



Pr. Youness KHOURLIFI, PhD en Informatique
Professeur à la Faculté Polydisciplinaire – Khouribga –
Université Sultan Moulay Slimane – Béni Mellal –
Consultant IT : SQL 2016 Database Administration, Core
Infrastructure 2016, Azure Solutions Architect Expert,
Data Analyst Associate, Ingénieur DevOps.
y.khourdifi@usms.ma

TECHNIQUES DE PROGRAMMATION EN C



Chapitre

4

Les boucles en algorithmique et en langage C

IV. 1. Les boucles en algorithmique

3

□ Les boucles en algorithmique

Si pour résoudre un certain problème, on est obligé de répéter une instruction ou bien un ensemble d'instructions un certain nombre de fois, on parle de **structure en boucle** ou encore de **structure itérative** ou **répétitive**.

Il existe en gros trois façon pour exprimer cette répétition :

- La structure itérative « pour »
- La structure itérative « Tant que »
- La structure itérative « répéter Tant que »

IV. 1. Les boucles en algorithmique

4

■ La structure itérative « pour »

Pour cette variante, on déclare un compteur « Identificateur » dont on doit connaître la valeur initiale et la valeur finale,

La syntaxe algorithmique de cette structure est la suivante :

```
Identificateur : entier
```

```
Pour identificateur allant de « Val initiale » à « Val finale » par pas de  
« pas » {traitement}
```

```
Fin pour
```



- Le nombre de traitement de la boucle = (Valeur Finale – Valeur Initiale) / pas
- On utilise cette structure quand on connaît le nombre de passage dans la boucle

IV. 1. Les boucles en algorithmique

5

❑ Exemple 1 : Calculer la somme des N premiers entiers

Algorithme : Calcul de la somme des N premiers entiers avec la boucle pour

```
//Déclaration
I, N, somme : entiers
// préparation du traitement
Ecris('donner un entier N positif')
Lis N //traitement
somme = 0 //initialisation de somme
Pour I allant de 1 à N pas de 1
    somme = somme + I
// Edition des résultats
Ecris ('somme = ', Somme)
Ecris ('Fin')
```

Exemple d'exécution

donner un entier N positif

10

Somme = 55

Fin

IV. 1. Les boucles en algorithmique

6

❑ Exemple 2 : Afficher la table de multiplication d'un entier

```
Algorithme : Afficher la table de multiplication d'un entier
// Déclaration
I,N : entiers
// préparation du traitement
  Ecris('donner un entier pour lequel vous voulez la table')
  Lis N
//traitement

  Pour I allant de 0 à 9  pas de 1
    Ecris(I,'*',N,'=',I*N)

// Edition des résultats
  Ecris ('Fin')
```

IV. 1. Les boucles en algorithmique

7

❑ **Exemple 3 :** Ecrire un algorithme qui permet d'afficher N fois de suite le mot 'phrase1', puis une fois le mot 'phrase2'.

Algorithme : Ecrire un algorithme qui permet d'afficher N fois de suite le mot 'phrase1', puis une fois le mot 'phrase2'.

```
// Déclaration
```

```
N, I : entiers
```

```
    // préparation du traitement
```

```
    Ecris('donner un entier positif N')
```

```
Lis N
```

```
    //traitement
```

```
Pour I allant de 1 à N pas de 1
```

```
{
```

```
    Ecris(' phrase1 ')
```

```
}
```

```
Ecris(' phrase2 ')
```

```
// Edition des résultats
```

```
Ecris ('Fin')
```

IV. 1. Les boucles en algorithmique

8

□ La structure itérative « Tant que »

Pour cette structure, il faut en général initialiser le compteur et préciser la condition sur ce compteur pour poursuivre le traitement.

Quand cette condition n'est plus vérifiée, le traitement sera arrêté.

Il ne faut pas oublier de préciser le pas de variation du compteur.

