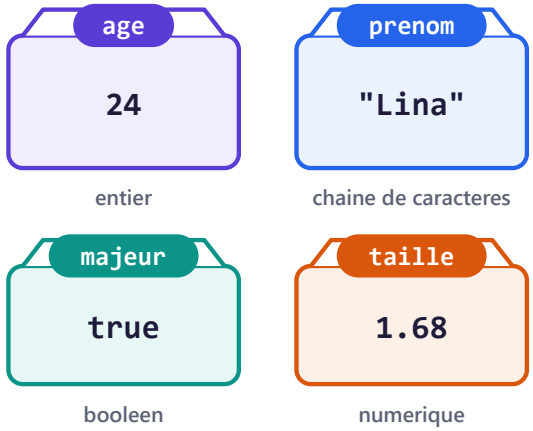
CHAPITRE 1

Les variables et l'affectation

Les bases de tout algorithme



AU PROGRAMME

1. Qu'est-ce qu'un algorithme ?
2. Les variables
3. Les types de variables
4. Déclarer et affecter
5. La structure d'un algorithme

Nom : ..... **Classe :** .....

AVANT DE COMMENCER

Ce que tu vas apprendre

Avant d'écrire un programme, il faut savoir ce qu'est un algorithme et comment un ordinateur garde une information en mémoire. Ce chapitre pose le vocabulaire de tout le cours : la variable, son type, sa déclaration et l'affectation.

1. **Définir un algorithme et citer les quatre familles d'instructions.**
2. **Expliquer ce qu'est une variable et choisir un identifiant valide.**
3. **Choisir le bon type : entier, numerique, chaine de caracteres ou booleen.**
4. **Déclarer une variable et lui affecter une valeur avec la flèche `<-`.**
5. **Écrire la structure complète d'un algorithme : déclarations, `DEBUT`, instructions, `FIN`.**
6. **Suivre un algorithme pas à pas avec un tableau de trace.**

PARTIE 1

Qu'est-ce qu'un algorithme ?

DÉFINITION

Un algorithme

Une **suite d'instructions ordonnées** qui, une fois exécutée correctement, conduit à un résultat donné. Si l'algorithme est juste, on obtient le résultat voulu ; s'il est faux, le résultat est inattendu.

POUR COMPRENDRE

Tu as déjà suivi une recette de cuisine, un mode d'emploi ou un itinéraire : ce sont des algorithmes. Ils ne marchent que si chaque étape est **compréhensible** par celui qui l'exécute. Un ordinateur ne devine rien : il fait exactement ce qu'on lui dit, dans l'ordre.

En programmation, les instructions s'écrivent dans un **langage de programmation** (C#, Python, Java...). Dans ce cours, on utilise d'abord un langage d'apprentissage, le **pseudocode** : des mots-clés en français, simples et précis. Dans VS Code, l'extension **PseudoCodeDC** traduit ce pseudocode en C# et l'exécute avec la touche **F5**.

Les quatre familles d'instructions

L'affectation
Ranger une valeur dans une variable.

La lecture et l'écriture
Recevoir une information du clavier, afficher un message à l'écran.

Les tests
Choisir entre plusieurs chemins selon une condition.

Les boucles
Répéter des instructions tant que c'est nécessaire.

À RETENIR

Tout programme se décompose en quatre familles

- Un ordinateur ne comprend que quatre familles d'instructions : affectation, lecture / écriture, tests et boucles.
- Un jeu vidéo ou une application, ce sont des milliers de ces instructions, combinées.

Pourquoi l'algorithmie avant la programmation ?

L'algorithmie, c'est trouver **les étapes** qui résolvent un problème, sans se soucier des détails d'un langage. On apprend d'abord à raisonner ; la syntaxe du C# vient ensuite (chapitre 6).

Deux qualités à travailler

De l'intuition

Trouver quelles instructions mènent au résultat. Ça s'acquiert : c'est de l'expérience répétée.

De la rigueur

Après avoir écrit, **se mettre à la place de la machine** et dérouler l'algorithme pas à pas, avec un papier et un crayon.

PARTIE 2

Les variables

DÉFINITION

Une variable

Un **emplacement de la mémoire** réservé pour garder une valeur et la réutiliser plus tard. Elle a un **nom** (l'identifiant) et un **type**.

