



# INFORMATIQUE

## Sequence 1 : Premiers pas

TD 1.1

v2025-09-05

IUT d'Annecy, 9 rue de l'Arc en Ciel, 74940 Annecy

## HELLO WORLD

### Sommaire

|  |          |
|--|----------|
| <b>1 Niveau 1</b>                              | <b>2</b> |
| 1.1 Variables . . . . .                        | 2        |
| 1.2 Conditions . . . . .                       | 3        |
| <b>2 Niveau 2</b>                              | <b>4</b> |
| 2.1 Opérations et conversions . . . . .        | 4        |
| 2.2 Conditions multiples et booléens . . . . . | 4        |
| <b>3 Niveau 3</b>                              | <b>5</b> |
| 3.1 Traces, erreurs et structures . . . . .    | 5        |
| 3.2 Imbrications et logique avancée . . . . .  | 6        |



#### La division en langage C

En langage C, l'opérateur division ne se comporte pas de la même façon selon les types de données.

**Si les deux variables sont des entiers :** La division est une division euclidienne (division entière). Le résultat de l'opération est donc un ENTIER qui ne contient que le quotient de la division.

- Exemple :  $5/2$  retournera la valeur 2

**Si une des deux variables est flottante (float ou double) :** La division est une division à virgule et le résultat sera donc un nombre à virgule.

- Exemple :  $5/2.0$  retournera la valeur 2.5



#### L'opérateur modulo % – Le reste de la division euclidienne

L'opérateur modulo % retourne le reste de la division euclidienne.

Par exemple :  $5\%2$  donnera 1, car  $5 = 2*2 + 1$ .

# 1 Niveau 1

## 1.1 Variables

### Exercice 1 : Informations personnelles (partie 1)

On souhaite stocker des informations simples concernant un étudiant :

- Son prénom
- Son âge
- Sa taille (en centimètres)

**Question 1** Pour chaque variable, proposer un type de données adapté et justifier le choix.

**Question 2** Écrire les lignes de **déclaration** de ces variables (sans initialisation).

**Question 3** Écrire des lignes **d'initialisation** cohérentes pour un étudiant fictif.

### Exercice 2 : Affectations successives

On considère le programme suivant :

```
int x = 2;
int y = 5;
int z;
z = x + y;
5 y = z - 1;
x = y + z + x;
```

**Question 1** Donner les valeurs finales de x, y et z.

### Exercice 3 : Types adaptés

On considère les informations suivantes :

- Nombre d'étudiants d'une promo
- Température extérieure
- Numéro de salle
- Prix d'un sandwich

**Question 1** Pour chacune, donner le type le plus approprié et la déclaration de la variable.

### Exercice 4 : Prédire l'affichage (entier vs réel)

On exécute le programme suivant.

```
int a = 7, b = 2;
float x = 7, y = 2;
printf("A:%d\n", a / b);
printf("B:%f\n", x / y);
5 printf("C:%f\n", a / (float)b);
printf("D:%d\n", a % b);
```

**Question 1** Si ce n'est pas déjà fait, lire l'encart sur la division, page 1

**Question 2** Prédire l'affichage généré par les lignes ci-dessus.

## 1.2 Conditions

### Exercice 5 : Majorité (partie 1)

On veut afficher « Majeur » si `age`  $\geq$  18, sinon « Mineur ».

**Question 1** Décrire la logique en français ou pseudo-code avec un `if/else`. Donner deux exemples d'entrées et la sortie attendue.

### Exercice 6 : Qui est le plus grand ?

On souhaite écrire un programme qui affiche lequel de deux nombres `a` et `b` est le plus grand. S'ils sont égaux, le programme le dira.

**Question 1** Proposer les lignes de codes qui, pour deux entiers `a` et `b`, affichent le plus grand ou s'ils sont égaux.

### Exercice 7 : Nombre mystère (logique)

On considère un programme dont deux variables ont été déclarées : `nombre_mystere` et `nombre_joueur`. Le nombre mystère n'est pas connu du joueur. Il doit le deviner en tentant des valeurs au hasard.

```
int nombre_mystere = 7;  
int nombre_joueur = get_int("Deviner le nombre mystere : ");
```

On veut écrire **bravo** si l'utilisateur a trouvé le nombre, **Raté** sinon.

**Question 1** Ecrire un pseudo-code décrivant la logique à implémenter

**Question 2** Traduire ces lignes en langage C

**Question 3** Réécrire le programme pour afficher si le nombre proposé est **Trop petit** et **Trop grand**.

## 2 Niveau 2

### 2.1 Opérations et conversions

#### Exercice 8 : Opérations et priorités

Soit le code suivant :

```
int r = 10 - 2 * 3 + 8 / 2;  
int s = r + 5 * (3 - 1);
```

**Question 1** Donner l'ordre exact des opérations.

**Question 2** Donner les valeurs finales de **r** et **s**

#### Exercice 9 : Conversions et cast

On considère :

```
int a = 13, b = 5;  
float u = 13, v = 5;  
int A = a/b;  
float C = (float)a / b;  
5 int D = (float)a / b;  
int E = a/b;
```

**Question 1** Donner les valeurs des variables A, B, C, D et E.



#### Le transtypage (cast)

En langage C, on peut momentanément considérer qu'une variable est d'un certain type au lieu de celui déclaré. Il suffit de spécifier le type entre parenthèses devant la variable.

Par exemple :

```
int variable_entiere;  
float resultat = (float) variable_entiere/2;
```

Durant cette opération, `variable_entiere` sera considérée comme `float`.