La syntaxe algorithmique de cette structure est la suivante :

Compteur: type du compteur

Compteur = Val initial

TantQue (condition sur le compteur) **Faire** { traitement + variation compteur} **FinTantQue**

IV. 1. Les boucles en algorithmique

9

❑ **Exemple 4** : Calculer la somme des N premiers entiers .

Première méthode

Algorithme : calcul de la somme des N premiers entiers avec la boucle tant que

I, N, somme : entiers // Déclaration

Ecris('donner un entier N positif') // préparation du traitement

Lis N //traitement

somme = 0 //initialisation de somme

I=0 //initialisation du compteur

Tant que (I<N) faire

{

I = I + 1

 somme = somme + I

}

Ecris ('somme = ',Somme) // Edition des résultats

Ecris ('Fin')

IV. 1. Les boucles en algorithmique

10

Deuxième méthode

Algorithme : calcul de la somme des N premiers entiers avec la boucle tant que

I, N, somme : entiers // *Déclaration*

Ecris(''donner un entier N positif'') // *préparation du traitement*

Lis N // *traitement*

somme = 0 // *initialisation de somme*

I=0 // *initialisation du compteur*

Tant que (I<=N) faire

{

somme = somme + I

I = I + 1

}

Ecris (''somme = ',Somme) // *Edition des résultats*

Ecris (''Fin'')

IV. 1. Les boucles en algorithmique

11

□ **Exemple 5** : Afficher la table de multiplication d'un entier. On ne tiendra compte de N que s'il est > 0 .

Algorithme : Afficher la table de multiplication d'un entier. On ne tiendra compte de N que s'il est > 0 .

```
I,N : entiers // Déclaration
```

```
Ecris('donner un entier>0') // préparation du traitement
```

```
Lis N
```

```
Tant que (N<=0)
```

```
{
```

```
Ecris('donner un entier>0 pour lequel vous voulez la table')
```

```
Lis N //traitement
```

```
}
```

```
    I=0 //initialisation du compteur
```

```
Tant que (I<=9) faire
```

```
Ecris(I,'*',N,'=',I*N) // Edition des résultats
```

```
Ecris ('Fin')
```

IV. 1. Les boucles en algorithmique

12

□ **Exemple 6** : Ecrire un algorithme qui permet d'afficher N fois de suite le mot 'phrase1', puis une fois le mot 'phrase2'.

```
Algorithme : d'afficher N fois de suite le mot 'phrase1', puis une fois le mot
'phrase2'.

I,N : entiers      // Déclaration

    Ecris('donner un entier>0') // préparation du traitement
Lis N
Tant que (N<=0)
{ Ecris('donner un entier>0 pour lequel vous voulez l'affichage la suite de mot')
Lis N } //traitement

    I=0      //initialisation du compteur
Tant que (I<N) faire // si on initialise le compteur à 1 la condition serait I<=N
{ Ecris('phrase1')
I=I+1 }
Ecris('phrase2') // Edition des résultats
Ecris ('Fin')
```

IV. 1. Les boucles en algorithmique

13

□ La structure itérative « répéter Tant que »

Pour cette structure aussi, il faut en général initialiser le compteur et préciser la condition sur ce compteur pour poursuivre le traitement. Quand cette condition n'est plus vérifiée, le traitement sera arrêté. Il ne faut pas oublier de préciser le pas de variation du compteur.

La syntaxe algorithmique de cette structure est la suivante :

```
Compteur: type du compteur
```

```
Compteur = Val initial
```

```
    Répéter {Traitement} TantQue (condition)
```



On utilise cette structure quand on ne connaît pas le nombre de passage dans la boucle mais que l'on va y passer au moins une fois.

