Les structures alternatives (si alors)

Table des matières

1. Introduction

La plupart des algorithmes exécutent des actions différentes en fonction de l'état courant des variables. Prenons l'exemple d'un logiciel de comptabilité : lorsque une entrée est un débit le signe *« moins »* est affiché, et lorsque l'entrée est un crédit le signe *« plus »* est affiché. Le comportement à l'affichage est donc conditionné au type de l'opération, et c'est information que l'on peut imaginer trouver dans une variable. Il est donc nécessaire de pouvoir modifier le comportement du programme en fonction de la valeur d'une variable.

C'est là qu'interviennent les structures alternatives, qui sont présentent dans la plupart des langages de programmation, et qui permettent d'appliquer des concepts mathématiques et logiques simples. Ce module a pour objectif d'initier aux différents concepts que sont les comparaisons, les instructions conditionnelles et l'algèbre booléenne appliquées aux langages en général.

2. Comparer des éléments

Objectif

- Savoir utiliser les opérateurs de comparaison.

Mise en situation

L'une des opérations les plus réalisées dans des programmes est la comparaison de deux éléments. Il existe différents opérateurs de comparaison, comme l'égalité ou non, la supériorité ou l'infériorité. Chaque comparaison doit être faite entre 2 éléments du même type, et produit un résultat qui est binaire : vrai ou faux. C'est ce que l'on appelle un booléen. Nous allons voir ici comment réaliser les opérations de comparaisons de base.

Az Définition

Les comparaisons de base sont :

- l'égalité,
- la différence,
- la supériorité,
- l'infériorité,
- la supériorité stricte,
- l'infériorité stricte.

Résultat d'une comparaison

En programmation, on retrouve ces mêmes comparaisons pour des nombres, des caractères, des phrases, etc. Les symboles de comparaison diffèrent cependant de l'écriture mathématique naturelle. De plus, les notations peuvent varier d'un langage à un autre.

Une comparaison produit un résultat : vrai ou faux (**true** ou **false**), appelé un **booléen**.

Tester l'égalité

Syntaxe

Le symbole **==** signifie *« est égal à »*. Il est bien distinct du simple **=** utilisé pour affecter une valeur à une variable.

Exemple

```javascript
/** JavaScript: affiche true et false */
console.log(5 == 5)
console.log(5 == -5)
```

```python
"""Python: affiche True et False."""
print(5 == 5)
print(5 == -5)
```

L'inégalité

Syntaxe

L'opposé de **==** est **!=** qui signifie *« n'est pas égal à »*.

Exemple

```javascript
/** JavaScript: affiche true et false */
console.log('A' != 'B')
console.log('A' == 'B')
```

```python
"""Python: affiche True et False."""
print('A' != 'B')
print('A' == 'A')
```

Supériorité et infériorité strictes

Syntaxe

L'opérateur **<** signifie *« est strictement inférieur à »* et **>** signifie *« est strictement supérieur à »*.

Exemple

```javascript
/** JavaScript: affiche true et false */
console.log(1 < 2)
console.log(5 > 5)
```

```python
"""Python: affiche True et False."""
print(1 < 2)
print(5 > 2)
```

Supériorité et infériorité

Syntaxe

L'opérateur **<=** signifie *« est inférieur ou égal à »* et **>=** signifie *« est supérieur ou égal à »*.

Exemple

```javascript
/** JavaScript: affiche true et false */
console.log(1 <= 1)
console.log(4 >= 5)
```

```python
"""Python: affiche True et False."""
print(1 <= 1)
print(4 >= 5)
```

Utilisation

Les **opérandes** d'une comparaison, c'est-à-dire les valeurs sur lesquelles elle s'applique, peuvent être des valeurs simples mais également des variables et des expressions.

Exemple

```javascript
/** JavaScript : affiche true */
let amount = 6 * 6
console.log(amount == 9 * 2 * 2)
```

```python
"""Python : affiche True."""
amount = 6 * 6
print(amount == 9 * 2 * 2)
```

Attention

En JavaScript, on préfère la comparaison stricte (===) à la comparaison simple (==) présentée ici. La comparaison stricte vérifie l'égalité de type en plus de l'égalité de valeur.

À retenir

Les opérateurs de comparaisons permettent de comparer deux opérandes, qu'elles soient des variables ou des expressions.

