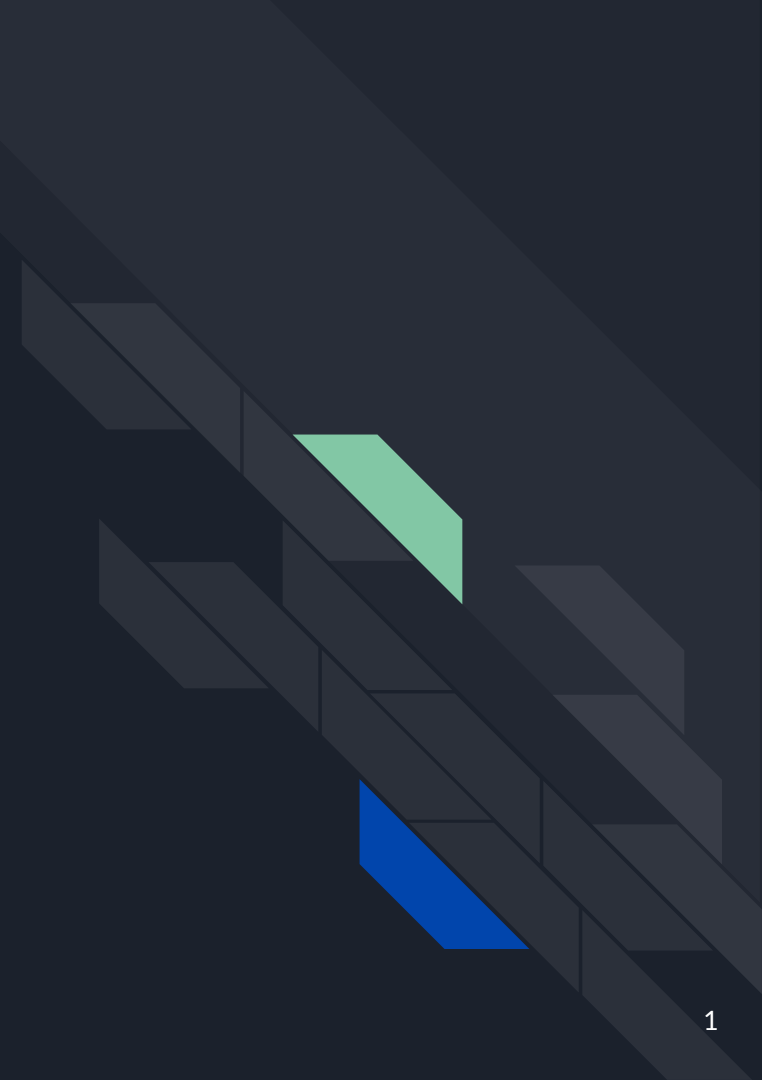


# Structures de contrôle





# Définition

Les **structures de contrôle** permettent d'exécuter leurs blocs d'instructions associés sous **conditions**.

L'expression conditionnelle testée en condition par une structure de contrôle a toujours un résultat booléen.



# Bloc d'instructions

Un **bloc d'instructions** commence et fini par une **accolade** :

```
{ // Début de bloc  
    ... // Instructions  
} // Fin de bloc
```

# If

**if** : Exécute le bloc d'instructions suivant uniquement si l'expression conditionnelle vaut "vrai".

```
if (expression) {  
    ... // Instructions  
}
```

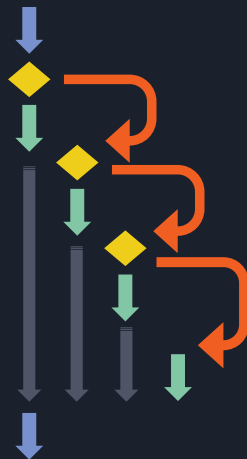


# If / Else if / Else

**else if** (optionnel et multiple) : Si la première condition n'est pas remplie, teste la seconde et ainsi de suite.

**else** (optionnel) : Si aucune condition n'est remplie, exécute le bloc.

```
if(exp1) {  
    ... // Instructions  
} else if(exp2) {  
    ... // Instructions  
} else if(exp3) {  
    ... // Instructions  
} else {  
    ... // Instructions  
}
```



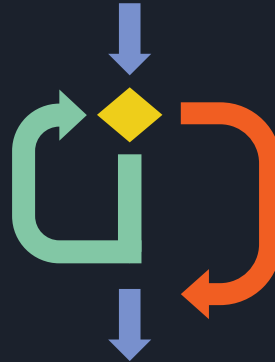
Un seul bloc d'instruction sera exécuté dans une structure "if / else if / else"

# While

**while** : Tant que la condition est vraie, exécute le bloc d'instructions puis revient au début.

```
int i = 0;
while (i < 10) {
    std::cout << "Loop";
    i++;
}
```

Affiche 10 fois "Loop".



# Do / While

**Do / while** : Exécute le bloc d'instructions puis revient au début, tant que la condition est vraie.

```
int i = 100;  
do {  
    std::cout << "Loop";  
    i++;  
} while (i < 10);
```

Affiche 1 fois "Loop".

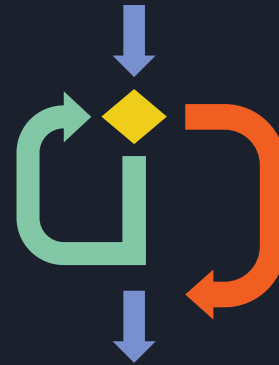


# For

**for** : Tant que la condition est vraie, exécute le bloc d'instruction, revient au début puis applique le pas.

```
for (int i = 0; i < 10; i++) {  
    std::cout << "Loop."  
}
```

Affiche 10 fois "Loop."



“for” est moins sujet à erreurs que “while”.





# Switch/case

**Switch/Case** : Équivalent à une structure “if / else if / else” avec des tests “==” de valeurs entières.

```
switch (myInt) {  
    case 0:  
        // myInt == 0. Instructions.  
        break; // break permet de sortir de la structure de contrôle  
    case 1:  
    case 2:  
        // myInt == 1 || myInt == 2. Instructions.  
        break;  
    default:  
        // myInt not equal to any case. Default instructions.  
        break;  
}
```



# Continue et break

Dans une structure de contrôle en boucle **while**, **do** ou **for**, le mot-clé **continue** saute directement à la dernière ligne du bloc d'instruction et commence une nouvelle boucle.

Dans une structure de contrôle **while**, **do**, **for** ou **switch**, le mot-clé **break** sort immédiatement de la structure de contrôle.



# Goto

**Goto** ? Ca existe !

... Oui ... On peut dire ça...

Ca existe ...

**Goto / label** : succédé d'un identifiant de label, goto déplace le curseur d'exécution directement à l'emplacement de ce label.

Cette structure de contrôle, difficile à maintenir et source d'erreur, n'est généralement pas utilisée par les programmeurs.