Chapitre 3

Les Structures itératives

I. Introduction

Il est souvent nécessaire d'exécuter plusieurs fois une action ou un groupe d'actions, non pas indéfiniment mais un certain nombre de fois (pas forcement connu à l'avance) : c'est la notion de boucle.

2. La structure " while "

Syntaxe :

while(<Condition>)

{

<Bloc d'instructions>;

}

1. Tant que la <Condition> fournit une valeur différente de zéro, le <bloc d' instructions > est exécuté.
2. Si la <Condition> fournit la valeur zéro, l'exécution continue avec l'instruction qui suit le bloc d'instructions.
3. Le <bloc d' instructions > est exécuté zéro ou plusieurs fois.

Exemple1

```
int i = 0 ;
while (i<5)
{ printf("%d\t ",i) ;
i++ ;
}
```

Résultat:

```
0	1	2	3	4
```

Exemple2

```
int i = 0 ;
while (i<5)
printf("%d\t ",++i) ;
```

Résultat:

```
1	2	3	4	5
```

Exemple3

```
int i = 5 ;
while (i)
printf("%d\t ",i--) ;
```

Résultat:

```
5	4	3	2	1
```

3. La structure " for "

Syntaxe

```
for(<expression-I>; <expression-C>; <expression-R>)
{
   <Bloc d'instructions>;
}
```

Est équivalente à :

```
<expression-I>;
while(<expression-C>)
{
   <Bloc d'instructions>;
   <expression-R>;
}
```

- expression-I : une expression initialisant les variables de contrôle qu'il faut initialiser avant d'entrer dans la boucle.
- expression-C : la condition de bouclage.
- expression-R : une expression permettant de réinitialiser (incrémentation, décrémentation) les variables de contrôles utilisées.

Fonction de l'instruction "for"

1. Exécuter les instructions initiales de "for" (expression-I).
2. Evaluer la condition : (expression-C)

- Faux (0) : branchement à l'instruction qui suit la fin de la boucle "for".
- Vrai (1) :

  (a) Exécution du <Bloc d'instructions>.

  (b) Exécution de la liste des instructions de réitération (expression-R).

  (c) Revenir à (2)

3. Fin de l'instruction "for".

Exemple1

```
int i ;
for( i=0; i<=10; i++)
printf("Le carré de %d est %d\n ",i,i*i) ;
```

Exemple 2

```
/* Calcul de la somme de 1+2+3 ... +100*/
int i ,S;
for( i=1,S=0 ;i<=100 ;i++)
   S+=i ;
printf("La somme de 1 à 100 = %d\n ",S) ;
```

4. La structure "do ... while"

Syntaxe :

```
do
{
  <Bloc d'instructions1>;
}while (<Condition>);
```

1. La boucle "do-while" teste sa condition après exécution du <Bloc d'instructions>.
2. Le <Bloc d'instructions> est éxécuté au moins une fois et aussi longtemps que la <Condition> fournit une valeur différente de zéro.
3. En pratique, la structure "do-while" n'est pas si fréquente que while ; mais dans certains cas, elle fournit une solution plus élégante. Une application typique de "do-while" est la saisie de données qui doivent remplir une certaine condition.

Exemple1

```
int N;
do
{
printf(" Donner un nombre entre 1 et 10 :");
scanf("%d",&N) ;
}while(N<1 || N>10);
```

Exemple2 :

```
#include<stdio.h>
void main()
{
int n ,div;
printf(" Donner le nombre à diviser :");
scanf("%d",&n) ;
do
{
  printf(" Entrer le diviseur :");
  scanf("%d",&div) ;
}while( !div);
printf("%d/%d = %f\n",n,div,(float)n/div);
}
```

5. Choix de la structure itérative

1. Si le bloc d'instructions ne doit pas être exécuté si la condition est fausse, alors utilisez "while" ou "for".
2. Si le bloc d'instructions doit être exécuté au moins une fois, alors utilisez "do...while".
3. Si le nombre d'exécutions du bloc d'instructions dépend d'une ou de plusieurs variables qui sont modifiées à la fin de chaque répétition, alors utilisez "for".
4. Le choix entre "for" et "while" n'est souvent qu'une question de préférence ou d'habitudes.