3. Appliquer la notion

Question 1

[solution n°1 p. 28]

Quel résultat produit le programme suivant ?

```
1 console.log(1 == 1.0)
```

Question 2

[solution n°2 p. 28]

On dispose d'un programme incomplet et on veut qu'il affiche le résultat de la comparaison *« âge supérieur ou égal à 18 »*. Compléter l'instruction `console.log` pour cela.

```
1 let age = 21
2 console.log()
```

4. Comparer le type des éléments

Objectifs

- Comprendre la différence entre comparaison simple et comparaison stricte ;
- Savoir utiliser les comparaisons strictes.

Mise en situation

Lorsque l'on compare des éléments entre eux, le résultat dépend non seulement de leur valeur mais également de leur type.

Type Rappel

Le type d'un élément est en quelque sorte sa nature : il peut être un nombre (entier, décimal, etc.), un caractère, une phrase, etc.

Le type est en partie identifiable selon la notation utilisée :

- On met des apostrophes pour les caractères mais rien pour les nombres.
- Les nombres décimaux possèdent un point et pas les entiers.

Comparaison de type

Même si les opérateurs de comparaison sont parfois identiques dans les langages, la comparaison n'est pas toujours exactement la même.

En Python, les éléments doivent avoir la même **valeur** et être de même **type** pour être égaux, alors qu'en JavaScript il existe des opérateurs spéciaux permettant d'indiquer que la comparaison est **stricte**, c'est à dire qu'elle doit prendre en compte le type des opérandes.

Exemple

La deuxième ligne montre la différence entre le JavaScript et le Python. Pour le JavaScript, seule la valeur est importante : que l'opérande soit une chaîne de caractères ou un nombre ne fait pas de différence. En revanche pour le Python, même si les valeurs sont *équivalentes*, un nombre et une chaîne ne sont pas égaux.

```javascript
/** JavaScript : affiche true et true */
console.log(91 == 91)
console.log('91' == 91)
```

```python
"""Python : affiche true et false."""
print(91 == 91)
print('91' == 91)
```

Égalité stricte

Méthode

Pour JavaScript, qui différencie l'**égalité** de l'**égalité stricte**, l'opérateur d'égalité stricte est **===** et signifie *« est égal et de même type que »*.

```
1 /** JavaScript: affiche true et false */
2 console.log('7' == 7)
3 console.log('7' === 7)
```

Inégalité stricte

Méthode

De même, l'inégalité stricte existe et se note **!==**, autrement dit *« est de type et/ou de valeur différent de »*.

```
1 /** JavaScript: affiche false et true */
2 console.log('7' != 7)
3 console.log('7' !== 7)
```

Attention

Les comparaisons strictes sont à préférer dans un code en JavaScript : leur utilisation fait partie des bonnes pratiques.

À retenir

Les langages ne gèrent pas les comparaisons de la même manière et il faut faire attention au type des opérandes.

5. Appliquer la notion

Question 1

[solution n°3 p. 28]

Quel résultat produit le programme ?

```
1 console.log(78 !== '78')
```

Question 2

[solution n°4 p. 28]

Quelle propriété des opérandes est regardée par les opérateurs strictes au contraire des opérateurs classiques ?

6. Instruction conditionnelle

Objectif

- Savoir utiliser les instructions conditionnelles.

Mise en situation

Avant d'exécuter des instructions, il est parfois nécessaire de vérifier certaines conditions.

Instruction conditionnelle

Pour utiliser des conditions dans un programme, on utilise les **instructions conditionnelles**. Combinées aux opérateurs de comparaison, elles permettent de tester des conditions avant de continuer le programme.

Instruction if

Az Définition

L'instruction `if` permet d'exécuter une instruction seulement si une condition donnée est vraie. Elle requiert :

- Le mot clé `if`.
- La condition en question.
- Le bloc d'instructions à exécuter si la condition est vraie.

Algorithmiquement, c'est l'instruction `if` équivaut à : *« **Si** condition vraie alors faire ... »*.

Exemple

```javascript
/** JavaScript : affiche "Suzy est dans..." */
const name = 'Suzy'
if (name[0] > 'M') {
  console.log('Suzy est dans la seconde moitié de l\'alphabet')
}

```

```python
"""Python : affiche "Suzy est dans...."."""
name = 'Suzy'
if name[0] > 'M':
  print('Suzy est dans la seconde moitié de l\'alphabet')
```

À retenir

L'instruction `if`, suivie d'une condition s'évaluant en un booléen, permet à des instructions de n'être exécutées que si une condition est d'abord respectée.

