Structure conditionnelle

- SI
- Opérateur de comparaison
- Opérateur d'aggrégation
- Booléen
- SINON
- SINON SI
- Priorité des opérations

Structure conditionnelle

- Jusqu'ici nous faisions des structures impératives. C'est-à-dire que chaque ligne de code était exécuté, une à la suite de l'autre.
- Nous allons maintenant ajouter des structures conditionnelles. C'est-à-dire que certaine ligne de code ne s'exécuteront, que si une condition est respecté.
- Cela ce fait en 2 étapes:
    - Déterminer la conditions
    - Écrire dans un bloc, le code qui sera exécuté si la condition est respecté.

SI

Voyons un exemple du concept en pseudo-code pour un programme qui détermine si une personne est majeur à partir de son âge.

```
entier AGE_OF_MAJORITY = 18
entier age;

ECRIRE "Veuillez écrire votre âge: "
LIRE age

SI age >= AGE_OF_MARJORITY   // Déterminer la condition.  La condition doit toujours être un valeur booléene (vrai / faux)
    ECRIRE "Vous êtes un adulte"  // Écrire le bloc de code qui s'exécuter seulement si la condition est respecté.
FINSI  // Fin du bloc de code propre à condition
```

Opérateur de comparaison

- Un opérateur de comparaison sert à comparer deux valeur et à retourner une valeur booléenne issue du résultat de la comparaison.
- Voici la liste des opérateurs de comparaison que nous pouvons utiliser pour déterminer une valeur booléenne.
    - `==` égalité *Notez ici l'utilisation de 2* `=` *, car l'opérateur* `=` *est utilisé pour l'affectation.*
    - `!=` différent de
    - `>` plus grand que
    - `<` plus petit que
    - `>=` plus grand ou égale à
    - `<=` plus petit ou égale à
    - `!` non

Opérateur d'aggrégation

- Il est possible de combiner plusieurs comparaisons dans une seule condition avec les opérateurs d'aggrégation ET et OU.
- Par exemple, si je reprend mon exemple précédent, mais que cette fois je veux vérifier si la personne est majeur au Québec, mais ne l'est pas au États-unis.

```
entier AGE_OF_MAJORITY_QC = 18
entier AGE_OF_MAJORITY_USA = 21
entier age;

ECRIRE "Veuillez écrire votre âge: "
LIRE age

SI age >= AGE_OF_MAJORITY_QC ET age < AGE_OF_MAJORITY_USA
    ECRIRE "Vous êtes un adulte au Québec, mais pas au États-unis"
FINSI
```

ET : Les deux conditions doivent être vrai pour entrer dans la condition

OU : Une des deux conditions doit être vrai pour entrer dans la condition

Booléen

- Nous avons vue que les variables de type booléen contiennent une valeur vrai ou faux.
- Ces variables peuvent donc être utilisé directement dans une condition ou même pour stocker une condition.
- Par exemple:

```
entier AGE_OF_MAJORITY = 18
entier age;

ECRIRE "Veuillez écrire votre âge: "
LIRE age

BOOLEEN isInQc = true
BOOLEEN isMajor = age >= AGE_OF_MARJORITY

SI isMajor ET isInQc
    ECRIRE "Vous êtes un adulte"
FINSI
```

SINON

- Chaque bloc de condition peut avoir un second bloc de code qui s'exécute uniquement si la condition n'est pas respecté.
- On appel ce bloc le SINON.
- Par exemple:

```
entier AGE_OF_MAJORITY = 18
entier age;

ECRIRE "Veuillez écrire votre âge: "
LIRE age

SI age >= AGE_OF_MAJORITY
    ECRIRE "Vous êtes un adulte"
SINON
    ECRIRE "Vous n'êtes pas encore un adulte"
FINSI
```

SINON SI

- En plus du SI et du SINON, il possible d'insérer des blocs de code supplémentaire en fonction de condition supplémentaire.
- On appelle ces blocs des SINON SI. Chacun d'eux doit avoir une condition pour déterminer si le programme doit exécuter le code du bloc du SINON SI.

```
entier AGE_OF_MAJORITY_QC = 18
entier AGE_OF_MAJORITY_USA = 21
entier age;

ECRIRE "Veuillez écrire votre âge: "
LIRE age

SI age >= AGE_OF_MAJORITY_USA
    ECRIRE "Vous êtes un adulte partout."
SINON SI age >= AGE_OF_MAJORITY_QC
    ECRIRE "Vous êtes un adulte, mais pas au États-Unis."
SINON
    ECRIRE "Vous n'êtes pas encore un adulte."
FINS
```

Priorité des opérations

- Notez que lorsqu'on fait des aggrégations, le **ET** est prioritaire sur le **OU**
- Vous pouvez aussi utilisez des parenthèses pour préciser vos priorités.
- Par exemple:

```
3 > 4 ET 4 == 4 OU !(3 > 2 OU 4 < 2)
false ET true OU !(true OU false) -> Convertir les comparaisons en valeur booléeene
false ET true OU !true -> Régler les parenthèse
false ET true OU false -> ! sert à inverser la valeur booléen.  !true == false
false OU false -> Le ET est prioritaire sur le OU
false
```

Quiz

- Est-ce que le programme suivant affichera "VRAI" ou "FAUX"

```
BOOLEN isGood = true;
SI !isGood
    ECRIRE "VRAI"
SINON
    ECRIRE "FAUX"
FINSI
```

```
!isGood
!true
false
"FAUX"
```

- Est-ce que le programme suivant affichera "VRAI" ou "FAUX"

```
ETNIER age = 56;
SI age > 75 || age < 60
    ECRIRE "VRAI"
SINON
    ECRIRE "FAUX"
FINSI
```

```
56 > 75 || 56 < 60
false  ||  true
true
"VRAI"
```

- Est-ce que le programme suivant affichera "VRAI" ou "FAUX"

```
ENTIER a = 10;
ENTIER b = 15;
SI !(a == b && a > 5 || b < 20 && b % 2 == 0)
    ECRIRE "VRAI"
SINON
    ECRIRE "FAUX"
FINSI
```

```
!(a == b && a > 5 || b < 20 && b % 2 == 0)
!(10 == 15 && 10 > 5 || 15 < 20 && 15 % 2 == 0)
!(10 == 15 && 10 > 5 || 15 < 20 && 1 == 0)
!(false && true || true && false)
!(false || false)
!false
true
"VRAI"
```