INF155 - Structures de contrôle

Hugo Leblanc

Cours 2

Structure de contrôle conditionnelle - if

Rappel

Le if nous permet d'exécuter un bloc de code sous une condition.

```
if (condition) {
    //Instruction
}
// Exemple
if (x > 4) {
    printf("Dans le if!");
}
```

else

L'ajout du else nous donne accès à des instructions alternatives si la condition n'est pas respectée.

```
if (x > 4) {
    printf("Dans le if!");
} else {
    printf("Dans le else!");
}
```

else if

Plusieurs conditionnelles peuvent être enchaînées les unes après les autres pour donner des conditions mutuellement exclusives.

```
if (x < 10) {
    printf("Plus petit que 10.");
} else if (x < 20) {
```

```
    printf("Plus petit que 20, mais plus grand ou égal à 10.");
} else if (x < 30) {
    printf("Plus petit que 30, mais plus grand ou égal à 20".);
}
```

Présentation du code

Commentaires

- En-tête
  - Nom
  - Description
  - Auteur
- Intérieur des blocs de code
  - Description des actions
  - Un commentaire par action

Espacement

- Indentation horizontale
- Espacement vertical (saut de ligne)

Noms significatifs

- Noms significatifs au contexte
- Utilisation du tiret-bas pour l'espace

La commande #define et les constantes

La commande de préprocesseur #define permet de nommer une valeur statique qui sera représentée dans le code. L'utilisation de constante se fera avec #define. Par convention, les noms des #constantes seront en majuscules.

Faire attention, les commandes de préprocesseur ne sont pas des instructions et n'ont donc pas besoin du point-virgule.

Exemple d'utilisation de #define

```
#include <stdio.h>

#define MA_CONSTANTE 56

int main(void){
    printf('La constante est : %d', MA_CONSTANTE);

    return 0;
}
```

Parler du return 0 et de EXIT_SUCCESS avec stdlib.h

Opérateurs

Opérateur logique

Les opérateurs logiques opèrent sur des valeurs booléennes. Les résultats des opérations logiques seront à leur tour des valeurs booléennes.

Opération	Opérateur	Résultat
Conjonction ET	a && b	Vrai si a et b sont vrai.
Disjonction OU	a || b	Vrai si a ou b sont vrai.
Négation NON	!a	Inverse de a.

Opérateur binaire

Les opérateurs binaires opèrent sur les données binaires des valeurs.

Opération	Opérateur	Résultat
Conjonction ET	a & b	Conjonction bit par bit.
Disjonction OU	a | b	Disjonction bit par bit.
Disjonction exclusive XOR	a ^ b	Disjonction exclusive bit par bit.
Inversion NON	~x	Inversion bit par bit.

Opérateur d'assignation

Les opérateurs d'assignation sont des raccourcis syntaxiques pour jumeler une assignation avec une opération. On ajoute l'opérateur avant l'opérateur assigna-

tion (=).

Exemple d'utilisation

```c
#include <stdio.h>

int main(void){
    int x = 5;
    x += 10;
    x <<= 2;
    printf("La valeur de x est : %d", x);
}
```

Structure de contrôle - switch case

Le switch case travaille sur une expression et la compare à de possibles valeurs littérales.

```c
switch (expression) {
    case cas1:
        // Instructions 1
    case cas2:
        // Instructions 2
    default:
        // Le "else" du switch
}
```

Instruction break

Par défaut, le switch a un passe-droit (falltrough). Pour briser le cycle du switch prématurément, on peut utiliser l'instruction break.

```c
switch (expression) {
    case cas1:
        // Instructions 1
        break;
    case cas2:
        // Instructions 2
        break;
    default:
        // Le "else" du switch
}
```

Structures itératives - while

La boucle while répète les instructions jusqu'à ce que la condition soit fausse.

Le while est habituellement utilisé quand on ne connaît pas le nombre d'itérations à faire.

```
while (condition) {
    // Instructions
}
```

do while

Le do while oblige d'avoir une première itération de la boucle, mais fonctionne autrement comme le while.

```
do {
    / Instructions
} while (condition); // Faire attention au ; ici..
```

Structures itératives - for

La boucle for inclut une instruction d'initialisation et d'incrémentation de fin de boucle en plus de la condition habituelle. Elle est particulièrement utilisée quand on connaît ou qu'il est possible de calculer le nombre d'itérations de la boucle à faire.

```
for (init; condition; incre) {
    // Instructions
}

// Exemple
int i;
for (i = 0; i < 10; i++){
    printf("Compteur : %d", i);
}
```