORITHME CALCUL TVA

```

1 ALGORITHME calculTVA
2 {Saisir un prix HT et affiche le prix TTC correspondant}
3
4 CONSTANTES TVA:réel <- 20
5 Titre:Chaine <- "Résultat : "
6
7 VARIABLES prixHT, prixTTC, montantTVA:réel
8
9 DEBUT
10 afficher("Entrez un prix HT :")
11 saisir(prixHT)
12
13 prixTTC <- prixHT x (1 + TVA/100)
14 montantTVA <- prixTTC - prixHT
15
16 afficher(Titre)
17 afficher(prixHT, "€ HT + TVA ", montantTVA, "€
    devient ", prixTTC, "€ TTC")
18 FIN

```

```

1 Nom de l'algorithme
2 Description de l'algorithme
3
4 Déclaration des valeurs qui ne changerons jamais
5 Ce sont nos constantes
6
7 Déclarations des données variables
8
9 Déclaration du début de l'algorithme
10 Affiche le message : "Entrez un prix HT :"
11 Saisi d'un prix, attribué à la variable prixHT
12
13 calcul du prix TTC, attribué à la var prixTTC
14 calcul du montant de la TVA
15
16 Affiche le message "Résultat : "
17 Affiche le résultat des calculs, ex : 8€ HT +
    TVA 1,6€ devient 9,6€ TTC
18 Déclaration de la fin de l'algorithme

```

EXERCICE

Quelles vont être les différentes valeurs des variables de l'algorithme suivant ?

```
1 ALGORITHME maSequence
2 {Instructions séquentielles}
3
4 CONSTANTE SEUIL:réel <- 13.25
5
6 VARIABLES valA, valB:réel
7           compteur:entier
8           mot, tom:chaînes
9
10 DEBUT
11   valA<-0.56
12   valB<-valA
13   valA<-valA x (10.5 + SEUIL)
14   compteur<-1
15   compteur<-compteur + 10|
16   mot<-"Bonjour"
17   tom<-"Au revoir !"
18 FIN
```

EXERCICE 2

```
1 ALGORITHME echangeValeur
2 {Echange les valeurs de deux variables}
3
4 VARIABLES      valA, valB:réel
5
6 DEBUT
7     afficher("Donnez deux valeurs")
8     saisir(valA, valB)
9
10    afficher("Vous avez entré : ", valA, " et ", valB)
11
12    valA<-valB
13    valB<-valA
14
15    afficher("Maintenant vos données sont : ", valA, " et ", valB)
16 FIN
```

Soit, l'algorithme suivant :

Répondez à ces questions :

- Que devrait faire cet algorithme ?
- Que fait cet algorithme ?

EXERCICE

Modifiez l'algorithme précédent afin qu'il effectue ce pour quoi il a été conçu.

CORRECTION

```
1 ALGORITHME echangeValeur
2 {Echange les valeurs de deux variables}
3
4 VARIABLES      valA, valB, valTemp:réel
5
6 DEBUT
7     afficher("Donnez deux valeurs")
8     saisir(valA, valB)
9
10    afficher("Vous avez entré : ", valA, " et ", valB)
11
12    valTemp<- valA
13    valA<-valB
14    valB<-valTemp
15
16    afficher("Maintenant vos données sont : ", valA, " et ", valB)
17 FIN|
```

ALLER PLUS LOIN AVEC L'ALGO

- Déclaration d'algorithme
- Les commentaires
- Déclaration des variables
- Déclaration des constantes
- Affichage d'information
- Saisie de données

Les conditions :

Lancer une partie de votre programme seulement lorsque les conditions sont remplies

