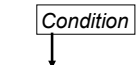


Les conditions

Les conditions

L'instruction `if` fait apparaître une **condition** entre parenthèses



```
if (n < 5) {  
    cout << "Votre nombre est plus petit que 5." << endl;  
} else {  
    cout << "Votre nombre est plus grand ou egal a 5." << endl;  
}
```

Attention, la condition est toujours entourée de parenthèses.

Valeurs des conditions

Une condition est un cas particulier d'expression, **qui ne peut prendre que deux valeurs**.

En C++, ces valeurs se notent `true` et `false`.

- Une condition vaut `true` quand elle est vraie, et
- une condition vaut `false` quand elle est fausse.

Par exemple, la condition `n < 5` vaut `true` si `n` vaut `0`, et `false` si `n` vaut `10`.

Pour l'instant, nous n'avons rencontré que des conditions simples, comme `n < 5` ou `x == y`.

Nous allons voir maintenant comment s'écrivent les conditions d'une façon générale.

Les conditions

L'instruction `if` fait apparaître une **condition** entre parenthèses

Condition

```
if (n < 5) {
    cout << "Votre nombre est plus petit que 5." << endl;
} else {
    cout << "Votre nombre est plus grand ou egal a 5." << endl;
}
```

Attention, la condition est toujours entourée de parenthèses.

Pour l'instant, nous n'avons rencontré qu'une condition simple, `n < 5`

Nous allons voir maintenant comment s'écrivent les conditions d'une façon générale.

Les conditions simples Les opérateurs de comparaison

Une **condition simple** compare deux expressions.

Elle utilise un **opérateur de comparaison**, comme `<` ou `>`

Opérateurs de comparaison du langage C++:

Opérateur de comparaison	Signification
<code><</code>	inférieur à
<code>></code>	supérieur à
<code>==</code>	égal à
<code><=</code>	inférieur ou égal à
<code>>=</code>	supérieur ou égal à
<code>!=</code>	différent de

Les conditions Les opérateurs de comparaison

Attention:

L'opérateur pour tester si deux valeurs sont égales s'écrit avec deux signes égal `==`

Une condition simple compare deux expressions.

Elle utilise un

Opérateurs de

Un seul signe `=` représente l'affectation

Par exemple, si on veut tester si la variable `n` est égale à 5, il faut écrire:

`if (n == 5)`

et non pas:

~~`if (n = 5)`~~

<code>==</code>	égal à
<code><=</code>	inférieur ou égal à
<code>>=</code>	supérieur ou égal à
<code>!=</code>	différent de

Attention:
Il n'y a pas d'espaces entre les deux caractères

Opérateur de comparaison	Signification
<code><</code>	inférieur à
<code>></code>	supérieur à
<code>==</code>	égal à
<code><=</code>	inférieur ou égal à
<code>>=</code>	supérieur ou égal à
<code>!=</code>	différent de

```

int a(1);
int b(2);

if (a == b) {
    cout << "Cas 1" << endl;
} else {
    cout << "Cas 2" << endl;
}

if (2 * a == b) {
    cout << "b est egal au double de a." << endl;
}

```

affiche

Cas 2

b est egal au double de a.

```

int a(1);
int b(2);

if (a != b) {
    cout << "Cas 2" << endl;
} else {
    cout << "Cas 1" << endl;
}

if (2 * a != b) {
    cout << "b est different du double de a." << endl;
}

```

affiche

Cas 2

b est different du double de a.

```

int a(1);
int b(2);

if (a <= b) {
    cout << "Cas 3" << endl;
} else {
    cout << "Cas 4" << endl;
}

if (2 * a <= b) {
    cout << "b est superieur ou egal au double de a." << endl;
}

```

affiche

Cas 3

b est superieur ou egal au double de a.

Les opérateurs logiques

On peut relier des conditions simples par des opérateurs logiques.

L'opérateur logique and (ET):

par exemple, la condition

$(a < b) \text{ and } (c < d)$

est vraie **uniquement** si les deux conditions $(a < b)$ et $(c < d)$ sont toutes les deux vraies.