IV. 1. Les boucles en algorithmique

14

□ Exemple 7 : Calculer la somme des N premiers entiers.

```
Algorithme : calcul de la somme des N premiers entiers avec la
boucle pour
I, N, somme : entiers // Déclaration
Ecris('donner un entier N positif') // préparation du traitement
Lis N //traitement
I=0 // initialisation du compteur I
somme = 0 //initialisation de somme
Répéter
{I=I+1
  somme =somme +I
}
TantQue (I<N)
Ecris ('somme = ',Somme) // Edition des résultats
Ecris ('Fin')
```

IV. 1. Les boucles en algorithmique

15

□ Exemple 8 : Afficher la table de multiplication d'un entier.

```
Algorithme : Afficher la table de multiplication d'un entier
I,N : entiers    // Déclaration
Ecris('donner un entier N positif') // préparation du traitement
Lis N           //traitement
                I=0    // initialisation du compteur I
Répéter
{
    Ecris(I,' '*',' ',N,' '=',' ',I*N)
    I=I+1
}
TantQue (I<=9) // Edition des résultats
Ecris ('Fin')
```

IV. 1. Les boucles en algorithmique

16

□ **Exemple 9 :** Ecrire un algorithme qui permet d'afficher N fois de suite le mot 'phrase1', puis une fois le mot 'phrase2'.

```
Algorithme : Ecrire un algorithme qui permet d'afficher N fois de suite
le mot 'phrase1', puis une fois le mot 'phrase2'.
I,N : entiers    // Déclaration
Ecris('donner un entier N positif')    // préparation du traitement
Lis N    //traitement
        I=0    // initialisation du compteur I
Répéter
{
    Ecris('phrase1')
    I=I+1
}
TantQue (I<N)
Ecris('phrase2')    // Edition des résultats
Ecris ('Fin')
```


IV. 1. Les boucles en algorithmique

17



- ☐ Avec la structure pour, le compteur est incrémenté automatiquement, il suffit de préciser le pas.
- ☐ Par contre avec les deux autres structures, le compteur doit être incrémenté (ou décrémenté) lors du traitement sans oublier de l'initialiser au départ.

IV. 2. Les boucles en langage C

18

□ La structure itérative « for »

La syntaxe de cette structure est la suivante :

```
int id ;  
for(id=valeur initiale ; condition qui fixe la valeur Finale de id ; id=id + pas)  
{traitement à faire ;}
```

Exemple :

si on veut afficher N fois le mot bonjour, on écrit :

```
int I;  
for (I=1 ; I<=10 ; I=I+1)  
printf("  Bonjour \n");
```

IV. 2. Les boucles en langage C

19

L'exécution se fait de la manière suivante :

I vaut 1

La condition sur I est vari

On affiche le mot bonjour et on revient à la ligne

On prend $I=I+1$

La condition sur I est vari

On affiche le mot bonjour et on revient à la ligne

On prend $I=I+1$

Et ainsi de suite jusqu'à ce que la condition sur I ne soit plus vérifiée, on arrête le traitement, c'est à dire qu'on n'affiche plus le mot Bonjour.

Les boucles en algorithmique

20

❑ **Exemple 10** : Calculer la somme des N premiers entiers en montant en valeurs du compteur.

```
/*calcul de la somme des N premiers entiers avec la boucle for
en montant en valeurs du compteur */
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int I,N,somme; //declaration
    printf("donner un entier N>0\n"); //preparation du traitement
    scanf("%d",&N);
    somme=0; // initialisation de somme
    for(I=1;I<=N;I++) //traitement
        somme=somme+I;
        printf("votre somme vaut somme= ");
        printf("%d \n", somme);
    printf("FIN \n");
}
```

```
donner un entier N>0
6
votre somme vaut somme= 21
FIN
```

Les boucles en algorithmique

21

❑ Exemple 11 : Afficher la table de multiplication d'un entier.

```
/* Table de multiplication d'un entier avec la boucle for */
#include <stdio.h>
#include <stdlib.h>
int main()
{ int I,N; //declaration
printf("donner un entier N>0\n"); //preparation du traitement
scanf("%d",&N); //traitement
for(I=0;I<=9;I++)
{
    printf("%d",I);
    printf("*");
    printf("%d",N);
    printf("=");
    printf("%d \n", I*N);
}
printf("FIN \n");
}
```

```
donner un entier N>0
7
0*7=0
1*7=7
2*7=14
3*7=21
4*7=28
5*7=35
6*7=42
7*7=49
8*7=56
9*7=63
FIN
```