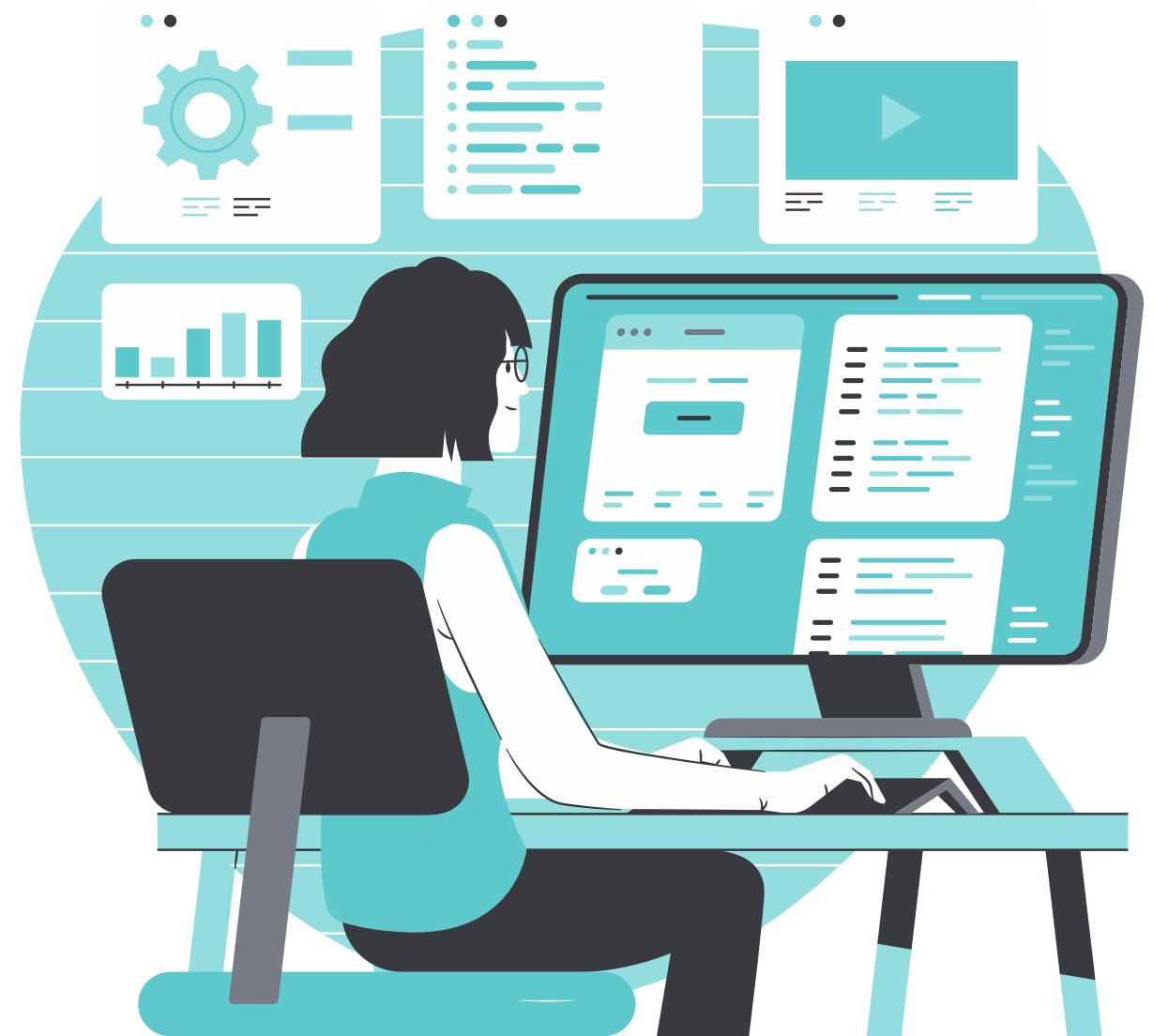


LES CONDITIONS

Les conditions :

- SI... ALORS... FIN SI
- SI... ALORS... SINON... FIN SI

Ne pas oublier de définir la fin d'une condition



EXEMPLE CONDITION

```
1 ALGORITHME conditionSimple
2 {Cet algorithme demande une valeur entière et affiche son double
   si cette donnée est inférieur à un seul donné.}
3
4 CONSTANTE    seuil:entier<-10
5 VARIABLES     val:entier
6
7 DEBUT
8     afficher("Entrez un nombre entier")
9     saisir(val)
10
11    SI val PLUS PETIT QUE seuil ALORS
12        afficher("Voici son double : ", val x2)
13    SINON
14        afficher("Voici la valeur de base : ", val)
15    FIN SI
16 FIN
```