L'opérateur and peut aussi s'écrire &&: On aurait pu écrire

$(a < b) \text{ \&\& } (c < d)$

Exemple avec l'opérateur logique and

```
cout << "Entrez un nombre entre 1 et 10:" << endl;
cin >> n;
```

faux

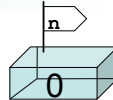
```

    faux
  {
if ((n >= 1) and (n <= 10)) {
    cout << "correct" << endl;
} else {
    cout << "incorrect" << endl;
}
}

```

Supposons que la valeur
entrée pour n soit 0

incorrect est affiché



Exemple avec l'opérateur logique and

```
cout << "Entrez un nombre entre 1 et 10:" << endl;
cin >> n;
```

faux

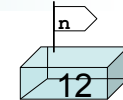
```

    vrai    faux
  {
if ((n >= 1) and (n <= 10)) {
    cout << "correct" << endl;
} else {
    cout << "incorrect" << endl;
}
}

```

Supposons que la valeur
entrée pour n soit 12

incorrect est affiché



Exemple avec l'opérateur logique and

```
cout << "Entrez un nombre entre 1 et 10:" << endl;
cin >> n;
```

vrai

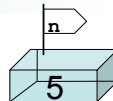
```

    vrai    vrai
  {
if ((n >= 1) and (n <= 10)) {
    cout << "correct" << endl;
} else {
    cout << "incorrect" << endl;
}
}

```

Supposons que la valeur
entrée pour n soit 5

correct est affiché



Les opérateurs logiques

L'opérateur logique or (OU):

par exemple, la condition

$(a < b) \text{ or } (c < d)$

est vraie **si au moins une** des deux conditions $(a < b)$ ou $(c < d)$ est vraie.

L'opérateur or peut aussi s'écrire || : On aurait pu écrire

$(a < b) \text{ || } (c < d)$

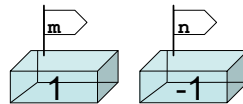
Exemple avec l'opérateur logique `or`

```
cout << "Entrez deux valeurs:" << endl;
cin >> m >> n;
```

```

      vrai
    -----
    vrai
  -----
if ((m >= 0) or (n >= 0)) {
    cout << "au moins une valeur est positive" << endl;
} else {
    cout << "les deux valeurs sont negatives" << endl;
}

```



Supposons que la valeur entrée pour `m` soit +1, et la valeur entrée pour `n` soit -1

au moins une valeur est positive est affiché

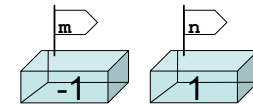
Exemple avec l'opérateur logique `or`

```
cout << "Entrez deux valeurs:" << endl;
cin >> m >> n;
```

```

      faux      vrai
    -----
    faux
  -----
if ((m >= 0) or (n >= 0)) {
    cout << "au moins une valeur est positive" << endl;
} else {
    cout << "les deux valeurs sont negatives" << endl;
}

```



Supposons que la valeur entrée pour `m` soit -1, et la valeur entrée pour `n` soit +1

au moins une valeur est positive est affiché

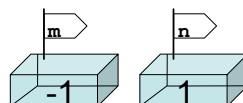
Exemple avec l'opérateur logique `or`

```
cout << "Entrez deux valeurs:" << endl;
cin >> m >> n;
```

```

      faux      faux
    -----
    faux
  -----
if ((m >= 0) or (n >= 0)) {
    cout << "au moins une valeur est positive" << endl;
} else {
    cout << "les deux valeurs sont negatives" << endl;
}

```



Supposons que la valeur entrée pour `m` soit -1, et la valeur entrée pour `n` soit -1

les deux valeurs sont negatives est affiché

Les opérateurs logiques

L'opérateur logique `not` (NON):

par exemple, la condition

`not(a < b)`

est vraie si $(a < b)$ est fausse, et fausse si $(a < b)$ est vraie.

L'opérateur `not` peut aussi s'écrire `!`: On aurait pu écrire

`!(a < b)`

Nous verrons des exemples d'utilisation de cet opérateur plus loin dans la suite du cours.