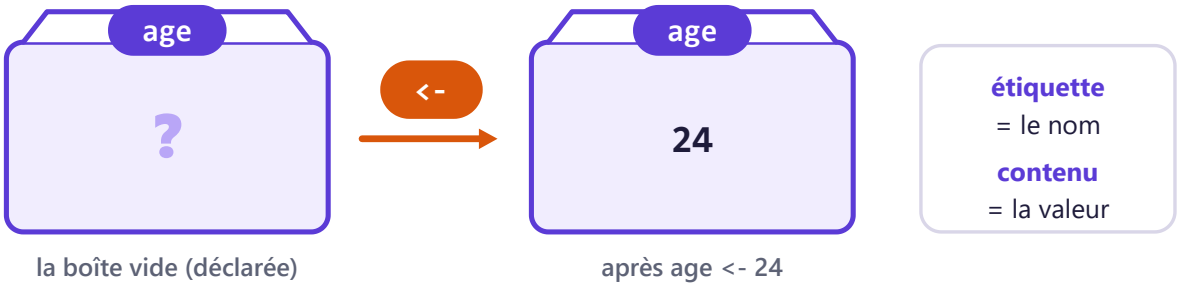


Figure 1.1 — Une variable est une boîte étiquetée : l'étiquette est son nom, le contenu est sa valeur.

Le programme utilise l'étiquette pour retrouver la boîte. Pour lire ce qu'il y a dedans, on la désigne par son nom ; pour y ranger une nouvelle valeur, aussi. Une boîte ne contient **qu'une seule valeur à la fois** : la nouvelle remplace l'ancienne.

L'identifiant (le nom de la variable)

- Il peut contenir des **lettres** minuscules et majuscules (a-z, A-Z), sans accent.
- Il peut contenir des **chiffres** (0-9) et le **tiret bas** `_`.
- Il ne peut **pas commencer par un chiffre** : `abc123` est valide, `123abc` ne l'est pas.
- Pas d'espace ni de tiret : `total score` et `total-score` sont refusés.

Identifiant	Valide ?	Pourquoi
`age`	oui	des lettres seulement
`totalScore`	oui	lettres ; une majuscule pour séparer les mots
`total_score`	oui	le tiret bas est permis
`note2`	oui	un chiffre, mais pas au début
`2note`	non	commence par un chiffre
`prix-ttc`	non	le tiret « - » est interdit
`mon age`	non	un espace est interdit

ATTENTION

Dans VS Code, n'appelle jamais une variable `de` ni `en` : ces petits mots font partie de la déclaration `chaine de caracteres` et la traduction se casse. Pour un dé, écris `jet` ou `de6`.

À RETENIR

Un bon nom de variable

- Il dit ce que la variable contient : `prixTotal` plutôt que `x`.
- Il est court et sans accent : `annee`, pas `année`.

PARTIE 3

Les types de variables

Le **type** dit à l'ordinateur quelle sorte de valeur la boîte va contenir. Il décide de la place réservée en mémoire et des opérations permises : on peut multiplier deux nombres, pas deux prénoms.

Type	Ce qu'il contient	Exemples de valeurs
`entier`	un nombre **sans virgule**	`24`, `0`, `-5`, `2026`
`numerique`	un nombre **entier ou à virgule**	`1.68`, `13.5`, `-0.25`
`chaine de caracteres`	du **texte** : lettres, chiffres, espaces, ponctuation	`"Lina"`, `"4TT"`, `"rue du Pont 12"`
`booleen`	**vrai ou faux**, et rien d'autre	`true`, `false`

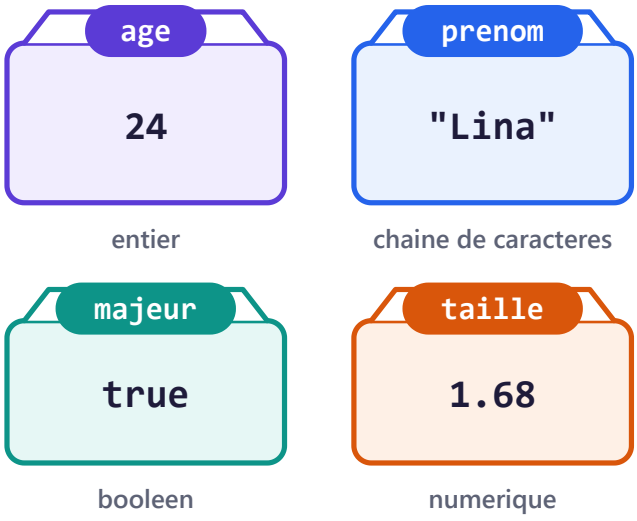


Figure 1.2 — Quatre variables, quatre types : la valeur doit correspondre au type de la boîte.

