

Éléments de syntaxe du langage C, partie 3 et 4

Sébastien Jean

IUT de Valence
Département Informatique

v1.0, 17 octobre 2025

Opérateurs relationnels

- Inégalité : $>$, $>=$, $<$, $<=$
- Égalité : $==$
- Différence : $!=$
- Pour l'évaluation des expressions (ex : $a == b < c$) :
 - $>$, $>=$, $<$ et $<=$ sont prioritaires sur $==$ et $!=$
 - Les opérateurs de même priorité sont évalués de gauche à droite

Opérateurs logiques

- **Négation** (NON) : !
- **Conjonction** (ET) : &&
- **Disjonction** (OU) : ||
- Pour l'évaluation des expressions (ex : `a && b || ! c`) :
 - ! est prioritaire devant tous les opérateurs arithmétiques et relationnels
 - || est moins prioritaire que && qui est moins prioritaire que tous les opérateurs arithmétiques et relationnels
 - Les opérateurs de même priorité sont évalués de gauche à droite

Opérateurs bit-à-bit et de décalage

- Ca existe mais c'est pour plus tard ...



Structure de contrôle if ...else

- **if ...else** → SI ...ALORS ...SINON en pseudo-code

Syntaxe

```
if (condition) {  
    instructions si condition vraie  
}  
else {  
    instructions si condition fausse  
}
```

- La clause **else** est **optionnelle**
- Les instructions peuvent être des structures de contrôles, donc parfois des **if imbriqués**

Traduction en C de *FizzBuzz*

Enoncé du problème

- Etant donné un nombre entier naturel n strictement positif, *Fizz buzz* vaut :

- 1 si le nombre est divisible par 3
- 2 s'il est divisible par 5
- 3 s'il est à la fois divisible par 3 et 5
- 0 sinon



- Pré-condition : $n \geq 1$

Traduction en C de *FizzBuzz*

Pseudo Code

```
FONCTION fizz_buzz (n : entier) : entier
  SI n mod 3 = 0 ALORS
    SI n mod 5 = 0 ALORS
      RETOURNER 3
    SINON
      RETOURNER 1
  FIN SI
SINON
  SI n mod 5 = 0 ALORS
    RETOURNER 2
  SINON
    RETOURNER 0
  FIN SI
FIN SI
FIN FONCTION
```



Traduction en C de *FizzBuzz*

Code C

```
#include <stdio.h>

int fizz_buzz(int n) {

    if (n % 3 == 0) {
        if (n % 5 == 0) { return 3; }
        else { return 1; }
    }
    else {
        if (n % 5 == 0) { return 2; }
        else { return 0; }
    }
}

... (suite après)
```

Traduction en C de *FizzBuzz*

Code C

```
...  
  
int main() {  
  
    int n = 15;  
    printf("n : %d\n", n);  
    printf("fizzbuzz : %d\n", fizz_buzz(n));  
}
```

Traduction en C de *FizzBuzz*



- En considérant que les pré-conditions sont remplies (pas de traitement des cas d'erreur) :
 - **Définir** un **jeu d'essai** (dans un fichier `essai` à la racine du répertoire)
 - **Valider l'algorithme** avec le jeu d'essai

Structure de contrôle switch ...case

- `switch ...case` → SELON en pseudo-code

Syntaxe

```
switch (expression de type entier) {  
  case valeur immédiate :  
    instructions  
  case valeur immédiate :  
    instructions  
  default :  
    instructions  
}
```

- La clause `default` (aucune valeur ne correspond) est **optionnelle**

Structure de contrôle switch ...case

- La structure switch ...case est particulière
 - Les instructions d'une clause **case** ne sont pas délimitées par un bloc
 - **Selon la valeur de l'expression, on exécute la séquence d'instruction à partir de l'étiquette case correspondante jusqu'à la sortie du switch**
 - L'instruction **break** permet de provoquer la sortie du switch

Structure de contrôle switch ...case

Exemple

```
int c = 0;
int a = 1 ;
switch (a) {
    case 0 :
        c = -1;
    case 1 :
        c = 1;
    default :
        c = 2;
}
```

- Que vaut c à la sortie du switch ?

Structure de contrôle switch ...case

Exemple

```
int c = 0;
int a = 1 ;
switch (a) {
    case 0 :
        c = -1;
    case 1 :
        c = 1;
        break;
    default :
        c = 2;
}
```

- Que vaut c à la sortie du switch ?

Structure de contrôle switch ...case

Exemple

```
int c = 0;
int a = 0 ;
switch (a) {
    case 0 :
        c = -1;
    case 1 :
        c = 1;
        break;
    default :
        c = 2;
}
```

- Que vaut c à la sortie du switch ?

Structure de contrôle switch ...case

Exemple

```
int c = 0;
int a = -1 ;
switch (a) {
    case 0 :
        c = -1;
    case 1 :
        c = 1;
        break;
    default :
        c = 2;
}
```

- Que vaut c à la sortie du switch ?

Structure de contrôle for

- **for** → POUR en pseudo-code

Syntaxe

```
for (int c = 0; c <= 10; c = c + 1) {  
    instructions  
}
```

- **3 parties distinctes** :
 - Déclaration / initialisation du **compteur** de boucle
 - Condition d'**entrée** dans la boucle (vraie pour entrer)
 - Evolution du **compteur** avant le prochain tour

Structure de contrôle while et do ...while

- **while** → TANT QUE en pseudo-code

Syntaxe

```
// entrée dans la boucle tant que condition vraie  
while (condition) {  
    instructions  
}
```

- **do ...while** → REPETER ...TANT QUE en pseudo-code

Syntaxe

```
do {  
    instructions  
} while (condition)  
// poursuite de la boucle tant que la condition est vraie  
// sortie de la boucle quand la condition est fausse
```

break et continue

- Utilisables dans une structure de contrôle for, while et do ...while
- break → sortie immédiate (sans condition) de la boucle
- continue → passage immédiat (sans condition) au tour suivant

Exemple

```
while (...) {  
    ...  
    if (...) {  
        continue;  
    }  
    ...  
    if (...) {  
        break;  
    }  
    ...  
}
```

Fonction de calcul de la factorielle

Pseudo-code

```
FONCTION factorielle (n : entier) : entier  
  
    VARIABLE resultat : entier  
    VARIABLE i        : entier  
  
    resultat ← 1  
    POUR i DE 1 A n PAR PAS DE 1  
        resultat ← resultat * i  
    FIN POUR  
    RETOURNER resultat  
  
FIN FONCTION
```



- Pré-conditions : $n \geq 0$

Fonction de calcul de la factorielle

Programme C

```
int factorielle(int n) {  
  
    int resultat = 1;  
  
    for (int i=1; i <= n; i = i+1) {  
        resultat = resultat * i;  
    }  
  
    return resultat;  
}  
  
... (suite après)
```

Fonction de calcul de la factorielle

Programme C

...

```
int main() {  
  
    int n = 5;  
  
    printf("%d! = %d\n", n, factorielle(n));  
  
    return 0;  
}
```

Fonction de calcul de la factorielle



- En considérant que les pré-conditions sont remplies (pas de traitement des cas d'erreur) :
 - **Valider l'algorithme** pour l'intervalle $[0, 10]$

Fin !

