

Chapitre 5: Structures et instructions de contrôle en C.

contrôle en C

5. Structures et instructions de contrôle en C.

5.1. Les instructions de contrôle (de branchement conditionnel)

- ❑ On appelle structure de contrôle toute instruction qui permet de contrôler le fonctionnement d'un programme.
- ❑ Parmi les structures de contrôle, on distingue :
 - ❑ structures de sélection et
 - ❑ Instruction de branchement conditionnel : **if ..else**
 - ❑ Instruction de branchement multiple: **switch**
 - ❑ structures répétitives (boucles)
 - ❑ L'instruction : **for**,
 - ❑ L'instruction : **while** et **do .. while**

5. Structures et instructions de contrôle en C.

5.1. Les instructions de contrôle (de branchement conditionnel)

a) L'instruction **if else**: Dans certain programme, où on désire choisir entre deux instructions selon une condition, on utilise la structure alternative **if**.

Syntaxe

```
if (condition)  
    instruction1  
else  
    instruction2
```

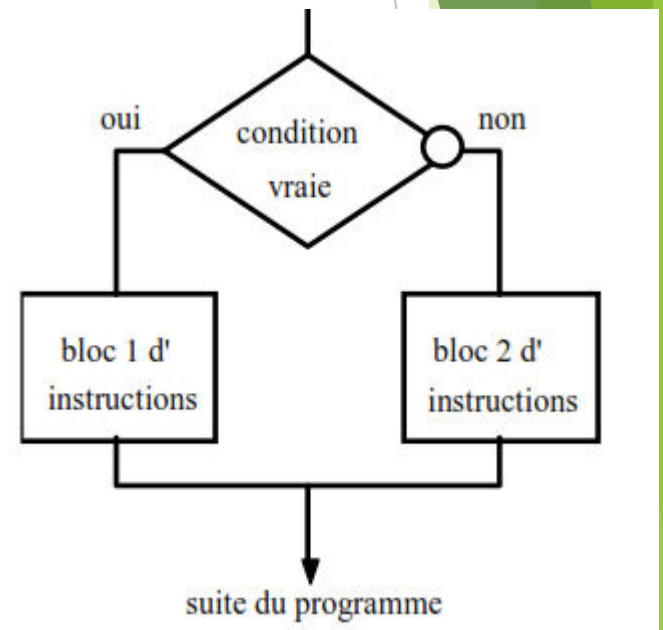
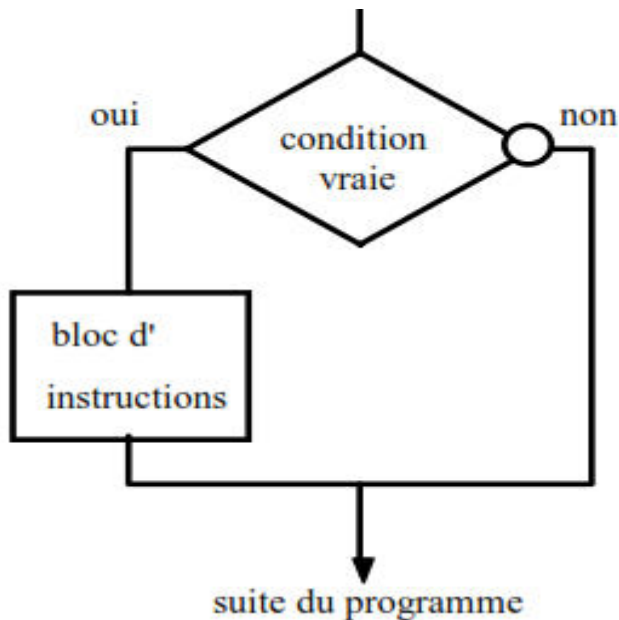
avec condition quelconque, instruction1 et instruction2 sont soit:

- Simples (terminées par un point-virgule;)
- Blocs (délimités par { et })
- Instructions structurés (boucles).

5. Structures et instructions de contrôle en C.

5.1. Les instructions de contrôle (de branchement conditionnel)

a) L'instruction if et if...else



* Tout ce qui est zéro 0 ('\0' 0 0.0000 NULL) est faux

* Tout ce qui est != de 0 (1 '\0' 0.0001 1.34) est vrai

```
if(32) printf("ceci sera toujours affiche\n");
```

```
if(0) printf("ceci ne sera jamais affiche\n");
```

5. Structures et instructions de contrôle en C.

5.1. Les instructions de contrôle (de branchement conditionnel)

Exemple1: if sans else	Exemple2: if avec else	Exemple 3 : if imbriquée
<pre>int a =3, b=7, max ; max=a ; if (max < b) max = b ; printf("%d \n", max) ;</pre>	<pre>int a = -3, vAbs; if (a > 0) vAbs= a ; else vAbs= -a; printf("%d\n", vAbs); ou vAbs = (a > 0) ? a : -a</pre>	<pre>int a = -3, b= 7, c= 2 , max; if (a > b) if (a > c) max = a ; else max = c; else if (b > c) max = b ; else max = c;</pre>