#### Exercice 10 : Prédire l'affichage (mélange de types)

```
int n = 9, d = 4;  
float p = 9, q = 4;  
n = n - d/2;  
p = p / q;  
5 printf("A:%d\n", n);  
printf("B:%f\n", p);  
printf("C:%f\n", (float)n / d);  
printf("D:%d\n", n % d);
```

**Question 1** Prédire chaque ligne affichée et expliquer le rôle de la **division entière** dans la mise à jour de **n**.

### 2.2 Conditions multiples et booléens

#### Exercice 11 : Intervalle inclusif et exclusif

**Question 1** Ecrire une condition en langage C pour tester  $5 \leq x \leq 10$ .

**Question 2** Ecrire ce même test d'une façon différente.

**Question 3** Expliquer pourquoi  $(x > 5 \ || \ x < 10)$  est incorrecte.

**Question 4** Expliquer pourquoi  $(5 \leq x \leq 10)$  est incorrecte.

**Exercice 12 : Autorisation d'accès**

On considère trois variables booléennes (le type **bool** est défini dans la bibliothèque **stdbool.h**) :

```
bool isRegistered;  
bool hasBadge;  
bool isAdmin;
```

On veut accorder l'accès à une personne si elle possède un badge et est enregistrée. L'accès sera aussi accordé si la personne est administratrice.

**Question 1** Donner une condition booléenne correctement parenthésée qui accorde, ou non, l'accès.

**Exercice 13 : Calcul d'expressions booléennes**

Soient  $a=3$ ,  $b=7$ ,  $c=7$ .

**Question 1** Évaluer les expressions booléennes suivante (pour chacune, dire si elle renvoie **vrai** ou **faux**) :

- |                               |                                |                              |
|-------------------------------|--------------------------------|------------------------------|
| • $(a < b) \ \&\& \ (b == c)$ | • $!(a < b) \ \&\& \ (b == c)$ | • $(a < b) \    \ (b == c)$  |
| • $(a >= b) \    \ (c != 7)$  | • $(a < b) \ \&\& \ !(b == c)$ | • $!(a < b) \    \ (b == c)$ |

### 3 Niveau 3

**Les nombres flottants en C**

En langage C, il existe deux types principaux pour représenter des nombres à virgule :

**float** : type virgule flottante simple précision (32 bits). Il permet de représenter environ 7 chiffres significatifs avec une plage de valeurs allant de  $10^{-38}$  à  $10^{38}$ . C'est rapide et économique en mémoire, mais moins précis.

**double** : type virgule flottante double précision (64 bits). Il permet de représenter environ 15 à 16 chiffres significatifs avec une plage beaucoup plus large ( $10^{-308}$  à  $10^{308}$ ). Il est plus précis mais consomme deux fois plus de mémoire.

**Attention** : les nombres flottants ne sont qu'une approximation des réels. Certaines opérations ne donnent pas toujours le résultat attendu exactement, à cause des limites de la représentation binaire.

- Exemple avec **float** : `1.0f/3` donnera une valeur approchée de 0.333333.
- Exemple avec **double** : `1.0/3` donnera une valeur plus précise, par exemple 0.3333333333333333.
- Exemple de limite : la somme de `0.1+0.1+0.1` peut afficher 0.30000000000000004.

#### 3.1 Traces, erreurs et structures

**Exercice 14 : Exécution pas à pas**

Analyser l'exécution suivante pas à pas :

```
int a = 5, b = 10, c = 0;  
c = a + b; // 1  
b = c - a; // 2  
a = a + 1; // 3  
5 | c = a + b + c; // 4
```

**Question 1** Donner les valeurs finales de **a**, **b** et **c**.

**Exercice 15 : Précision et débordements**

**Question 1** Expliquer ce qui peut arriver pour `int x = 2000000000 + 2000000000;`. Donner un exemple d'impact réel.

**Question 2** Comparer `float` vs `double` pour accumuler des montants (somme de 1/10 répété 100 fois).

**Exercice 16 : Prédire l'affichage (mélange et cast)**

```
int i = 7, j = 3;
float u = 7, v = 3;
i = i + j/2;      // division entière
u = u / v;        // division réelle
5 printf("A:%d\n", i);
printf("B:%f\n", u);
printf("C:%f\n", (float)i / j);
printf("D:%d\n", (i * j) % 5);
```

**Question 1** Prédire chaque ligne d'affichage et expliquer l'effet du cast et de la division entière sur `i`.

### 3.2 Imbrications et logique avancée

**Exercice 17 : Écrire ses propres conditions**

On dispose de drapeaux `badgeOK`, `tenueOK`, `formationOK`, `isResponsable`.

Règles d'accès :

- Les responsables sont autorisés quelque soit leur tenue, l'état de leur badge ou de leur formation.
- Les autres seront autorisés s'ils sont en possession d'un badge et soit qu'ils ont passé une formation, soit qu'ils possèdent une tenue de sécurité

**Question 1** Écrire l'expression booléenne correspondante.

**Exercice 18 : Majorité avancée (partie 3)**

Une billetterie donne un tarif réduit si l'âge de la personne est un multiple de 10. Le tarif sera majoré pour les plus de 60 ans.

**Question 1** Proposer une structure `if/else if/else` lisible.

**Exercice 19 : Court-circuit et sécurité**

Analyser les deux versions suivantes :

```
if ( i / j > 2 && j != 0 ) {
    printf("OK\n");
}
```

```
if ( (j != 0) && (i / j > 2) ) {
    printf("OK\n");
}
```

**Question 1** Quelle différence y a-t-il entre ces deux versions ?

**Question 2** Laquelle vous semble la plus efficace et pourquoi ? Justifier votre réponse et donner un exemple.