Leçon 3 : Les structures itératives

SANE PIERRE-MARIE ARNAUD : Enseignant - Chercheur

AYIKPA KACOUTCHY JEAN : Enseignant - Chercheur

KONE MOUSSA GBONGUE : Enseignant - Chercheur

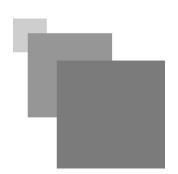


Table des matières

I - Objectifs	3
II - Pré requis	4
III - INTRODUCTION	5
IV - I- LA STRUCTURE while	6
V - Exercice : EXERCICE N° 1	7
VI - II- LA STRUCTURE do while	9
VII - Exercice : EXERCICE N° 2	11
VIII - III- LA STRUCTURE for	13
IX - Exercice : EXERCICE N° 3	15
X - IV- APPLICATION	16

Object ifs

Ce cours devra vous permettre :

- $\bullet\,$ D'utiliserau mieux les structures itératives en langage C
- $\bullet\,$ De distinguer les différentes structures itératives en langage C

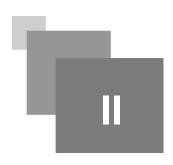
Pré requis



Avant de suivre cette leçon vous devez :

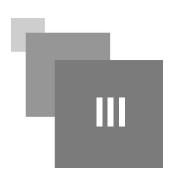
- Avoir pour acquis la connaissance de l'algorithme ;
- Avoir survie la leçon 1 : Concepts généraux du Langage C
- Avoir survie la leçon 2 : Les structures conditionnelles

INTRODUCTION



Les structures itératives, encore appelées structures répétitives font partie des structures de contrôle en programmation et en langage C en l'occurrence. Elles permettent d'exécuter plusieurs fois un même bloc d'instructions en tenant compte d'un critère d'exécution. Ainsi, il existe trois types de structures itératives en langage C que sont : la structure while, la structure do.. while et la structure for.

I- LA STRUCTURE while



1

Définition : 1- Définition

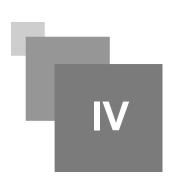
La structure *while* simple est une structure qui exécute plusieurs fois un bloc d'instructions tant que la condition posée est vérifiée.

```
2- Syntaxe
Initialisation
while(condition)
{
Bloc d'instructions;
variation;
}
```

Exemple : 3- Exemple

```
\begin{tabular}{ll} \#include < stdio.h > // & elle permet d'utiliser les fonctions printf() et scanf() \\ \#include < conio.h > // & elle permet de manipuler la console windows (par exemple effacer l'écran, changer la couleur de police de console etc.) \\ main() \\ \{ & Int \ val1; \ // \mbox{Déclaration de la variable val1} \\ val1=1; \ // \mbox{initialisation de la variable ( lnitialisation )} \\ while (val1<=10) \ // \mbox{vérification de la condition} \\ \{ & printf("\%d \ /",val1); \ // \mbox{affichera 1 } | 2 \ | 3 \ | 4 \ | 5 \ | 6 \ | 7 \ | 8 \ | 9 \ | 10 \ | \\ val1++; \ // \mbox{incrémentation de la variable ( la variation )} \\ \} \\ \} \\ \end{tabular}
```

Exercice: EXERCICE N° 1



Exercice: EXO 1

```
1- Quelle est la dernière valeur retournée par v2 dans le programme défini ci-dessous ? #include<stdio.h>
main()
{
int \ v1,v2;
v1=1;
while(v1>=0)
{
v2=5;
v1--;
v2++;
}
}
O a. 8
O b. 6
O c. 5
```

Exercice: EXO 2

The second second

II- LA STRUCTURE do . . while



1

Définition : 1- Définition

Contrairement à la structure *while*, la structure *do . . while* exécute le bloc d'instructions avant que la condition ne soit posée ce qui a pour conséquence, l'exécution du bloc d'instructions au moins une fois même si la condition posée n'est pas vérifiée.

```
2- Syntaxe
do
{
Bloc d'instructions1;
}
while( condition );
```

E

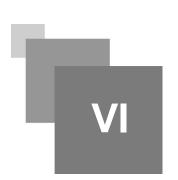
main()