7. Appliquer la notion

Question 1

[solution n°5 p. 28]

Quel résultat renvoie la condition de l'instruction `if` ? Le bloc d'instruction est-il exécuté ?

```
if (5 + 3 === 53) {
  console.log('Monty Python')
}
```

Question 2

[solution n°6 p. 28]

Par quel(s) opérateur(s) pourrait-on remplacer « *===* » pour avoir le résultat inverse dans l'exemple précédent, c'est à dire une comparaison vraie ?

8. Instructions conditionnelles alternatives

Objectif

- Savoir enrichir une instruction conditionnelle avec des alternatives.

Mise en situation

L'instruction `if` peut être complétée par autant d'alternatives que nécessaire dans le cas où la condition est fausse.

Si... alors si... sinon...

L'instruction `if` va souvent de pair avec le `else`. Ce dernier permet de ne pas faire se succéder des blocs `if` inutilement et d'offrir une alternative au premier bloc.

En effet, un bloc `else` n'est exécuté que si la condition du bloc `if` n'est pas remplie, tandis que les conditions des blocs `if` simples sont **systématiquement** évaluées.

Instruction else

Méthode

L'instruction `else` est le *« sinon »* de l'instruction conditionnelle. Si la première condition n'est pas remplie et que le premier bloc est par conséquent ignoré, il est possible d'exécuter un autre bloc d'instructions alternatif. Le `else` est ignoré si la première condition est remplie.

Cela équivaut à dire : *« Si condition vraie alors faire ... **Sinon** faire ... »*.

Exemple

```javascript
/** JavaScript : condition fausse, affiche 'Un Vodka-Martini' */
if (7 === '007') {
  console.log('...Bond, James Bond')
} else {
  console.log('Un Vodka-Martini')
}
```

```python
"""Python : condition fausse, affiche 'Un Vodka-Martini'."""
if 7 == '007':
  print('...Bond, James Bond')
else:
  print('Un Vodka-Martini')
```

Il n'est pas obligatoire de se limiter à une seule alternative : une instruction conditionnelle peut contenir plusieurs `else if` à la suite permettant d'enchaîner les conditions jusqu'à en remplir une. Dès qu'une condition est vraie, les suivantes sont ignorées.

Exemple

```javascript
/** JavaScript : affiche "Plutôt grand" */
const height = 180

if (height < 150) {
  console.log('Plutôt petit')
} else if (height < 180) {
  console.log('Plutôt moyen')
```

```
} else {
  console.log('Plutôt grand')
}
```

```
"""Python : affiche "Plutôt grand"."""
height = 180

if height < 150:
  print('Plutôt petit')
elif height < 180:
  print('Plutôt moyen')
else:
  print('Plutôt grand')
```

Remarque

Selon les langages, on peut retrouver plusieurs manières d'écrire l'instruction `else if`. En JavaScript, on utilise simplement `else` suivit d'un nouveau `if`, tandis qu'en Python la syntaxe est raccourcie en `elif` (contraction de `else` et `if`).

Imbrication

Il est possible d'écrire une structure conditionnelle à l'intérieur d'une autre structure conditionnelle, pour simplifier le code et éviter d'évaluer plusieurs fois la même condition. Cette imbrication peut être réalisée un nombre quelconque de fois, même s'il est préférable d'éviter une imbrication trop profonde.

Imbrication Exemple

```
const number = 42
if(number > 10) {
  if (number < 30) {
    console.log('Nombre entre 10 et 30')
  }
  else if (number < 50) {
    console.log('Nombre entre 30 et 50')
  }
  else {
    console.log('Nombre supérieur ou égal à 50')
  }
}
else {
  console.log('Nombre inférieur ou égal à 10')
}
```

L'imbrication des structures conditionnelles permet ici d'éviter de répéter plusieurs fois la condition `number > 10` : à l'intérieur du bloc, on a déjà la garantie que `number` est supérieur à 10.

Complément

En JavaScript, l'instruction `switch` est une autre syntaxe pour exprimer des alternatives.

À retenir

Un bloc `if` peut être prolongé par un bloc `else` pour exprimer une alternative par défaut, et il est même possible d'enchaîner plusieurs blocs `else if` pour exprimer plusieurs alternatives conditionnelles.