`if (delta != 0) <=> if (delta)`

`if (delta == 0) <=> if (!delta) /* == pas = /*`

5. Structures et instructions de contrôle en C.

5.1. Les instructions de contrôle

- b) L'instruction **switch** : On l'appelle aussi instruction d'aiguillage, elle teste si une *expression* prend une valeur parmi une suite de *constantes* et effectuer le branchement correspondant si c'est le cas:

Syntaxe:

switch (expression)

{ case constante1: [suite_instructions1]

case constante2: [suite_instructions2]

.....

case constanten : [suite_instructionsn]

[default : suite_instructions]

}

5. Structures et instructions de contrôle en C.

5.1. Les instructions de contrôle

b) L'instruction switch : Le fonctionnement de switch est le suivant

- ❑ *expression* est évaluée ;
- ❑ s'il existe un *énoncé case* avec une constante qui égale la valeur de **expression**, le contrôle est transféré à l'instruction qui suit cet énoncé;
- ❑ si un tel case n'existe pas, et si énoncé default existe, alors le contrôle est transféré l'instruction qui suit l'énoncé *default* ;
- ❑ si la valeur de *expression* ne correspond aucun *énoncé case* et s'il n'y a pas d'énoncé *default*, alors aucune instruction n'est exécutée.
- ❑ *constantei* une expression constante entière (*char* sera accepté car il sera converti en *int*).
- ❑ *Suite_instructions*: séquence d'instructions. Ce qui est entre crochets [] est facultatif.
- ❑ L'usage du **break** dans l'instruction switch permet de **quitter** le switch.

5. Structures et instructions de contrôle en C.

5.1. Les instructions de contrôle

```
#include<stdio.h>
int a ;
int main(){
printf ("saisir un entier s'il vous plaît : ") ;
scanf ("%d ",&a) ;
switch(a){
    case 1 : printf ("lundi") ;break ;
    case 2 : printf("mardi") ; break ;
    case 3 : printf("mercredi") ; break ;
    case 4 : printf("jeudi") ; break ;
    case 5 : printf("vendredi") ; break ;
    case 6 : printf("samedi") ; break ;
    case 7 : printf("dimanche") ; break ;
    default :printf("incorrecte") ; break ;
}
return 0 ;
}
```


5. Structures et instructions de contrôle en C.

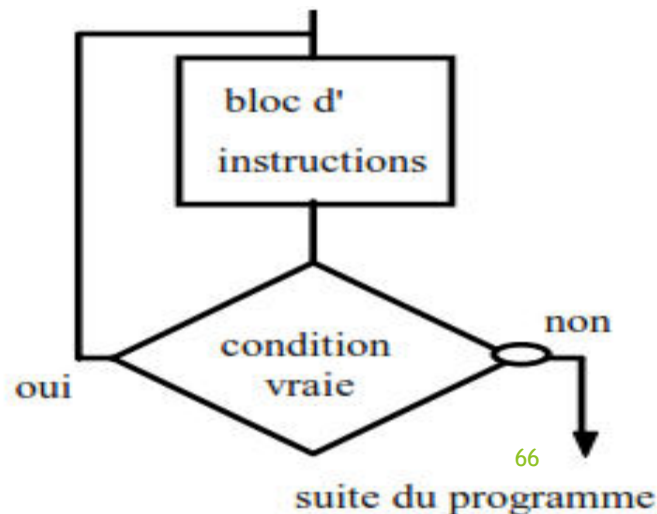
5.2. Les instructions structurés(Boucles):

Dans certain programme, où on désire répéter un bloc instructions plusieurs fois, on utilise les boucles.

a) Boucle do.....while

La syntaxe de la boucle **do ... while** a la forme suivante:

```
do  
{  
  bloc d'instructions  
} while(condition) ;
```