LES CONDITIONS

1 <	PLUS PETIT QUE
2 >	PLUS GRAND QUE
3 <=	PLUS PETIT OU EGAL A
4 >=	PLUS GRAND OU EGAL A
5 =	EGAL A
6 !=	DIFFERENT DE

LES CONDITIONS

```
1 ALGORITHME conditionSimple
2 {Cet algorithme demande une valeur entière et affiche son double
   si cette donnée est inférieur à un seul donné.}
3
4 CONSTANTE    seuil:entier<-10
5 VARIABLES    val:entier
6
7 DEBUT
8     afficher("Entrez un nombre entier")
9     saisir(val)
10
11    SI val < seuil ALORS
12        val<- val x2
13    FIN SI
14
15    afficher("Voici la valeur de val : ", val)
16 FIN
```

LES CONDITIONS IMBRIQUÉES

Imbrication de conditions possible.

Écrivez un algorithme qui affiche :

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

LES CONDITIONS IMBRIQUÉES

```
1 ALGORITHME conditionImbriquée
2 {Affiche un texte en fonction de la note}
3
4 VARIABLES    note:entier
5
6 DEBUT
7     SI note >= 12 ALORS
8         afficher("Reçu avec mention Assez Bien")
9     SINON
10        SI note >= 10 ALORS
11            afficher("Reçu mention passable")
12        SINON
13            afficher("Insuffisant")
14        FIN SI
15    FIN SI
16 FIN|
```

SÉLECTION À CHOIX MULTIPLES

Tester une variable pour effectuer une instruction en fonction de la valeur de celle-ci.

Utilisation du mot-clé SELON

Écrire simplement les conditions sans imbrications

Ex. On souhaite afficher un message selon le sexe, on peut utiliser SELON.

SÉLECTION À CHOIX MULTIPLES

```
1 ALGORITHME afficheGenre
2 {Affiche un texte en fonction du genre d'une personne}
3
4 VARIABLES    genre:chaine
5
6 DEBUT
7     SELON genre
8         "M": afficher("Monsieur")
9         "Mme": afficher("Madame")
10        "Mlle": afficher("Mademoiselle")
11        autres: afficher("Non genre")
12 FIN
```

SÉLECTION À CHOIX MULTIPLES

```
1 ALGORITHME afficheGenre
2 {Affiche un texte en fonction du genre d'une personne}
3
4 VARIABLES    genre:chaine
5
6 DEBUT
7     SI genre = "M"
8         ALORS
9             afficher("Monsieur")
10        SINON SI genre = "Mme"
11            ALORS
12                afficher("Madame")
13            SINON SI genre = "Mlle"
14                ALORS
15                    afficher("Mademoiselle")
16                SINON
17                    afficher("Non genre")
18                FIN SI
19            FIN SI
20        FIN SI
21 FIN
```

EXERCICE

Vous êtes en charge d'une classe de 24 étudiants. Vous souhaitez classer les étudiants selon leur genre afin d'apprendre plus facilement à les connaître.

Parmi ces étudiants :

- 5 étudiantes sont célibataires
- 4 étudiants sont non genrés
- Il y a en tout 10 étudiantes

Répondez aux questions suivantes :

1. Combien d'instructions sont nécessaires pour évaluer l'exécution avec le premier algorithme ?
2. Combien d'instructions sont nécessaires pour évaluer l'exécution avec le deuxième algorithme ?

LA BOUCLE POUR

S'utilise lorsque vous avez besoin de répéter une itération plusieurs fois.

La boucle POUR évite d'utiliser des conditions SI par centaine, voir plus en fonction du nombre de répétitions.

La boucle POUR permet d'effectuer le travail de test à notre place.