Les nombres

En pseudocode, la virgule s'écrit avec un **point** : `1.68`. On choisit `entier` dès que la virgule est inutile (un âge, une quantité) et `numerique` seulement quand elle compte (une taille, un prix, une moyenne).

Le texte

Une chaîne de caractères s'écrit **toujours entre guillemets** droits : `"Bonjour"`. Un numéro de téléphone ou un code postal est du texte : on ne calcule pas avec.

Le booléen

Il répond à une question par oui ou non : `majeur`, `permis`, `connecte`. Il vaut `true` (vrai) ou `false` (faux).

EXEMPLE

Pour une fiche d'inscription : `nom` en chaine de caracteres, `age` en entier, `taille` en numerique, `delegue` en booleen.

PARTIE 4

Déclarer et affecter

La déclaration

Avant de s'en servir, on **déclare** chaque variable : on annonce son nom et son type. En pseudocode, toutes les déclarations se placent **avant** `DEBUT`.

PSEUDOCODE Déclarer une ou plusieurs variables

```
variable age en entier
variable prenom en chaine de caracteres
variables largeur, longueur en entier
```

Le mot-clé s'écrit `variable` pour une seule, `variables` pour plusieurs du même type, séparées par des virgules. Les types s'écrivent en minuscules et **sans accent**.

L'affectation

DÉFINITION

L'affectation

Ranger une valeur dans une variable. Elle se note avec la flèche `<-` : la valeur de droite va dans la variable de gauche.

PSEUDOCODE

```
age <- 24
prenom <- "Nicolas"
majeur <- true
```

La valeur doit correspondre au type déclaré : ranger `"Nicolas"` dans une variable `entier` provoque une erreur. Une variable non déclarée en provoque une aussi.

Affecter une expression ou une autre variable

À droite de la flèche, on peut écrire une valeur, une autre variable ou un **calcul** avec `+`, `-`, `*` (multiplier) et `/` (diviser).

PSEUDOCODE

```
1  somme <- 12
2  age <- somme
3  age <- somme + 12
```

Ligne 2 : `age` reçoit une **copie** de la valeur de `somme` (12). Ligne 3 : `age` vaut 24, mais `somme` vaut toujours 12. Une affectation ne modifie **que la variable à gauche** de la flèche.

ATTENTION

Le piège des guillemets

Les guillemets changent tout. `prenom <- "Nicolas"` range le texte Nicolas. `prenom <- Nicolas` cherche une **variable** nommée Nicolas et provoque une erreur si elle n'existe pas.

PARTIE 5

La structure d'un algorithme

Un algorithme a toujours la même forme : les déclarations, puis les instructions encadrées par `DEBUT` et `FIN`. Ces deux mots-clés marquent le début et la fin du programme.

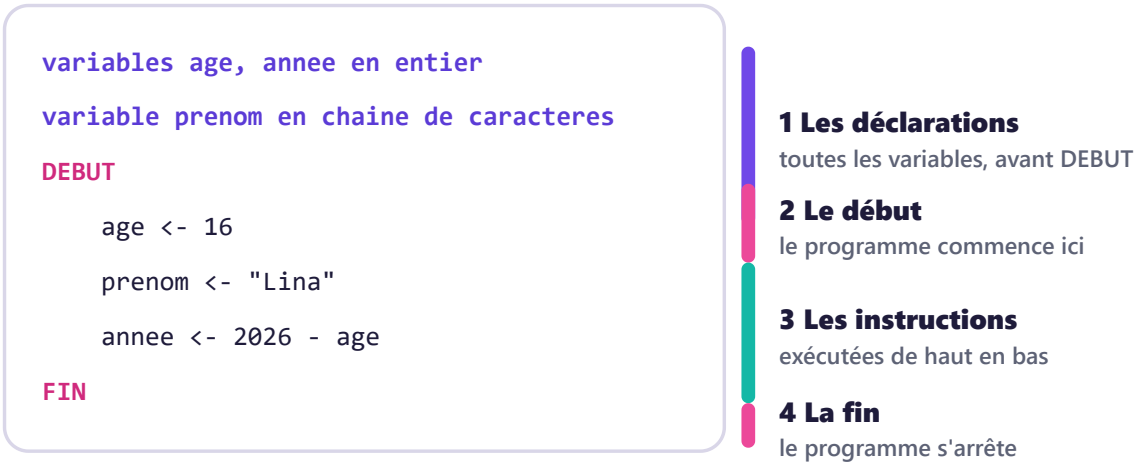


Figure 1.3 — Les quatre zones d'un algorithme.