9. Appliquer la notion

Question 1

[solution n°7 p. 28]

Ajouter une **alternative** au programme suivant pour effectuer un affichage pour les nombres strictement inférieurs à 100 mais pas inférieurs à 10.

```
let number = 99

if (number < 10) {
  console.log(number, 'strictement inférieur à 10')
}
```

Indice :

Un second `if` ne suffit pas, car un nombre tel que 9 déclencherait deux affichages.

Question 2

[solution n°8 p. 29]

Ajouter une alternative par défaut affichant simplement le numéro.

Indice :

Cette alternative doit s'exécuter uniquement si les deux conditions précédentes sont fausses.

10. Algèbre booléenne

Objectif

- Apprendre à utiliser l'algèbre booléenne.

Mise en situation

L'algèbre de Boole permet d'utiliser des raisonnements logiques avec des opérations de comparaison.

Il est possible de combiner des valeurs de vérité (`true` et `false`) pour constituer des conditions plus complexes.

Trois fonctions logiques existent :

- la conjonction (**et**),
- la disjonction (**ou**),
- la négation (**non**),

Comme les comparaisons produisent des booléens, il est possible de combiner directement deux comparaisons d'éléments avec une fonction logique **et**, afin de ne valider la condition globale que si les deux comparaisons sont vraies **simultanément**.

Conjonction

Syntaxe

La conjonction permet de représenter le « **et** » logique : pour qu'elle soit vraie, tous les éléments qui la composent doivent aussi être vrais.

On utilise « ***&&*** » en JavaScript et « ***and*** » en Python.

En langue naturelle, on peut traduire simplement une conjonction par : `condition1 && condition2 => « condition1 vraie ET condition2 vraie »`

Exemple

```javascript
/** JavaScript: affiche true et false */
console.log(true && true)
console.log(false && true)
```

```python
"""Python : affiche True et False."""
print(True and True)
print(False and True)
```

Disjonction

Syntaxe

La disjonction permet de représenter le « **ou** » logique : pour qu'elle soit vraie, au moins un de ses éléments doit être vrai.

On utilise `||` en JavaScript et `or` en Python.

En langue naturelle, on peut traduire simplement une disjonction: `condition1 || condition2 => « condition1 vraie OU condition2 vraie »`

Exemple

```javascript
/** JavaScript: affiche true */
console.log(false || true || false)
```

```python
"""Python : affiche True."""
print(False or True or False)
```