LA BOUCLE POUR

```
1 ALGORITHME FaitLeTotal
2 {Effectue la somme des valeurs saisit}
3
4 VARIABLES    nbVal, count:entiers
5           valeur, totalValeurs:réels
6
7 DEBUT
8     afficher("Combien de valeur voulez-vous saisir ?")
9     saisir(nbVal)
10
11    totalValeurs<-0
12
13    POUR count<-1 à nbVal FAIRE
14        afficher("Donnez une valeur : ")
15        saisir(valeur)
16        totalValeurs<-totalValeurs + valeur
17    FIN POUR
18
19    afficher("Le total des ", nbVal, " valeurs est ", totalValeurs)
20 FIN
```

LA BOUCLE POUR

La boucle POUR s'écrit :

```
POUR <variable> <- valeur initiale A valeur finale FAIRE  
    instruction  
FIN POUR
```

La boucle POUR avec une donnée supplémentaire :

```
POUR <variable> <- valeur initiale A valeur finale [PAR PAS DE X] FAIRE  
    instruction  
FIN POUR
```

LA BOUCLE POUR

```
1 ALGORITHME FaitLeTotal
2 {Effectue la somme des valeurs saisit}
3
4 VARIABLES    nbVal, count:entiers
5           valeur, totalValeurs:réels
6
7 DEBUT
8     afficher("Combien de valeur voulez-vous saisir ?")
9     saisir(nbVal)
10
11    totalValeurs<-0
12
13    POUR count<-1 à nbVal PAR PAS DE 2 FAIRE
14        afficher("Donnez une valeur : ")
15        saisir(valeur)
16        totalValeurs<-totalValeurs + valeur
17    FIN POUR
18
19    afficher("Le total des ", nbVal, " valeurs est ", totalValeurs)
20 FIN
```

LA BOUCLE POUR : EN RÉSUMÉ

Elle a pour fonction de répéter une suite d'instructions un certain nombre de fois.

Elle est utilisée quand le nombre d'itérations est connu à l'avance.

L'instruction POUR :

- Initialise une variable de boucle (le compteur)
- Incrémente (ajoute) à cette variable la valeur de PAS
- Effectue les instructions données

EXERCICE 01

Écrivez un algorithme qui affiche tous les nombres pairs de 0 jusqu'à 100.

EXERCICE 02

Écrivez un algorithme pour un mini-jeu. Le but du jeu est de faire deviner un nombre de votre choix à un joueur qui ne le connaît pas. Pour cela, vous allez :

1. Initialiser un nombre "secret" entre 1 et 100. Ce nombre sera déjà défini dans votre programme et ne pourra pas être modifié.
2. Votre programme doit demander au joueur de deviner le nombre
3. Le joueur n'a droit qu'à 10 tentatives
4. À chaque tentative, le programme devra dire "Plus grand", "Plus petit", ou "Gagné"

BONUS :

- Modifiez votre algorithme pour que le programme affiche également "Perdu" si le joueur dépasse les 10 essais.

EXERCICE 01 : CORRIGÉ

```
1 ALGORITHME afficheNombrePair
2 {Algorithme qui affiche les nombres pairs jusqu'à 100}
3
4 VARIABLES      compte:entier
5
6 DEBUT
• 7   POUR compte<-0 à 100 PAR PAS DE DEUX FAIRE
     afficher('Nombre pair : ', compte)
9   FIN POUR
10 FIN|
```

EXERCICE 02 : CORRIGÉ

```
1 ALGORITHME devineNombre
2 {Programme qui permet à un joueur de tenter de deviner un nombre
   compris entre 1 et 100}
3
4 VARIABLES     compte:entier <- 10
5             joueur, tentative: entier
6
7 CONSTANTE     nombreSecret:entier <- 18
8
9 DEBUT
10    POUR tentative<-1 à compte FAIRE
11        afficher("Devinez le nombre (essai ", tentative, "/10) : ")
12        saisir(joueur)
13
14        SI joueur PLUS PETIT QUE nombreSecret
15            ALORS
16                afficher("Plus grand")
17            SINON SI joueur PLUS GRAND QUE nombreSecret
18                ALORS
19                    afficher("Plus petit")
20                SINON
21                    afficher("Gagné !")
22                FIN SI
23            FIN SI
24        FIN POUR
25 FIN
```