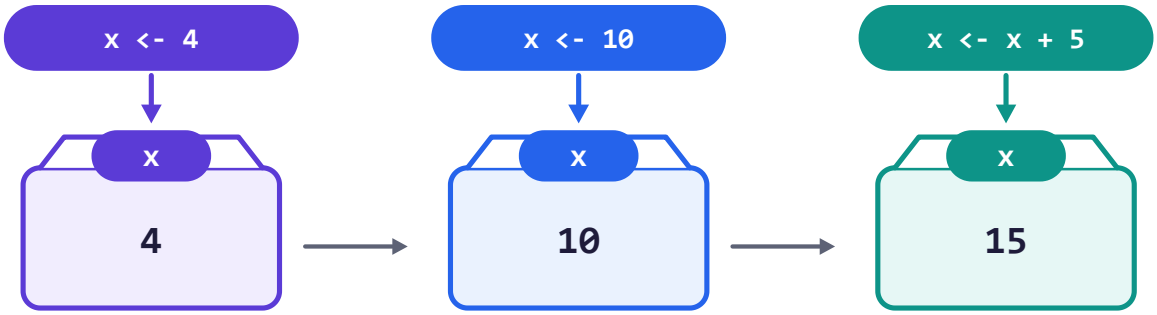
PSEUDOCODE **Un algorithme complet**

```
variables age, annee en entier
variable prenom en chaine de caracteres
DEBUT
    age <- 16
    prenom <- "Lina"
    annee <- 2026 - age
FIN
```

On **décale** (indente) les instructions entre `DEBUT` et `FIN` : l'ordinateur s'en moque, mais le lecteur voit tout de suite la structure.

L'ordre des instructions

Les instructions s'exécutent **de haut en bas**, une à la fois. Si deux affectations visent la même variable, la seconde **remplace** la première.



Les instructions s'exécutent de haut en bas : la dernière affectation remplace la précédente.

Figure 1.4 — x vaut d'abord 4, puis 10, puis 15 : seule la dernière valeur reste.

Le tableau de trace

Pour vérifier un algorithme, on se met à la place de la machine : une ligne par instruction, une colonne par variable, et on note la valeur de chaque variable **après** l'instruction.

PSEUDOCODE

```
variables a, b en entier
DEBUT
    a <- 5
    b <- a * 2
    a <- b - 1
FIN
```

Instruction	a	b
`a <- 5`	5	?
`b <- a * 2`	5	10
`a <- b - 1`	9	10

À RETENIR

La méthode

- Une valeur pas encore affectée se note « ? ».
- On recopie les valeurs qui ne changent pas : la trace montre l'état complet de la mémoire.

BILAN DU CHAPITRE 1

L'essentiel à retenir

1. Un **algorithme** est une suite d'instructions ordonnées qui mène à un résultat. On l'écrit en **pseudocode** avant de le traduire dans un langage comme le C#.

2. Quatre familles d'instructions suffisent : **affectation**, **lecture / écriture**, **tests** et **boucles**.

3. Une **variable** est une boîte étiquetée en mémoire : un **nom** (identifiant), un **type** et une seule valeur à la fois.

4. Un identifiant contient des lettres sans accent, des chiffres et `_`, sans espace, et ne commence pas par un chiffre.

5. Les quatre types : `entier`, `numerique`, `chaine de caracteres` (entre guillemets) et `booleen` (`true` ou `false`).

6. On **déclare** avant `DEBUT` (`variable age en entier`), puis on **affecte** avec `<-` : seule la variable de gauche change.

7. Les instructions s'exécutent **de haut en bas** ; un **tableau de trace** permet de les suivre pas à pas.

Le lexique

Algorithme
Suite d'instructions ordonnées qui mène à un résultat.

Pseudocode
Langage d'apprentissage en français, traduit en C# par l'extension de VS Code.

Variable
Emplacement de la mémoire qui garde une valeur, avec un nom et un type.

Identifiant
Le nom d'une variable.

Type
La sorte de valeur qu'une variable peut contenir (entier, numerique, texte, booléen).

Déclaration
Annoncer le nom et le type d'une variable, avant `DEBUT`.

Affectation
Ranger une valeur dans une variable avec la flèche `<-`.

Booléen
Type qui ne vaut que `true` (vrai) ou `false` (faux).

Chaîne de caractères
Du texte, toujours écrit entre guillemets.

Tableau de trace
Tableau qui suit la valeur des variables instruction après instruction.