Il ne faut pas confondre les ET et les OU

Attention

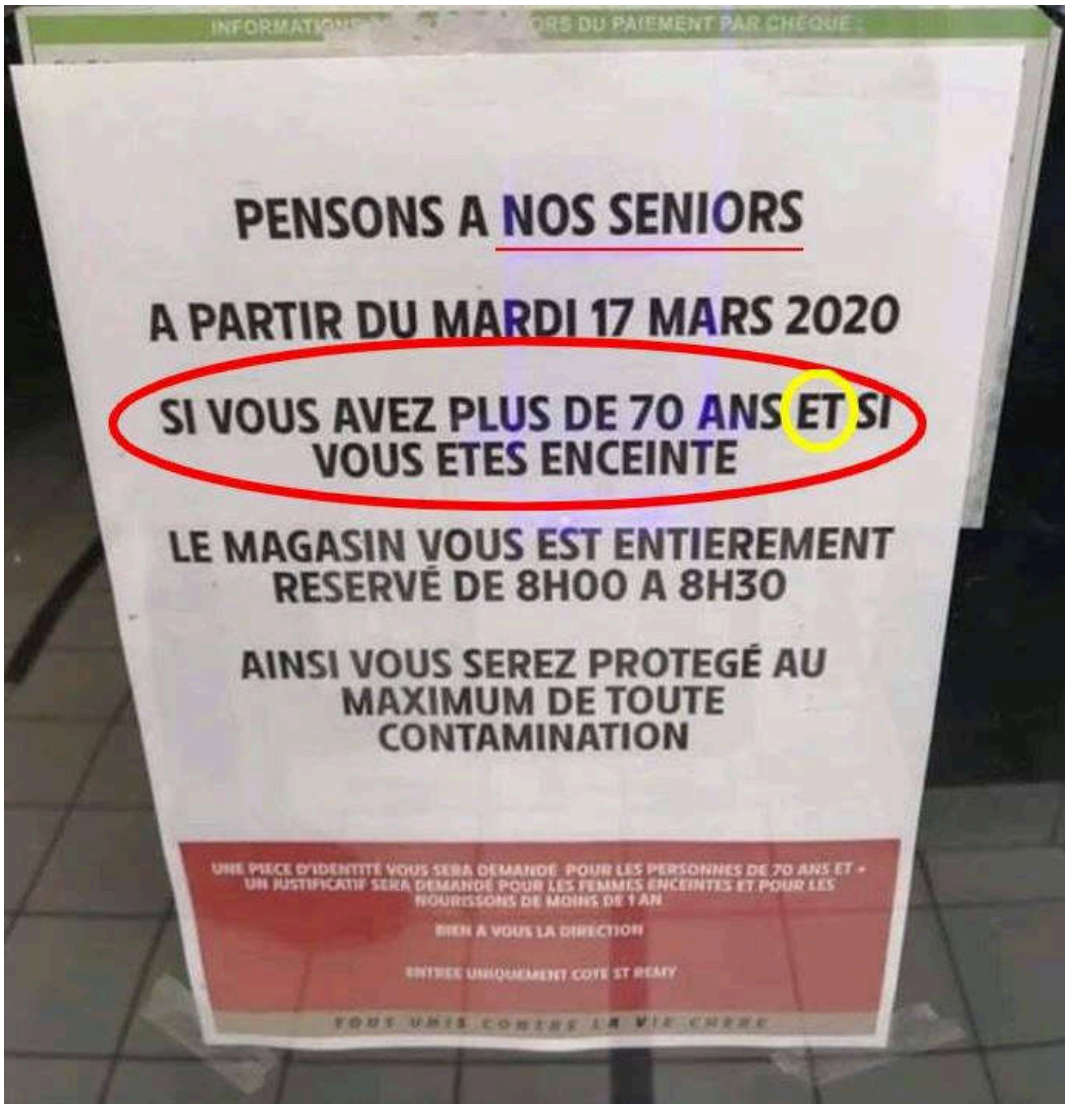

Photo prise sur la devanture d'un magasin en 2020 pendant la crise COVID-19

Négation

Syntaxe

La négation permet de représenter le « **non** » logique : la valeur de l'expression est simplement inversée de vraie à fausse ou inversement.

On utilise ***« ! »*** en JavaScript et « ***not*** » en Python.

En langue naturelle, on peut traduire simplement une négation : `!condition` => *« condition "non" vraie (fausse) »*

Exemple

```javascript
/** JavaScript: affiche true */
console.log(!false && true)
```

```python
"""Python : affiche True."""
print(not False and True)
```

Utilisation des parenthèses

Méthode

Pour s'assurer que l'expression produit bien le résultat attendu, il faut prêter attention à l'ordre des parenthèses : comme dans les expressions mathématiques classiques, elles ont des effets sur l'interprétation de l'expression. Elles servent aussi à faciliter leur lecture.

Exemple

```javascript
/** JavaScript: affiche false et true */
console.log(!true && false)
console.log(!(true && false))
```

```python
"""Python: affiche False et True."""
print(not True and False)
print(not(True and False))
```

À retenir

Les ***« et »*** et ***« ou »*** permettent de manipuler plusieurs conditions successives tandis que le ***« non »*** permet d'inverser la valeur de vérité.

11. Appliquer la notion

Question

[solution n°9 p. 29]

Compléter le code suivant pour afficher le résultat de la proposition logique suivante : « *age est compris entre 18 et 70* ».

```
1 const age = 15
```

Indice :

« *compris entre 18 et 70* » peut s'interpréter comme : « plus grand que 18 **et** plus petit que 70 ».

12. Algèbre booléenne et instructions conditionnelles

Objectif

- Utiliser les instructions conditionnelles avec l'algèbre booléenne.

Mise en situation

Les expressions logiques sont très utiles pour exprimer des conditions avec les structures conditionnelles.

Comme avec les expressions conditionnelles simples, les expressions utilisant des fonctions logiques telles que ***« et »***, ***« ou »*** et ***« non »*** permettent d'obtenir un résultat qui est **vrai** ou **faux**. Ces expressions sont donc utilisables comme conditions dans les instructions conditionnelles.

Elles sont également stockables dans des variables.

