Programmation en C — Cours 2

6–17 septembre 2021

1 Conditionnelles simples

```c
int
main(void)
{
    int i;
    scanf("%d", &i);
    if(i <= 5) {
        printf("Inférieur ou égal à 5.\n");
    } else {
        printf("Pas inférieur ou égal à 5.\n");
    }
    return 0;
}
```

La conditionnelle `if` consiste du mot clé `if` suivi d'une condition entre parenthèses et de deux *branches* séparées par le mot clé `else`. Si la condition est vraie, c'est la première branche qui est exécutée ; sinon, c'est la deuxième. La deuxième branche peut être omise lorsqu'elle est vide.

2 Conditionnelles imbriquées

```c
int
main(void)
{
    int i, j;
    scanf("%d", &i);
    scanf("%d", &j);
    if(i == j) {
        printf("Match nul.\n");
    } else {
        if(i > j)
            printf("Le premier joueur a gagné.\n");
        else
            printf("Le deuxième joueur a gagné.\n");
    }
    return 0;
}
```

Une conditionnelle peut aussi apparaître au sein d'une des deux branches d'une autre conditionnelle. On parle alors de *conditionnelles imbriquées*[1].

3 Conditionnelles filées

```c
int
main(void)
{
    int i, j;
    scanf("%d", &i);
    scanf("%d", &j);
    if(i == j) {
        printf("Match nul.\n");
    } else if(i > j) {
        printf("Le premier joueur a gagné.\n");
    } else {
        printf("Le deuxième joueur a gagné.\n");
    }
    return 0;
}
```

Une suite de conditionnelles imbriquées augmente l'indentation du code. On peut éviter ce problème en omettant les accolades de chaque **else**, et en mettant chaque **if**

[1] Remarquez que les accolades peuvent être omises lorsqu'une branche ne consiste que d'une seule instruction.

sur la même ligne que le `else` qui précède. Une telle suite de conditionnelles s'appelle une *conditionnelle filée*.

4 Conditions

Dans le cas le plus simple, une condition est simplement constituée de deux expressions numériques séparées par un opérateur de comparaison. Le C connaît les opérateurs de comparaison suivants : « `==` » égal[2], « `!=` » non égal, « `<` » strictement inférieur, « `>` » strictement supérieur, « `<=` » inférieur, et enfin « `>=` » supérieur.

Les conditions peuvent aussi être combinées à l'aide des opérateurs dits *booléens*[3]. Le C reconnaît les opérateurs booléens suivants : « `&&` » (« et ») vrai si ses deux opérandes sont tous deux vrais, faux sinon ; « `||` » (« ou »), faux si ses deux opérandes sont tous deux faux, vrai sinon ; et enfin « `!` » (« non »), vrai si son opérande est faux, faux sinon.

[2]Le double symbole d'égalité « `==` » est utilisé pour éviter la confusion avec l'affectation.
[3]Du nom de George Boole, un logicien anglais du 19e siècle.