Je fais le point

Coche une case par ligne : où en es-tu avant l'interro ?

Je suis capable de…	Oui	Presque	Pas encore
Définir un algorithme et citer les quatre familles d'instructions.	☐	☐	☐
Dire si un identifiant est valide et expliquer pourquoi.	☐	☐	☐
Choisir le type qui convient à une information.	☐	☐	☐
Déclarer des variables et leur affecter une valeur.	☐	☐	☐
Écrire un algorithme complet avec `DEBUT` et `FIN`.	☐	☐	☐
Remplir le tableau de trace d'un algorithme.	☐	☐	☐

CHAPITRE 1 · EXERCICES

Teste tes connaissances

avant l'interro

COMMENT T'Y PRENDRE

1. **Relis la synthèse** : l'essentiel, le lexique et « Je fais le point ».
2. **Fais les exercices sans le cours**, comme à l'interro.
3. **Corrige-toi** avec la correction vue en classe ou sur le site.
4. **Reviens sur tes erreurs** : relis la partie du cours concernée.

LES 7 EXERCICES

1. **Le vocabulaire de base**
2. **Identifiant valide ou non ?**
3. **Quel type pour quelle valeur ?**
4. **Déclaration et affectation**
5. **Le tableau de trace**
6. **Trouve les erreurs**
7. **Mon premier algorithme**

Exercice 1 Le vocabulaire de base

▸ **Coche la bonne réponse.**

1. Un algorithme, c'est...

- [ ] un langage de programmation
- [ ] une suite d'instructions ordonnées
- [ ] un composant de l'ordinateur

2. Laquelle n'est PAS une famille d'instructions ?

- [ ] les tests
- [ ] les boucles
- [ ] les dessins
- [ ] l'affectation

3. Quel type choisir pour le nombre d'élèves d'une classe ?

- [ ] entier
- [ ] numerique
- [ ] booleen
- [ ] chaine de caracteres

4. Quel type choisir pour un numéro de téléphone ?

- [ ] entier
- [ ] numerique
- [ ] chaine de caracteres
- [ ] booleen

5. Que vaut une variable booleen ?

- [ ] un nombre
- [ ] true ou false
- [ ] un texte entre guillemets

Exercice 2 Identifiant valide ou non ?

▸ **Classe chaque identifiant dans la bonne colonne.**

`prenom` `nbEleves` `note_finale` `jour7` `7jours` `mon-age` `prix total` `année`

Valide	Non valide

Exercice 3 Quel type pour quelle valeur ?

- **Relie chaque valeur au type de variable qui peut la contenir (écris la lettre).**

1	`17` →	**A**	numerique
2	`17.5` →	**B**	entier
3	`"17"` →	**C**	booleen
4	`false` →	**D**	chaine de caracteres

Exercice 4 Déclaration et affectation

- **Coche vrai ou faux. Si c'est faux, corrige la phrase sur la ligne.**

Affirmation	Vrai	Faux
Les déclarations se placent avant `DEBUT`.	☐	☐
`age <- 15` range la valeur 15 dans la variable age.	☐	☐
Après `a <- b`, la variable b est vide.	☐	☐
`ville <- Waremme` range le texte Waremme dans ville.	☐	☐
Une variable peut contenir deux valeurs en même temps.	☐	☐

Exercice 5 Le tableau de trace

▸ **Suis l'algorithme pas à pas : écris la valeur de x et de y après chaque instruction.**

PSEUDOCODE

```
variables x, y en entier
DEBUT
    x <- 3
    y <- x + 4
    x <- y * 2
    y <- y - x
FIN
```

Instruction	x	y
`x <- 3`		
`y <- x + 4`		
`x <- y * 2`		
`y <- y - x`		

Exercice 6 Trouve les erreurs

- **Cet algorithme contient trois erreurs. Explique chacune en une phrase.**

PSEUDOCODE

```
1  variable 2emeNote en entier
2  variable classe en chaine de caracteres
3  variable reussi en booleen
4  DEBUT
5      classe <- 4TT
6      reussi <- "oui"
7  FIN
```

1. Ligne 1 : quelle est l'erreur ?

2. Ligne 5 : quelle est l'erreur ?

3. Ligne 6 : quelle est l'erreur ?

Exercice 7 Mon premier algorithme

- **Écris l'algorithme complet : déclarations, DEBUT, instructions, FIN.**

Écris un algorithme qui déclare trois variables : le **nom** d'un jeu vidéo, son **année** de sortie et son **prix** en euros. Range ensuite dans chacune une valeur de ton choix.