



High-Tech

Droit/Finances

Santé/Médecine

575690 membres 28865 connectés 3774 questions/jour Taux de réponse: 84.73% Mardi 8 janvier 2008 - 20:29:3



Rechercher :
avec Sur Commen

Accueil | Forum | Astuces | Emploi | Professionnels | Télécharger | Pilotes | Actualités | Liv



Langage C - Structures conditionnelles

Qu'est-ce qu'une structure conditionnelle ?

On appelle **structure conditionnelle** les instructions qui permettent de tester si une condition est vraie ou non. Ces structures conditionnelles peuvent être associées à des structures qui se répètent suivant la réalisation de la condition, on appelle ces structures des **structures de boucle**.

La notion de bloc

Une expression suivie d'un point-virgule est appelée instruction. Voici un exemple d'instruction :

```
a++;
```

Lorsque l'on veut regrouper plusieurs instructions, on peut créer ce que l'on appelle un **bloc**, c'est-à-dire un ensemble d'instructions (suivies respectivement par des points-virgules) et comprises entre les accolades { et }.

Les instructions *if*, *while* et *for* peuvent par exemple être suivies d'un bloc d'instructions à exécuter...

L'instruction if

L'instruction *if* est la structure de test la plus basique, on la retrouve dans tous les langages (avec une syntaxe différente...). Elle permet d'exécuter une série d'instructions si jamais une condition est réalisée.

La syntaxe de cette expression est la suivante :

```
if (condition réalisée) {
    liste d'instructions;
}
```

Remarques :

la condition doit être entre des parenthèses

il est possible de définir plusieurs conditions à remplir avec les opérateurs *ET* et *OU* (&& et //)

Par exemple l'instruction suivante teste si les deux conditions sont vraies :

```
if ((condition1)&&(condition2))
```

L'instruction suivante exécutera les instructions si l'une ou l'autre des deux conditions est vraie :

```
if ((condition1)|| (condition2))
```

Langage C

- [Langage C](#)
- [Caractéristiques](#)
- [Préprocesseur](#)
- [Types de données](#)
- [Les variables](#)
- [Les opérateurs](#)
- [Structures conditionnelles](#)
- [Les fonctions](#)
- [Les tableaux](#)
- [Les structures](#)
- [Les pointeurs](#)
- [Chaîne de caractère](#)
- [Listes chaînées](#)

s'il n'y a qu'une instruction, les accolades ne sont pas nécessaires...

les instructions situées dans le bloc qui suit *else* sont les instructions qui seront exécutées si la ou les conditions ne sont pas re

L'instruction `if ... else`

L'instruction *if* dans sa forme basique ne permet de tester qu'une condition, or la plupart du temps on aimerait pouvoir choisir des instructions à exécuter **en cas de non réalisation de la condition**...

L'expression *if ... else* permet d'exécuter une autre série d'instructions en cas de non-réalisation de la condition.

La syntaxe de cette expression est la suivante :

```
if (condition réalisée) {
    liste d'instructions
}
else {
    autre série d'instructions
}
```

Une façon plus courte de faire un test

Il est possible de faire un test avec une structure beaucoup moins lourde grâce à la structure suivante :

```
(condition) ? instruction si vrai : instruction si faux
```

Remarques :

la condition doit être entre des parenthèses

Lorsque la condition est vraie, l'instruction de gauche est exécutée

Lorsque la condition est fausse, l'instruction de droite est exécutée

En plus d'être exécutée, la structure `?:` renvoie la valeur résultant de l'instruction exécutée. Ainsi, cette forme `?:` est souvent utilisée comme suit :

```
position = ((enAvant == 1) ? compteur+1 : compteur-1);
```

L'instruction `switch`

L'instruction *switch* permet de faire plusieurs tests de valeurs sur le contenu d'une même variable. Ce branchement conditionnel simplifie beaucoup le test de plusieurs valeurs d'une variable, car cette opération aurait été compliquée (mais possible) avec des *if* imbriqués. La syntaxe est la suivante :

```
switch (Variable) {
case Valeur1 :
    Liste d'instructions;
    break;
case Valeur2 :
    Liste d'instructions;
    break;
case Valeurs... :
    Liste d'instructions;
    break;
default:
    Liste d'instructions;
```

```
}
```

Les parenthèses qui suivent le mot clé *switch* indiquent une expression dont la valeur est testée successivement par chacun des cas. Lorsque l'expression testée est égale à une des valeurs suivant un *case*, la liste d'instructions qui suit celui-ci est exécutée. Le mot clé *break* indique la sortie de la structure conditionnelle. Le mot clé *default* précède la liste d'instructions qui sera exécutée si l'expression n'est jamais égale à une des valeurs.