Les boucles en algorithmique

22

□ Exemple 12 : Calculer $N!$ avec la boucle for

```
/*calcul de N! avec la boucle for */
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int I,N; //declaration
    int FACTN;
    printf("donner un entier N>0\n"); //preparation du traitement
    scanf("%d",&N); //traitement
    FACTN=1; //initialisation
    for(I=2;I<=N;I++)
        FACTN=FACTN*I;
        printf("%d \n", FACTN);
    printf("FIN \n");
}
```

Les boucles en algorithmique

23

❑ Exemple 12 : Calculer $N !$ avec la boucle for

```
donner un entier N>0
4
24
FIN

-----
Process exited after 1.992 seconds with return value 0
Appuyez sur une touche pour continuer... _
```

```
donner un entier N>0
20
-2102132736
FIN

-----
Process exited after 1.218 seconds with return value 0
Appuyez sur une touche pour continuer...
```



On peut remarquer que le résultat de cette exécution est aberrant, puisqu'il nous donne une valeur négative pour le factoriel de l'entier 20. Ceci peut s'expliquer par un dépassement des bornes de l'intervalle permis pour les entiers. Lorsqu'un tel problème risque de se poser, il est conseillé de déclarer la variable dans laquelle on veut placer le résultat d'un tel calcul comme variable réelle.

Les boucles en algorithmique

24

□ **Exemple 13** : Calculer $N! / ((P!) * ((N-P)!))$

```
donner un entier N>0
4
donner un entier P>0 et <N
2
FACTN=24.000000
FACTP=2.000000
FACTNMP=2.000000
CNP=6.000000
FIN

-----
Process exited after 10.91 seconds with return value 0
Appuyez sur une touche pour continuer...
```


Les boucles en langage C

25

□ La structure itérative « while »

La syntaxe de cette structure est la suivante :

```
while (condition) {traitement;}
```

Exemple :

si on veut afficher N fois le mot bonjour, on écrit :

```
int I,N;  
scanf( "%d", &N);  
I=1;    //initialisation du compteur  
while (I<=N)  
{printf("  Bonjour \n");  I=I+1 ;}
```

Les boucles en algorithmique

26

❑ Exemple 14 :

Calculer la somme des N premiers entiers en montant en valeurs du compteur.

❑ Exemple 15 :

Calculer la somme des N premiers entiers en descendant en valeurs du compteur.

❑ Exemple 16 :

Afficher la table de multiplication d'un entier

❑ Exemple 17 :

Calculer $N !$ avec la boucle while

❑ Exemple 18 :

Calculer $N ! / (P !)((N-P) !)$ avec la boucle while

Les boucles en langage C

27

❑ La structure itérative « do while »

La syntaxe en langage C de cette structure est la suivante :

```
Do traitement while (condition) ;
```

On utilise cette structure quand on ne connaît pas le nombre de passage dans la boucle mais que l'on va y passer au moins une fois.

Exemple :

si on veut afficher N fois le mot bonjour, on écrit :

```
int I,N;
scanf( "%d", &N);
I=1;    //initialisation du compteur
do
{printf("  Bonjour \n");  I=I+1 ;}
while (I<=N)  ;
```

Les boucles en algorithmique

28

❑ Exemple 19 :

Calculer la somme des N premiers entiers en montant en valeurs du compteur.

❑ Exemple 20 :

Calculer la somme des N premiers entiers en descendant en valeurs du compteur.