Exemple

```javascript
/** JavaScript: affiche "Plus de place à ce prix" */
let availablePlace = true
let budget = 40

if (availablePlace && budget > 120) {
  console.log('1ère classe')
} else if (!availablePlace || budget < 60) {
  console.log('Plus de place à ce prix')
}
```

```python
"""Python: affiche "Plus de place à ce prix"."""
availablePlace = True
budget = 40

if availablePlace and budget > 120:
  print('1ère classe')
elif not availablePlace or budget < 60:
  print('Plus de place à ce prix')
```

À retenir

Les opérateurs booléens permettent de construire des expressions avec plusieurs conditions en combinant leur résultat.

13. Appliquer la notion

Question 1

[solution n°10 p. 29]

Remplacer les « ... » par la condition suivante : *« température comprise entre 10 inclus et 30 exclus »*.

```
const temperature = 35

if (temperature < 0) {
  console.log(temperature, 'Gelées'
} else if (...) {
  console.log(temperature, 'Journée tempérée')
}

```

Question 2

[solution n°11 p. 29]

Ajouter l'état "Fortes chaleurs" quand la température est comprise entre 40° et 50° exclus et l'état "Inconnu" dans tous les autres cas.

Question 3

[solution n°12 p. 29]

Ajouter l'état "Température recherchée" quand la température est entre 5° et 9° inclus ou quand elle est de 21°.

14. Quiz

Quiz - Culture

[solution n°13 p. 30]

Exercice

Quelle est la différence entre l'opérateur d'égalité et l'opérateur d'égalité stricte en JavaScript ?

- A L'opérateur stricte n'accepte pas les nombres décimaux.
- B L'opérateur stricte permet de comparer le type en plus de la valeur.
- C L'opérateur d'égalité stricte n'accepte pas les opérandes qui ne sont pas des variables.

Exercice

En JavaScript, la valeur entière 2 est égale à la chaîne de caractères `'2'`.

- A Toujours
- B Jamais
- C Cela dépend de l'opérateur que l'on utilise

Quiz - Méthode

[solution n°14 p. 30]

Exercice

Quelles fonctions logiques sont utilisables en algèbre booléenne ?

- A ET
- B OU
- C NI
- D NOT

Exercice

Une instruction conditionnelle :

A est au minimum composée de `if` et `else`.

B est au minimum composée de `if` et `else if`.

C peut contenir un nombre illimité d'alternatives `else if`.

D peut être imbriquée dans une autre instruction conditionnelle.

E peut contenir plusieurs clauses `else`.

Quiz - Code

[solution n°15 p. 31]

Exercice

Quelle comparaison renvoie `true` en JavaScript ?

A '10.0' == 10

B '10.0' === 10

C '10' == 10

D 10.0 === 10

Exercice

Quelle(s) expression(s) tradui(sen)t en JavaScript la condition : *« angle droit et côté1 et côté2 égaux »* ?

A `angle = 90 && cote1 = cote2`

B `angle == 90 and cote1 == cote2`

C `angle === 90 && cote1 === cote2`

D `angle == 90 || cote1 == cote2`

E `angle == 90 && cote1 == cote2`

15. Défi final

On dispose du programme suivant :

```
const grade = 'A'

if (grade === 'A') {
  console.log('Mention exceptionnelle')
} else if (grade !== 'A' && grade !== 'B' && grade !== 'C' && grade !== 'D' && grade !== 'E' && grade !== 'F') {
  console.log('Pas de note')
}
```

Question 1

[solution n°16 p. 32]

Qu'affiche le programme si `grade` vaut 'A' ?

Question 2

[solution n°17 p. 32]

Qu'affiche le programme si `grade` vaut `'C'` ? Pourquoi ?

```
const grade = 'C'

if (grade === 'A') {
  console.log('Mention exceptionnelle')
} else if (grade !== 'A' && grade !== 'B' && grade !== 'C' && grade !== 'D' && grade !== 'E' && grade !== 'F') {
  console.log('Pas de note')
}
```

Question 3

[solution n°18 p. 33]

Ce code contient un test inutile, trouver lequel et le supprimer.

```
const grade = 'C'

if (grade === 'A') {
  console.log('Mention exceptionnelle')
} else if (grade !== 'A' && grade !== 'B' && grade !== 'C' && grade !== 'D' && grade !== 'E' && grade !== 'F') {
  console.log('Pas de note')
}
```

Indice :

La clause `else if` contient un test qui est redondant avec le `if`.