EXERCICE 02 BONUS : CORRIGÉ

```
1 ALGORITHME devineNombre
2 {Programme qui permet à un joueur de tenter de deviner un nombre compris
entre 1 et 100}
3
4 VARIABLES    compte:entier <- 11
5           joueur, tentative: entier
6
7 CONSTANTE    nombreSecret:entier <- 18
8
9 DEBUT
10   POUR tentative<-1 à compte FAIRE
11     SI tentative EST PLUS PETIT QUE compte
12       ALORS
13         afficher("Devinez le nombre (essai ", tentative, "/10) : ")
14         saisir(joueur)
15
16     SI joueur PLUS PETIT QUE nombreSecret
17       ALORS
18         afficher("Plus grand")
19       SINON SI joueur PLUS GRAND QUE nombreSecret
20         ALORS
21           afficher("Plus petit")
22         SINON
23           afficher("Gagné !")
24         FIN SI
25       FIN SI
26     SINON
27       afficher("Vous avez perdu")
28     FIN SI
29   FIN POUR
30 FIN
```

LA BOUCLE TANT QUE... FAIRE

La boucle TANT QUE permet d'effectuer une boucle dont le nombre d'itérations n'est pas connu à l'avance.

On commence cette boucle en effectuant l'amorçage :

- Initiation de la ou des variables de condition

Ensuite, vous pouvez commencer la boucle

LA BOUCLE TANT QUE... FAIRE

```
1 ALGORITHME afficheNombrePair
2 {Algorithme qui affiche les nombres pairs jusqu'à 100}
3
4 VARIABLES      compte:entier
5
6 DEBUT
• 7     compte<-0
8     TANT QUE compte EST DIFFERENT DE 100 FAIRE
9         afficher("Nombre pair : ", compte)
10        compte<- compte + 2
11     FIN TANT QUE
12 FIN
```

LA BOUCLE TANT QUE... FAIRE

La boucle TANT QUE est appelée structure itérative universelle, car n'importe quel contrôle d'itération peut se traduire par un TANT QUE.

À savoir : Vous pouvez renseigner plus d'une condition dans votre boucle TANT QUE

La valeur d'arrêt de la boucle n'est jamais traitée, et donc jamais comptabilisé.

EXERCICE

Écrire le même mini-jeu que précédemment mais au lieu d'utiliser la boucle
POUR vous devez utiliser la boucle TANT QUE

Lorsque vous avez fini, modifiez l'algorithme afin de prendre en compte la
défaite.

EXERCICE CORRIGÉ 1ERE PARTIE

```
1 ALGORITHME devineNombre
2 {Programme qui permet à un joueur de tenter de deviner un nombre compris entre 1 et 100}
3
4 VARIABLES     compte:entier <- 11
5             joueur, tentative: entier
6
7 CONSTANTE    nombreSecret:entier <- 18
8
9 DEBUT
10    tentative<-1
11    TANT QUE tentative EST PLUS PETIT QUE compte ET joueur DIFFERENT DE nombreSecret FAIRE
12        afficher("Devinez le nombre (essai ", tentative, "/10) : ")
13        saisir(joueur)
14
15    SI joueur PLUS PETIT QUE nombreSecret
16        ALORS
17            afficher("Plus grand")
18        SINON SI joueur PLUS GRAND QUE nombreSecret
19            ALORS
20                afficher("Plus petit")
21            SINON
22                afficher("Gagné !")
23            FIN SI
24        FIN SI
25        tentative<- tentative + 1
26    FIN TANT QUE
27 FIN
```

EXERCICE CORRIGÉ 2E PARTIE

```
1 ALGORITHME devineNombre
2 {Programme qui permet à un joueur de tenter de deviner un nombre compris entre 1 et 100}
3
4 VARIABLES    compte:entier <= 11
5           joueur, tentative: entier
6
7 CONSTANTE    nombreSecret:entier <- 18
8
9 DEBUT
10    tentative<-1
11    TANT QUE tentative EST PLUS PETIT QUE compte ET joueur DIFFERENT DE nombreSecret FAIRE
12        afficher("Devinez le nombre (essai ", tentative, "/10) : ")
13        saisir(joueur)
14
15        SI joueur PLUS PETIT QUE nombreSecret
16            ALORS
17                afficher("Plus grand")
18            SINON SI joueur PLUS GRAND QUE nombreSecret
19                ALORS
20                    afficher("Plus petit")
21                SINON
22                    afficher("Gagné !")
23                FIN SI
24            FIN SI
25            tentative<- tentative + 1
26        FIN TANT QUE
27
28        SI(tentative EST EGAL A compte ET joueur DIFFERENT DE nombreSecret)
29            afficher("Vous avez perdu")
30        FIN SI
31    FIN
```

