Les instructions de contrôle

DAKKAR Borhen-eddine

Lycée le Corbusier

BTS SN-IR

Table des matières

Table des matières

Etapes de développement d'un programmes C

- La programmation en C est une programmation séquentielle, c'est-à-dire les instruction sont éxecutées dans l'ordre où elles apparaissent.
- Pour rendre le comportement d'un algorithme intelligent, C offre un ensemble d'outils qui permettent d'automatiser les processus.
- Un programme C peut se comporter différemment suivant les circonstances.
- C permet la possibilité d'effectuer des boucles qui peuvent répéter plusieurs fois un ensemble donné d'instructions.

Expressions relationnelles

- Tous les ordinateurs ont des capacités logiques qui permettent de faire des comparaisons de diverses quantités.
- Une expression relationnelle simple consiste en un opérateur relationnel reliant deux opérandes variables.

$$\underset{\text{opérande}}{1} \quad \underset{\text{Opérateur}}{>} \quad \underset{\text{opérande}}{2}$$

- Dans le cas des expressions relationnelles,la valeur de l'expression ne peut être qu'une valeur entière de 1 ou 0, qui est interprétée respectivement comme vrai et faux.
- Les instructions suivantes :

```
printf("La valeur de 3 > 4 est %d", 3 > 4);
printf("La valeur de 3 < 4 est %d", 3 < 4);
```

affiche :
La valeur de 3 > 4 est 0
La valeur de 3 < 4 est 0

L'instruction if-else

- L'instruction if-else (Si ... alors ...Sinon) demande à l'ordinateur de sélectionner une séquence d'un ou plusieurs instructions basées sur le résultat d'une comparaison.

```
if (expression)
    instructions;
else
    instructions;
```

- Regardons le code suivant :

```
if (n>10)
    printf ("Note supérieur à la moyenne");
else
    printf ("Note inférieur à la moyenne");
```

En fonction de la valeur de **n**, l'un des deux messages sera affiché.

L'instruction if-else

- Les accolades sont utilisées pour entourer un ensemble d'instructions individuelles créant un seul bloc d'instructions.

```
if (expression)
    {instructions1;
    instructions2;}
else
    {instructions3;
    instructions4;}
```

- L'instruction suivant **if (expression)** n'est exécutée que si le expression a une valeur différente de zéro (une condition vraie).

Instructions if imbriquées

- Nous pouvons mettre plusieurs instructions **if** à l'interieur d'une instruction **if**. Cela est appelé instructions imbriquées.

```
if (expression_1)
    instruction1;
else if (expression_2)
    instruction2;
else
    instruction3;
```

- Un **else** se rapporte toujours au dernier **if** rencontré auquel un else n'a pas encore été attribué.

- Chaque condition est évaluée dans l'ordre dans lequel elle apparaît. Pour la première condition qui est vraie, l'instruction correspondante est exécutée et le le reste des instructions de la chaîne ne sont pas exécutées.

Exemple

```c
#include <stdio.h>
#include <stdlib.h>
main( )
{
char matcode;
printf("Entrez un code matrimonial : ");
scanf("%c", &matcode);
if (matcode == 'M')
    printf("\nlndividu est marié.");
else if (matcode == 'S')
    printf("\nlndividu est célibataire.");
else if (matcode == 'D')
    printf("\nlndividu est divorcé.");
else if (matcode == 'W')
    printf("\nlndividu est veuf.");
else
    printf ("\nLe code entré est invalide.");
    printf ("\nMerci d'avoir participé à l'enquête");
}
```

Table des matières

Syntaxe d'une instruction switch

- L'instruction **switch** fournit une alternative à l'instruction **if-else** pour les cas qui comparent les valeurs d'une expression entière à une valeur spécifique.

```
switch (expression)
{
case valeur_1://terminé par un deux-points
    instructions;
    break;
case valeur_2:
    instructions;
    break;
    .
    .
case valeur_n:
    instructions;
    break;
default:
instructions;
}
```

Syntaxe d'une instruction switch

- L'expression entre parenthèses après **switch** est évaluée et le résultat est comparé aux diverses valeurs de **case**.
- Le mot-clé **case** est utilisé pour identifier les valeurs individuelles comparées à la valeur de l'expression **switch**.
- N'importe quel nombre de **case** peut être contenu dans l'expression **switch**. Si la valeur de l'expression ne correspond à aucune des valeurs de **case**, aucune instruction n'est exécutée sauf si le mot-clé **defaut** est rencontré.
- Le mot **default** est facultatif et fonctionne de la même manière que le dernier **else** d'une chaîne **if-else**.
- Une fois un **case** a été identifié, toutes les instructions sont exécutées sauf si une instruction **break** est rencontrée. Cette instruction indique la fin du **case** et provoque une sortie immédiate de l'instruction **switch**.