Question 4

[solution n°19 p. 33]

Améliorer le programme afin :

- d'ajouter les mentions *« Très bien »* en cas de **B** et *« Bien »* en cas de **C**.
- de mentionner *« Admis »* en cas de **D** ou de **E**.

Indice :

Penser à supprimer les tests inutiles dans le dernier `else if`.

Question 5

[solution n°20 p. 33]

On souhaite enfin ajouter l'information *« Non admis »* pour les notes **F**.

Ajouter ce cas et transformer le dernier `else if` en conséquence.

16. Conclusion

Les instructions conditionnelles sont essentielles et sont utilisées dans tous les programmes. Elles se retrouvent à l'identique dans tout les principaux langages de programmation, moyennant quelques adaptations de syntaxe. Ces structures permettent de comparer différentes variables et d'exécuter du code en fonction du résultat de ces comparaisons. Grâce à l'algèbre de Boole, il est possible de créer des conditions complexes, et donc de maîtriser finement la logique de son algorithme.

Solutions des exercices

Solution n°1

[exercice p. 7]

Le programme affiche `true`. Même si la représentation initiale est différente, l'entier `1` représente la même valeur que le flottant `1.0`.

Solution n°2

[exercice p. 7]

```
let age = 21
console.log(age >= 18)
```

Solution n°3

[exercice p. 10]

Le programme affiche `true` : en effet, les deux valeurs sont équivalentes, mais ne sont pas du même type. L'opérateur de comparaison stricte détermine donc que ces deux valeurs sont différentes.

Solution n°4

[exercice p. 10]

Le type des opérandes est également comparé par les opérateurs strictes.

Solution n°5

[exercice p. 12]

La condition vaut *false* et le bloc est ignoré. Une addition de deux chiffres ne permet pas de les concaténer.

Solution n°6

[exercice p. 12]

<, **<=** ou **!=** .

Solution n°7

[exercice p. 16]

```
let number = 99

if (number < 10) {
  console.log(number, 'strictement inférieur à 10')
} else if (number < 100) {
  console.log(number, 'strictement inférieur à 100')
}
```

On utilise la structure `else if`, qui permet d'effectuer le second affichage **seulement si** la première condition est fausse **et** la deuxième condition est vraie.

Solution n°8

[exercice p. 16]

```
let number = 99

if (number < 10) {
  console.log(number, 'strictement inférieur à 10')
} else if (number < 100) {
  console.log(number, 'strictement inférieur à 100')
} else {
  console.log(number)
}
```

Solution n°9

[exercice p. 20]

```
const res = (age > 18) && (age < 70)
console.log(res)
```

On utilise le ***« ET »*** logique pour exprimer la combinaison des deux conditions.

Les parenthèses ne sont pas nécessaires, car les opérateurs booléens sont moins prioritaires que les opérateurs de comparaison, mais permettent d'améliorer la lisibilité.

Solution n°10

[exercice p. 22]

```
const temperature = 35

if (temperature < 0) {
  console.log(temperature, 'Gelées')
} else if (temperature >= 10 && temperature < 30) {
  console.log(temperature, 'Journée tempérée')
}

```

Solution n°11

[exercice p. 22]

```
const temperature = 35

if (temperature < 0) {
  console.log(temperature, 'Gelées')
} else if (temperature >= 10 && temperature < 30) {
  console.log(temperature, 'Journée tempérée')
} else if (temperature > 40 && temperature < 50) {
  console.log(temperature, 'Fortes chaleurs')
} else  {
  console.log(temperature, 'Inconnu')
}
```

Solution n°12

[exercice p. 22]

```
const temperature = 35

if (temperature < 0) {
  console.log(temperature, 'Gelées')
} else if ((temperature >= 5 && temperature <= 9) || temperature === 21) {
  console.log(temperature, 'Température recherchée')
} else if (temperature >= 10 && temperature < 30) {
  console.log(temperature, 'Journée tempérée')
```

```
} else if (temperature > 40 && temperature < 50) {
  console.log(temperature, 'Fortes chaleurs')
} else  {
  console.log(temperature, 'Inconnu')
}
```

Solution n°13

[exercice p. 23]

Exercice

Quelle est la différence entre l'opérateur d'égalité et l'opérateur d'égalité stricte en JavaScript ?

