Programmation C pour les débutants

INSTRUCTION BREAK ET CONTINUE EN LANGAGE C

31 Aout 2019 / *M. ESSADDOUKI* / *Langage C* / *89 Visites*

Instruction break

Supposons que nous écrivions un programme pour rechercher un nombre particulier parmi 1000 numéros. À la dixième itération, nous avons trouvé le nombre souhaité. À ce stade, nous ne voulons pas traverser les 990 numéros restants, mais nous voulons que la boucle se termine et continuer avec l'exécution de l'instruction qui suit la boucle. C'est ici que la déclaration de break entre en scène.

Lorsqu'une instruction break est rencontrée dans la boucle, le contrôle du programme quitte immédiatement la boucle et reprend l'exécution avec l'instruction suivant la boucle.

L'instruction break est presque toujours utilisée avec l'instruction **if ... else** à l'intérieur de la boucle.

Exemple 1 :

```c
#include < stdio.h>

int main(void){
    int i;

    for(i=0;i < 10;i++){
        if( i==5 ){
            break; // quitter la boucle
        }
        printf("i = %d \n",i);
    }

    printf("bye");

    return 0;
}
```

```
i = 0
i = 1
i = 2
i = 3
i = 4
bye
```

Dans la 6ème itération, la valeur de i devient 5. La condition (**i == 5**) est vérifiée car elle est vraie. L'instruction **break** est exécutée et le contrôle sort de la boucle for pour exécuter l'instruction qui la suit. S'il n'y avait pas eu de déclaration de **break**, cette boucle aurait été exécutée 9 fois.

La déclaration de break dans une boucle imbriquée

Lorsque l'instruction **break** est utilisée à l'intérieur d'une boucle imbriquée, elle ne provoque la sortie que de la boucle la plus interne.

Exemple 2 :

```c
#include < stdio.h>

int main(void){
    int i,j;

    for(i=0;i < 10;i++){
        printf("i = %d \n",i);
        for(j=0;j< 4;j++){
            if( j==2 ){
                break; // quitter la boucle interne
            }
            printf("j = %d \n",j);
        }

    }

    printf("bye");

    return 0;
}
```

```
i = 0
j = 0
j = 1

i = 1
j = 0
j = 1

i = 2
j = 0
j = 1

bye
```

Instruction continue

L'instruction **continue** ignore l'itération actuelle de la boucle et continue à l'itération suivante.
Lorsque l'instruction **continue** est rencontré dans une boucle, toutes les déclarations après l'instruction **continue** sont omis et la boucle continue avec la prochaine itération.

L'instruction **continue** est presque toujours utilisée avec l'instruction **if ... else** à l'intérieur de la boucle.

> Rappelez-vous toujours que l'instruction **break**, lorsqu'elle est rencontrée, sort de la boucle, mais lorsque l'instruction **continue** est rencontrée, la boucle n'est pas terminée, mais le contrôle est passé au début de la boucle.

Lorsque l'instruction **continue** est rencontrée dans la boucle **while** et **do while**, le contrôle est transféré à la condition de test, puis la boucle continue. alors que dans la boucle **for** lorsque l'instruction continue est trouvée, le contrôle est transféré à l'expression de mise à jour, puis la condition est testée.

Exemple 3 :

```c
#include < stdio.h>

int main(void){
    int i;

    for(i=0;i < 10;i++){
        if( i==5 ){
            continue; // passer à l'itération suivante
        }
        // cette instruction est ignorée lorsque i==5
        printf("i = %d \n",i);
    }

    printf("bye");

    return 0;
}
```

```
i = 0
i = 1
i = 2
i = 3
i = 4

i = 6
i = 7
i = 8
i = 9
bye
```

Exemple 4 :

```c
#include < stdio.h>

int main(void){
    int i,j;

    for(i=0;i < 2;i++){
        printf("i = %d \n",i);
        for(j=0;j< 4;j++){
            if( j==2 ){
                continue; // quitter la boucle interne
            }
            // cette instruction est ignorée lorsque j==2
            printf("j = %d \n",j);
        }

    }

    printf("bye");

    return 0;
}
```

```
i = 0
j = 0
j = 1
j = 3

i = 1
j = 0
j = 1
```

```
j = 3

i = 2
j = 0
j = 1
j = 3

bye
```

Partager ce cours avec tes amis :

← les boucles en c

les fonctions en C →

Rédigé par M. ESSADDOUKI

Learning a new programming language is an easy thing, but the most difficult thing is how to design efficient algorithms for real-world problems, so don't be a programmer, be a problems solver.

Cours Similaires :

Introduction au langage C

25 Aout 2019 / M. ESSADDOUKI / 222

les pointeurs en C

15 Septembre 2017 / M. ESSADDOUKI / 5962

Variables locales et globales en langage C

31 Aout 2019 / M. ESSADDOUKI / 146

les fonctions en C

25 Aout 2019 / M. ESSADDOUKI / 165

Developpement InformatiqueLa première plateforme éducative dans le domaine informatique

ESSAADA, Immeuble 13, Etage 3, Appartement 6, Marrakech,Marrakech-Safi, Maroc (Headquarters)

Téléphone: (Phone Number) (+212) 695 550 379
Email: (Email Address) essaddouki@gmail.com

Pointeurs et tableaux en langage C
M. ESSADDOUKI / *21 Septembre 2019*

Foire aux questions sur la programmation en C - FAQ 2
M. ESSADDOUKI / *20 Septembre 2019*

Foire aux questions sur la programmation en C - FAQ 1
M. ESSADDOUKI / *20 Septembre 2019*

Nouveautés de Java 11
M. ESSADDOUKI / *02 Septembre 2019*

Apprendre Langage C++ - Guide complet pour les débutants
24 Septembre 2019

Langage Scilab - Guide complet pour les étudiants de CPGE
15 Septembre 2019

Cours Java pour les débutants
02 Septembre 2019

Langage SQL - Guide complet pour les débutants
31 Aout 2019