N'oubliez pas d'insérer des instructions *break* entre chaque test, ce genre d'oubli est difficile à détecter car aucune erreur n'est signalée...

En effet, lorsque l'on omet le *break*, l'exécution continue dans les blocs suivants !

Cet état de fait peut d'ailleurs être utilisé judicieusement afin de faire exécuter les instructions pour différentes valeurs consécutives, on peut ainsi mettre plusieurs cases avant un bloc :

```
switch(variable)
{
case 1:
case 2:
{ instructions exécutées pour variable = 1 ou pour variable = 2 }
break;
case 3:
{ instructions exécutées pour variable = 3 uniquement }
break;
default:
{ instructions exécutées pour toute autre valeur de variable }
}
```

Les boucles

Les boucles sont des structures qui permettent d'exécuter plusieurs fois la même série d'instructions jusqu'à ce qu'une condition soit plus réalisée...

On appelle parfois ces structures *instructions répétitives* ou bien *itérations*.

La façon la plus commune de faire une boucle est de créer un compteur (une variable qui s'incrémente, c'est-à-dire qui augmente chaque tour de boucle) et de faire arrêter la boucle lorsque le compteur dépasse une certaine valeur.

La boucle for

L'instruction *for* permet d'exécuter plusieurs fois la même série d'instructions : c'est une boucle !

Dans sa syntaxe, il suffit de préciser le nom de la variable qui sert de compteur (et éventuellement sa valeur de départ, la condition pour laquelle la boucle s'arrête (basiquement une condition qui teste si la valeur du compteur dépasse une limite) et enfin l'instruction qui incrémente (ou décrémente) le compteur.

La syntaxe de cette expression est la suivante :

```
for (compteur; condition; modification du compteur) {
    liste d'instructions;
}
```

Par exemple :

```
for (i=1; i<6; i++) {
    printf("%d", i);
}
```

Cette boucle affiche 5 fois la valeur de *i*, c'est-à-dire 1, 2, 3, 4, 5.

Elle commence à *i*=1, vérifie que *i* est bien inférieur à 6, etc. jusqu'à atteindre la valeur *i*=6, pour laquelle la condition ne sera plus réalisée, la boucle s'interrompt et le programme continuera son cours.

il faudra toujours vérifier que la boucle a bien une condition de sortie (i.e. le compteur s'incrémente correctement) une instruction *printf()*; dans votre boucle est un bon moyen pour vérifier la valeur du compteur

pas à pas en l'affichant !

il faut bien compter le nombre de fois que l'on veut faire exécuter la boucle :

```
for(i=0;i<10;i++) exécute 10 fois la boucle (i de 0 à 9)
for(i=0;i<=10;i++) exécute 11 fois la boucle (i de 0 à 10)
for(i=1;i<10;i++) exécute 9 fois la boucle (i de 1 à 9)
for(i=1;i<=10;i++) exécute 10 fois la boucle (i de 1 à 10)
```

L'instruction while

L'instruction *while* représente un autre moyen d'exécuter plusieurs fois la même série d'instructions.

La syntaxe de cette expression est la suivante :

```
while (condition réalisée) {
    liste d'instructions;
}
```

Cette instruction exécute la liste d'instructions **tant que** (*while* est un mot anglais qui signifie *tant que*) la condition est réalisée.

La condition de sortie pouvant être n'importe quelle structure conditionnelle, les risques de boucle infinie (boucle dont la condition est toujours vraie) sont grands, c'est-à-dire qu'elle risque provoquer un plantage du programme en cours d'exécution !

Saut inconditionnel

Il peut être nécessaire de faire sauter à la boucle une ou plusieurs valeurs sans pour autant mettre fin à celle-ci.

La syntaxe de cette expression est « *continue* » (cette instruction se place dans une boucle !), on l'associe généralement à une structure conditionnelle, sinon les lignes situées entre cette instruction et la fin de la boucle seraient obsolètes.

Exemple : Imaginons que l'on veuille imprimer pour x allant de 1 à 10 la valeur de $1/(x-7)$; il est évident que pour $x=7$ il y a une erreur. Heureusement, grâce à l'instruction *continue* il est possible de traiter cette valeur à part puis de continuer la boucle !

```
x=1;
while (x<=10) {
    if (x == 7) {
        printf("Division par zéro !");
        continue;
    }
    a = 1/(x-7);
    printf("%f", a);
    x++;
}
```

Il y avait une erreur dans ce programme... peut-être ne l'avez-vous pas vue :

Lorsque x est égal à 7, le compteur ne s'incrémente plus, il reste constamment à la valeur 7, il aurait fallu écrire :

```
x=1;
while (x<=10) {
    if (x == 7) {
        printf("Division par 0");
        x++;
        continue;
    }
    a = 1/(x-7);
    printf("%f", a);
    x++;
}
```

```
    }  
    a = 1/(x-7);  
  
    printf("%f", a);  
  
    x++;  
  
}
```

Arrêt inconditionnel

A l'inverse, il peut être voulu d'arrêter prématurément la boucle, pour une autre condition que celle précisée dans l'en-tête de la boucle. L'instruction *break* permet d'arrêter une boucle (*for* ou bien *while*). Il s'agit, tout comme *continue*, de l'associer à une structure conditionnelle, sans laquelle la boucle ne ferait jamais plus d'un tour !