COMPARAISON ENTRE LES BOUCLE POUR ET TANT QUE

La boucle POUR :

- Initialise un compteur
- Incrémente le compteur à chaque pas
- Vérifie que le compteur ne dépasse pas la borne supérieure

La boucle TANT QUE doit :

- Initialiser un compteur avant, appelé amorçage
- Incrémenter le compteur à chaque pas, appelé relance
- Vérifier que le compteur ne dépasse pas la borne supérieure, c'est le test de boucle.

QUAND CHOISIR POUR OU TANT QUE

Lorsque vous avez un nombre d'itération connu à l'avance, vous devez choisir POUR, c'est le cas lorsqu'on :

- Parcours un tableau de données
- Effectue des tests sur un nombre donné de valeurs

Lorsque vous avez un évènement particulier sur lequel vous souhaitez que votre boucle s'arrête, vous devez choisir la boucle TANT QUE.

LA BOUCLE RÉPÉTER

La boucle REPETER... TANT QUE :

- Exécuter une suite d'instructions au moins une fois.
- Répéter la suite d'instructions tant que la condition est remplie.

Quoi qu'il arrive la boucle s'exécutera au moins une fois.

LA BOUCLE RÉPÉTER

```
1 ALGORITHME valeurPositive
2 {Programme qui a besoin d'une valeur positive}
3
4 VARIABLES    valeur:entier
5
6 DEBUT
7     REPETER
8         afficher("Donnez une valeur positive non nulle : ")
9         saisir(valeur)
10        TANT QUE valeur INFERIEUR OU EGAL A 0
11
12        afficher("La valeur positive non nulle que vous avez saisie est ", valeur)
13 FIN
```

LA BOUCLE RÉPÉTER

```
1 ALGORITHME valeurPositive
2 {Programme qui a besoin d'une valeur positive}
3
4 VARIABLES    valeur:entier
5
6 DEBUT
7     afficher("Donnez une valeur positive non nulle : ")
8     saisir(valeur)
9     TANT QUE valeur INFERIEUR OU EGAL A 0 FAIRE
10        afficher("Donnez une valeur positive non nulle : ")
11        saisir(valeur)
12    FIN TANT QUE
13
14    afficher("La valeur positive non nulle que vous avez saisie est ", valeur)
15 FIN
```

COMPARAISON ENTRE LES BOUCLE REPETER ET TANT QUE

La boucle TANT QUE :

- Vérifie la condition avant chaque exécution du traitement
- Le traitement peut ne jamais être exécuté
- La condition porte surtout sur la saisie de nouvelles variables (appelée relance)

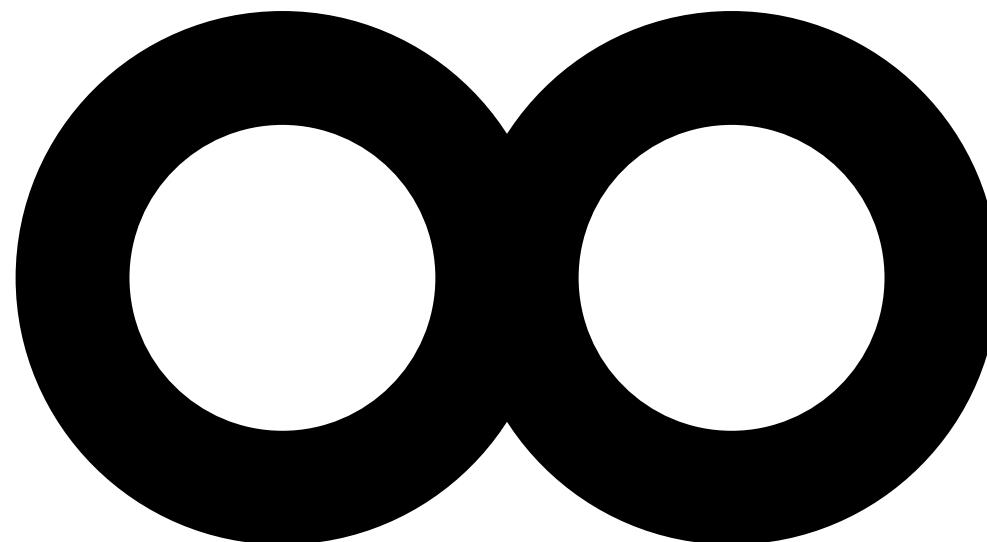
La boucle REPETER....TANT QUE :