Exemple

```c
#include <stdio.h>
#include <stdlib.h>
main( )
{
int nombre ;
printf ("donnez un entier : ") ;
scanf ("%d", &nombre) ;
switch (nombre)
{
case 1:
    printf("Vous êtes un élève de BTS SN");
    break;
case 2:
    printf ("Vous êtes un élève de BTS SN\n Au lycée le Corbusier");
    break;
case 3:
    printf ("Vous êtes un élève de BTS SN\n Au lycée le Corbusier");
    printf ("\n à Aubervilliers");
default:
    printf("\nVous n'etes pas un eleve du lycee le Corbusier");
}
}
```

Table des matières

Syntaxe d'une instruction while

- L'instruction while est une instruction de répétition générale qui peut être utilisée dans diverses situations de programmation.

```
while (expression)
    instructions;
```

- L'expression contenue entre parenthèses est évaluée exactement de la même manière manière comme une expression contenue dans une instruction **if-else**.
- L'expression utilisée comme condition de poursuite est évaluée avant le premier tour de boucle.
- L'instruction qui suit l'expression est exécutée à plusieurs reprises tant que l'expression conserve une valeur différente de zéro.
- La sortie de la boucle **while** est conditionnée par la passage de la valeur de l'expression de 1 à 0.

Exemple

```c
#include <stdio.h>
main( )
{
int count;
count = 1; /* initialization de count */
while (count <= 10)
    {
    printf("%d ",count);
    count = count + 1; /* Ajouter 1 à count */
    }
}
```

Après l'initaialisation de count, l'instruction **while** évalue l'expression pour la première fois. Tant que la valeur de count est inférieure ou égale à 10, l'expression est vraie et les instructions sont exécutées.

Table des matières

Syntaxe d'une instruction for

- L'instruction **for** exécute les mêmes fonctions que l'instruction **while**, mais utilise une forme différente.
- Dans de nombreuses situations, en particulier celles qui utilisent une condition de comptage fixe, le format d'instruction for est plus facile à utiliser que son équivalent d'instruction while.

```
for ( expression_1 ; expression_2; expression_3)
    instructions;
```

- **expression_1 :** utilisée pour définir le début (valeur initiale) d'un compteur.
- **expression_2 :** contient la valeur maximale ou minimale que le compteur peut avoir et détermine à quel moment la boucle est terminée.
- **expression_3 :** est la valeur d'incrément qui est ajoutée à ou soustrait du compteur à chaque fois que la boucle est exécutée.

Exemple

```c
#include <stdio.h>
#include <math.h>
main( )
int count;
printf("\nNombre    La racine carrée ");
for (count = 1; count <= 5; count = count + 1)
printf ("\n %d          %f", count,  sqrt(count));
}
```

Nombre	La racine carree
1	1.000000
2	1.414214
3	1.732051
4	2.000000
5	2.236068

Table des matières

Syntaxe d'une instruction do while

- Les instructions **while** et **for** évaluent une expression au début de la boucle de répétition. Dans certains cas, il est plus pratique de tester l'expression à la fin de la boucle.

```
do instruction
   while (expression) ;
```

- Toutes les instructions après **do** sont exécutées au moins une fois avant que l'expression ne soit évaluée.
- Si l'expression a une valeur différente de zéro, les instructions sont à nouveau exécutées. Ce processus se poursuit jusqu'à l'expression est évaluée à zéro.

Exemple

```c
#include <stdio.h>
#include <stdlib.h>
main()
{
int n ;
do
{ printf ("donnez une valeur >5 : ") ;
scanf ("%d", &n) ;
printf ("vous avez fourni %d\n", n) ;
}
while (n<=5) ;
printf ("réponse correcte") ;
}
```

donnez une valeur >5 : 4
vous avez fourni 4
donnez une valeur >5 : 6
vous avez fourni 6
réponse correcte

Références