Exemple : 3- Exemple

```
Exemple 1
#include < stdio.h > // elle permet d'utiliser les fonctions printf() et scanf()
#include < conio.h > // elle permet de manipuler la console windows (par exemple effacer l'écran,
changer la couleur de police de console etc.)
main()
{
Int val1; //Déclaration de la variable val1
val1=1; ///initialisation de la variable ( lnitialisation )
do
printf("%d /",val1); //affichera 1 |2 |3 |4 |5 |6 |7 |8 |9 |10 |
val1++; //incrémentation de la variable
while(val1<=10) //vérification de la condition
Exemple 2:
#include < stdio.h > // elle permet d'utiliser les fonctions printf() et scanf()
#include < conio.h > // elle permet de manipuler la console windows (par exemple effacer l'écran,
changer la couleur de police de console etc.)
```

```
{ Int val1; //Déclaration de la variable val1 val1=11; //initialisation de la variable do { printf("\%d \mid ",val1); //affichera 10 | val1++; //incrémentation de la variable } while(val1<=10) //vérification de la condition }
```

Exercice: EXERCICE N° 2



Exercice: EXO 1

```
1- Combien d'itérations effectue ce programme ci-dessous ?
\#include {<} stdio.h{>}
main()
{
int v1, v2;
v1=4;
do
{
v2=3;
v2++;
v1--;
}
while(v1>=1);
O a. 5
O b. 6
O c. 4
```

Exercice: EXO 2

```
2- Combien d'itérations effectue ce programme ci-dessous ?
\#include {<} stdio.h {>}
main()
int\ v1, v2;
v1=4;
do
{
v2=3;
v2++;
v1--;
}
while(v1>1);
}
O a. 5
O b. 6
O c. 3
```

III- LA STRUCTURE for



1

Définition : 1- Définition

Contrairement aux structures while et do.. while, la structure for est une structure itérative qui lors de son utilisation, initialise la variable, définit la condition d'arrêt et le compteur.

2- Syntaxe for(initialisation; condition; compteur) { Bloc d'instructions;

Remarque

- *Initialisation* : ce paramètre est exploité une seule fois (juste avant la première itération) car il permet d'initialiser la variable qui sera vérifiée dans le paramètre suivant (à savoir la condition) avant l'exécution du bloc d'instructions.
- Condition : ce paramètre vérifie la condition d'arrêt de la boucle. Avant la première itération, cette condition est vérifiée juste après l'initialisation de la variable. Après cela, la condition est toujours vérifiée après l'exécution du compteur. Ainsi la condition d'arrêt précède toujours l'exécution du bloc d'instructions.
- Compteur : ce paramètre est toujours sollicité après l'exécution du bloc d'instructions et juste avant que la condition ne soit vérifiée.

F Exemple : 3- Exemple

```
\begin{tabular}{ll} \#include < stdio.h > // & elle permet d'utiliser les fonctions printf() et scanf() \\ \#include < conio.h > // & elle permet de manipuler la console windows (par exemple effacer l'écran, changer la couleur de police de console etc.) \\ main() \\ \{ & Int \ val1; \ // \text{Déclaration de la variable val1} \\ for(val1=1;val1<=10;val1++) \ // \text{initialisation de val1, condition et incrémentation} \\ \{ & printf("\%d \ /",val1); \ // \ // \text{affichera 1 } |2 \ |3 \ |4 \ |5 \ |6 \ |7 \ |8 \ |9 \ |10 \ | \\ \} \\ \} \\ \end{tabular}
```

Explication : La variable val1 est initialisée à 1 dans un premier temps, la condition est ensuite vérifiée avant que l'instruction ne soit exécutée. Juste après, val1 est incrémentée et la condition est à nouveau vérifiée avant l'exécution de l'instruction et le cycle se poursuit jusqu'à ce que val1 soit

supérieur à 10 avant de sortir de la boucle.

Exercice: EXERCICE N° 3



Exercice: EXO 1

```
1- Combien d'itérations effectue ce programme ci-dessous ? #include<stdio.h>
main()
{
int\ v1,v2;
v1=4;
for(i=0\ ;i<11\ ;i++)
{
v2=3;
v2++;
v1--;
}
}
orrow a=11
orrow b=10
orrow c-9
```

IV- APPLICATION



1- Énonce

Écrire un programme en langage C qui contraint l'utilisateur à saisir un nombre pair avant l'affichage du triple de ce nombre.

2- Correction

```
#include<stdio.h> // elle permet d'utiliser les fonctions printf() et scanf()
#include<conio.h> // elle permet de manipuler la console windows (par exemple effacer l'écran, changer la couleur de police de console etc.)

main()

{
    int nb; // déclaration de la variable nb
    do
    {
        printf("Saisissez un nombre pair svp! | n"); // affichera : Saisissez un nombre pair svp!

scanf("%d",&nb); // récupère la valeur saisie
}

while(nb%2!=0); // vérifie si le nombre est impair

printf("\nLe triple de %d est: ",nb); // affichera : Le triple de (contenu de nb ) est:

nb*=3;

printf("%d",nb); // affichera : la nouvelle valeur de la variable nb
}
```