❑ Exemple 21 :

Afficher la table de multiplication d'un entier

❑ Exemple 22 :

Calculer $N!$ avec la boucle do while

❑ Exemple 23 :

Calculer $N!/(P!)((N-P)!)!$ avec la boucle do while

Les boucles en langage C

29

□ Les structures de contrôle avec les boucles

Dans certains cas, on peut avoir besoin de mélanger différentes structures pour arriver à résoudre un problème donné. On peut aussi dans certains cas avoir besoin d'utiliser des opérations logiques. Pour illustrer cela nous donnons l'exemple suivant :

Exemple 24:

afficher les entiers compris entre deux bornes données au clavier (bornes comprises) en utilisant la structure de contrôle ||

Les boucles en langage C

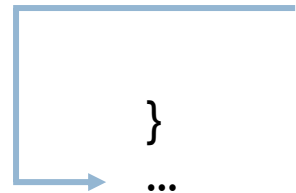
30

□ L'instruction **break**

L'instruction **break** permet d'arrêter le déroulement d'une boucle, et de passer à l'instruction qui suit cette boucle.

Syntaxe :

```
boucle(...) {  
    ...  
    break;  
    ...  
}  
...
```



Les boucles en langage C

31

Exemple 25:

Ecrire un programme qui calcule la somme d'un maximum de 10 nombres entrés par l'utilisateur, si un nombre négatif est entré, la boucle se termine.

```
/*Calcule la somme d'un maximum de 10 nombres */
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int N, S, i;
    S = 0;
    for (i=1;i<=10; i++) {
        printf("Entrer N%d : ",i);
        scanf("%d", &N);
        if (N<0)
            break;
        S = S+N;
    }
    printf("La somme est : %d",S);
return 0;
}
```

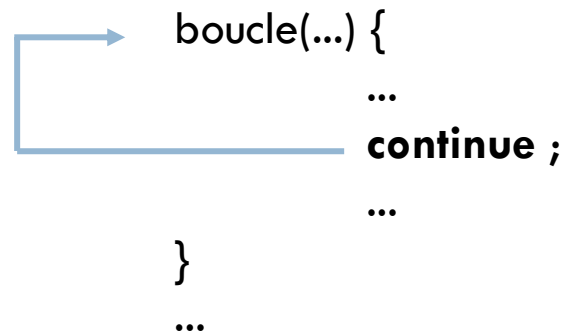
Les boucles en langage C

32

□ L'instruction continue

L'instruction **continue** permet d'ignorer l'itération actuelle de la boucle et de passer à l'itération suivante.

Syntaxe :



```
boucle(...) {  
    ...  
    continue ;  
    ...  
}
```


Les boucles en langage C

33

Exemple 26:

Ecrire un programme qui calcule la somme d'un maximum de 10 nombres entrés par l'utilisateur, si un nombre négatif est entré, la boucle ignore ce nombre.

```
/*Calcule la somme d'un maximum de 10 nombres */
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int N, S, i;
    S = 0;
    for (i=1;i<=10; i++) {
        printf("Entrer N%d : ",i);
        scanf("%d", &N);
        if (N<0)
            continue;
        S = S+N;
    }
    printf("La somme est : %d",S);
    return 0;
}
```


Les boucles en langage C

34

□ L'instruction goto

L'instruction **goto** permet de se repositionner sur une autre section de code exécuter, introduite par une étiquette, au lieu de poursuivre une exécution séquentielle.

Syntaxe :



```
...  
goto etiquette;  
..  
..  
etiquette :  
..  
...
```

Les boucles en langage C

35

Exemple 27:

Sans utiliser de boucles, écrivez un programme qui demande un nombre entre 1 et 10, jusqu'à ce que la réponse soit appropriée.

```
// Titre : un programme qui demande un nombre entre 1 et 10
// Auteur : Y. Khourdifi
// Date : 17/12/2021 //30/11/2022

#include <stdio.h>
#include <stdlib.h>
int main()
{
    int N;

    debut :
    printf("Entrer un nombre entre 1 et 10 :");
    scanf("%d",&N);
    if (N<1 || N>10)
        goto debut;
    printf("Bravo, vous avez saisi un nombre compris entre 1 et 10");

    return 0;
}
```