- A L'opérateur stricte n'accepte pas les nombres décimaux.
- B L'opérateur stricte permet de comparer le type en plus de la valeur. ✓
- C L'opérateur d'égalité stricte n'accepte pas les opérandes qui ne sont pas des variables.

Exercice

En JavaScript, la valeur entière 2 est égale à la chaîne de caractères `'2'`.

- A Toujours
- B Jamais
- C Cela dépend de l'opérateur que l'on utilise ✓

L'opérateur de comparaison simple == conclura à l'égalité, car le type de la variable n'est pas pris en compte, seules les valeurs sont examinées. L'opérateur de comparaison stricte === indiquera que les deux variables ne sont pas égales, car elles sont de type différent (`number` et `string`).

Solution n°14

[exercice p. 23]

Exercice

Quelles fonctions logiques sont utilisables en algèbre booléenne ?

- A ET ✓
- B OU ✓

C NI

D NOT ✓

Exercice

Une instruction conditionnelle :

A est au minimum composée de `if` et `else`.

B est au minimum composée de `if` et `else if`.

C peut contenir un nombre illimité d'alternatives `else if`. ✓

D peut être imbriquée dans une autre instruction conditionnelle. ✓

E peut contenir plusieurs clauses `else`.

Une structure conditionnelle peut n'être composée que d'un simple `if`, sans alternatives.

En revanche, le nombre de `else if` est illimité, mais il ne peut y avoir qu'un seul `else`, exécuté si aucune des conditions n'est remplie.

Solution n°15

[exercice p. 24]

Exercice

Quelle comparaison renvoie `true` en JavaScript ?

A '10.0' == 10 ✓

L'opérateur de comparaison simple ne tient pas compte du type, mais compare uniquement les valeurs. Or 10.0 est équivalent à 10.

B '10.0' === 10

C '10' == 10 ✓

10.0 === 10

D

Les nombres entiers et les nombres flottants sont représentés par le même type, `number`, en JavaScript.

Exercice

Quelle(s) expression(s) tradui(sen)t en JavaScript la condition : *« angle droit et côté1 et côté2 égaux »* ?

```
angle = 90 && cote1 = cote2
```

A

L'opérateur = est l'opérateur d'**assignation**, non de comparaison.

B `angle == 90 and cote1 == cote2`

C `angle === 90 && cote1 === cote2` ✓

D `angle == 90 || cote1 == cote2`

E `angle == 90 && cote1 == cote2` ✓

L'utilisation de l'opérateur de comparaison simple == ou de l'opérateur de comparaison stricte === fonctionnera dans les deux cas, mais il est préférable d'utiliser l'opérateur de comparaison stricte pour éviter les mauvaises surprises.

Solution n°16

[exercice p. 25]

```
Mention exceptionnelle
```

Solution n°17

[exercice p. 25]

Il n'affiche rien, parce qu'aucune alternative n'existe pour indiquer quoi faire si une note n'est pas égale à A mais est néanmoins valide (entre A et F).

Solution n°18

[exercice p. 25]

```
const grade = 'C'

if (grade === 'A') {
  console.log('Mention exceptionnelle')
} else if (grade !== 'B' && grade !== 'C' && grade !== 'D' && grade !== 'E' && grade !== 'F') {
  console.log('Pas de note')
}

```

La vérification `grade !== 'A'` est inutile : en effet les conditions du `else if` ne seront évaluées que si `grade` n'est pas égal à A.

Solution n°19

[exercice p. 25]

```
const grade = 'C'

if (grade === 'A') {
  console.log('Mention exceptionnelle')
} else if (grade === 'B') {
  console.log('Mention très bien')
} else if (grade === 'C') {
  console.log('Mention bien')
} else if (grade === 'D' || grade === 'E') {
  console.log('Admis')
} else if (grade !== 'F') {
  console.log('Pas de note')
}

```

Solution n°20

[exercice p. 26]

```
const grade = 'F'

if (grade === 'A') {
  console.log('Mention exceptionnelle')
} else if (grade === 'B') {
  console.log('Mention très bien')
} else if (grade === 'C') {
  console.log('Mention bien')
} else if (grade === 'D' || grade === 'E') {
  console.log('Admis')
} else if (grade === 'F') {
  console.log('Non admis')
} else {
  console.log('Pas de note')
}

```