Dans l'exemple de tout à l'heure, par exemple si l'on ne savait pas à quel moment le dénominateur (x-7) s'annule (bon... OK... pour des équations plus compliquées par exemple) il serait possible de faire arrêter la boucle en cas d'annulation du dénominateur, pour une division par zéro !

```
for (x=1; x<=10; x++) {  
    a = x-7;  
  
    if (a == 0) {  
        printf("Division par 0");  
  
        break;  
    }  
    printf("%f", 1/a);  
  
}
```

Trucs & astuces pertinents trouvés dans la base de connaissances		
27/12 03h01	C/C++ Erreur de segmentation	Langage C
14/09 10h42	Critères de choix d'un langage/framework	Programmation
21/07 11h53	Comment débiter, quel langage?	Langages
09/05 23h00	Où trouver un compilateur C/C++ ?	Langage C
05/05 16h31	Utiliser des accents et autres caractères spéciaux	Programmation
24/08 01h03	Création environnement OpenSSH-CHROOT	Linux
10/04 21h29	Screen	Shell
18/11 15h14	Exécuter un script shell	Shell
Plus d'astuces sur « Langage »		

Discussions pertinentes trouvées dans le forum			
16/02 12h22	[Langage C] Jeu	Programmation	16/11 21h43
05/04 13h09	EOF en langage C	Programmation	28/06 15h45
04/03 00h40	[langage c et c++]	Programmation	31/10 13h08
04/05 14h49	Fonction connect() en langage C	Programmation	04/05 15h06
11/05 10h11	Langage C question debutant	Programmation	10/12 07h39
10/01 01h00	gestion d'erreur langage c?!?	Programmation	13/08 13h17
30/03 09h35	Langage C pointeurs, creation de liste.	Programmation	08/04 00h34
03/09 02h58	[langage C] comment remplacer[for] par [if]	Programmation	05/09 23h07
04/05 16h34	getopt() langage C	Programmation	07/05 14h23
13/08 10h49	langage C	Programmation	11/11 13h29
Discussion en cours		Discussion fermée	Problème résolu
			Plus de discussions sur « Langage »

Logiciels pertinents trouvés dans les téléchargements		
	Dev-C++ - Dev-C++ est un environnement de développement intégré (IDE) en C/C++. Son compilateur est basé sur Mingw de GCC, mais il...	Catégorie: C/C++ Licence: Freeware/gratuit
	Visual Basic Express - Le langage de programmation Visual Basic est historiquement dans les gènes de la société Microsoft. Avec plus de 30.000...	Catégorie: Visual Basic Licence: Freeware/gratuit
	DS Monkey Audio - Filtre APE - Le format MonkeyAudio (extension *.ape) est un format de compression audio lossless (sans pertes) permettant de réduire la...	Catégorie: Plugins audio Licence: Freeware/gratuit
	ICQ - ICQ est un client et un réseau de messagerie instantanée, proposant un certain nombre de fonctionnalités qui lui sont...	Catégorie: Messagerie instantanée Licence: Freeware/gratuit
Plus de logiciels gratuits sur « Langage »		

Produits relatifs			
	Yamaha CRW2200EZ-VK 20x10x40x /	Catégorie: Graveur CD/DVD	Aucun avis
	Sony CMT-CPZ1	Catégorie: Chaîne Hi-Fi	Aucun avis
	Thomson CS704		Aucun avis 265.00 €

Ce document intitulé « [Langage C - Structures conditionnelles](#) » issu de [Comment Ça Marche](#) (www.commentcamarche.net) est mis à disposition sous les termes de la licence [Creative Commons](#). Vous pouvez copier, modifier des copies de cette page, dans les conditions fixées par la licence, tant que cette note apparaît clairement.

[Contribuer à cet article](#)

[Format imprimable](#)

[Glossaire](#) [0-9](#) [A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#)

[A propos](#) | [Conditions générales](#) | [Plan](#) | [Partenaires](#) | [Contact](#) | [Index des marques](#) | ©