- Vérifie la condition après chaque exécution du traitement
- Quoi qu'il arrive le traitement est exécuté au moins une fois
- La condition porte surtout sur le résultat du traitement

EXERCICE

Écrire un algorithme qui permet de saisir des données et s'arrêter dès que leur somme dépasse 500.

Réalisez l'algorithme avec la boucle REPETER, puis recommencez avec la boucle TANT QUE



CORRECTION 1ERE PARTIE

```
1 ALGORITHME calculSomme
2 {Programme calcul la somme de données jusqu'à 500}
3
4 VARIABLES      valeur, somme:entier
5
6 DEBUT
• 7     somme< -0
8     REPETER
9         saisir(valeur)
10        somme<- somme + valeur
11    TANT QUE somme PLUS PETIT QUE 500
12 FIN
```

CORRECTION 2E PARTIE

```
1 ALGORITHME calculSomme
2 {Programme calcul la somme de données jusqu'à 500}
3
4 VARIABLES    valeur, somme:entier
5
6 DEBUT
7     saisir(valeur)
8     somme<- valeur
9     TANT QUE somme PLUS PETIT QUE 500 FAIRE
10        saisir(valeur)
11        somme<- somme + valeur
12    FIN TANT QUE
13 FIN
```

LES FONCTIONS

Un algorithme qui peut être utilisé dans d'autres algorithme ou plusieurs fois dans le même programme.

La fonction peut récupérer des informations grâce à des paramètres.

La fonction ne peut retourner qu'une seule information

Les fonction s'écrivent comme pour les algorithmes. A la place du mot-clé ALGORITHME vous devez écrire FONCTION.

LES FONCTIONS

```
1 FONCTION calculSomme(nombre1, nombre2)
2 {Calcul la somme de deux nombre}
3
4 VARIABLES      somme:entier
5
6 DEBUT
7   somme <- nombre1 + nombre2
8   RETOURNER somme
9 FIN
```

LES FONCTIONS

```
1 FONCTION calculSomme(nombre1, nombre2)
2 {Calcul la somme de deux nombre}
3
4 VARIABLE     somme:entier
5
6 DEBUT
• 7     somme <- nombre1 + nombre2
8     RETOURNER somme
9 FIN
10
11
12 ALGORITHME main
13 {Programme qui utilise la fonction pour calculer une somme}
14
15 VARIABLE     resultat:entier
16
17 DEBUT
18     resultat <- calculSomme(10, 8)
19     afficher("Le résultat est : ", resultat)
20 FIN
```

LES PROCÉDURES

Lorsque vous ne souhaitez pas retourner une information, mais exécuter un traitement spécifique, vous pouvez utiliser une PROCEDURE à la place d'une FONCTION.

Les procédures vous permettent d'afficher des informations si vous le souhaitez.

LES PROCÉDURES

```
1 ALGORITHME main
2 {Programme qui utilise une procédure}
3
4 PROCEDURE calculSomme(nombre1: Entier, nombre2: Entier)
5 {Calcul la somme de deux nombres}
6
7 VARIABLE     somme: Entier
8
9 DEBUT
•10    somme <- nombre1 + nombre2
11    afficher("Le résultat est : ", somme)
12 FIN
13
14 VARIABLE resultat: Entier
15
16 DEBUT
17    calculSomme(10, 8)
18 FIN
```

EXERCICE 01

Écrire une fonction qui permet d'afficher le tableau de multiplication (jusqu'à 10) d'un entier positif choisi par l'utilisateur

EXERCICE 02

Écrire un algorithme qui permet à l'utilisateur d'entrer deux nombres, une fonction doit calculer la somme ainsi que le produit de ces deux nombres et afficher si ceux-ci sont positifs ou négatifs.