5. Structures et instructions de contrôle en C.

5.2. Les instructions structurés(Boucles):

a) Boucle do.....while

Exemple 1 : $s = 1 + 2 + 3 + \dots + 10$

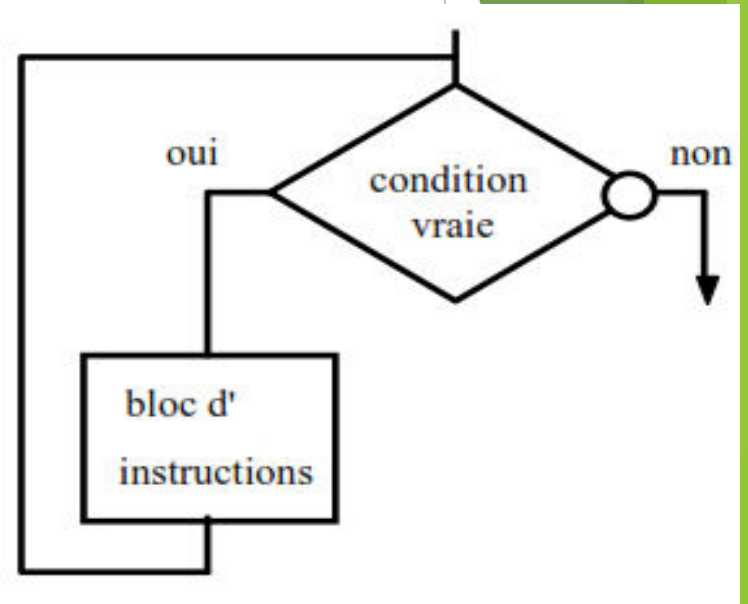
```
int main() {  
    int s = 0 ;  
    int i =1 ;  
    do  
    {      s = s + i;  
          i++;  
    } while(i<=10) ;  
    printf ("la somme est: %d\n " ,s) ;  
}
```

5. Structures et instructions de contrôle en C.

5.2. Les instructions structurés(Boucles):

b) Boucle while: Il s'agit de l'instruction:

```
while ( condition )  
{  
  bloc d'instructions  
}
```



- ❑ L'expression utilisée comme condition de la boucle est évaluée avant la première itération de la boucle. Ainsi, il est nécessaire que sa valeur soit définie à ce moment.

5. Structures et instructions de contrôle en C.

5.2. Les instructions structurés(Boucles):

b) L'instruction while:

Exemple 1 : $s = 1 + 2 + 3 + \dots + 10$

```
int main() {
    int s = 0 ;
    int i=1 ;
    while (i<=10)
    {   s = s + i; //s=s+i++;
        i++;
    }
    printf ("la somme est:%d \n " ,s);
}
```

Exemple 2 : Ecrire un programme qui lit deux entiers a et b au clavier et affiche leur Plus Grand Commun Diviseur (PGCD). On suppose que $a \geq b$.

5. Structures et instructions de contrôle en C.

5.2. Les instructions structurés(Boucles):

Boucle while:

```
#include <stdio.h>
int main() {
    int a, b , r ;
    printf("Entrez a: \t");
    scanf("%d\n", &a);
    printf("Entrez b: \t");
    scanf("%d\n", &b);
    r = a%b;
    while (r!=0)
        {
            a = b;
            b = r;
            r = a%b;
        }
    printf ("PGCD =%d \n" ,b );
}
```

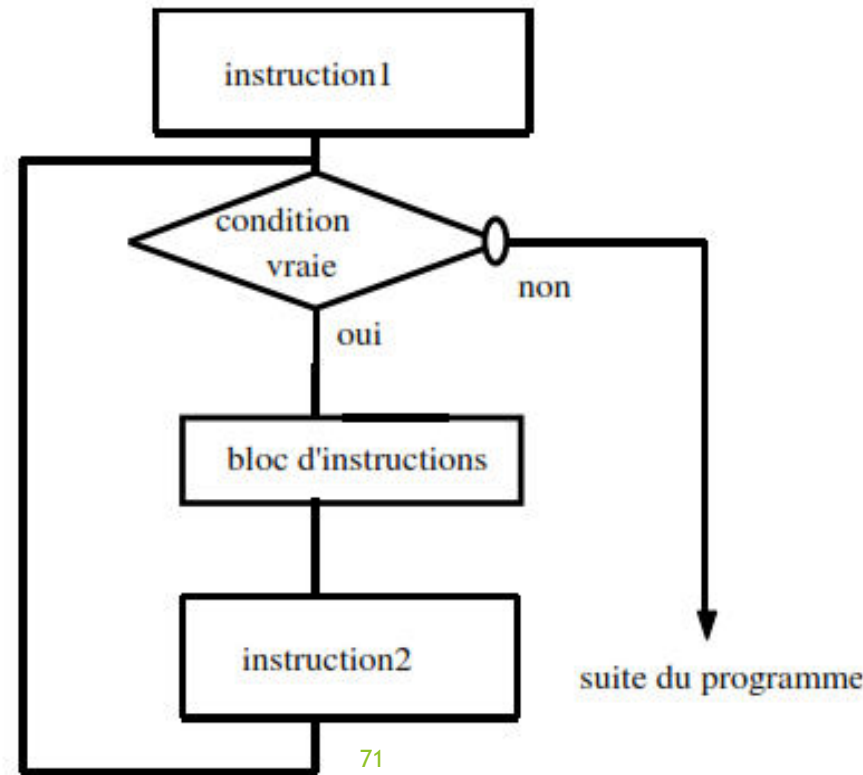
5. Structures et instructions de contrôle en C.

5.2. Les instructions structurés(Boucles):

c) Boucle for

Il s'agit de l'instruction:

```
for ( instruction1 ; condition ; instruction2 )  
{  
    bloc d'instructions  
}
```



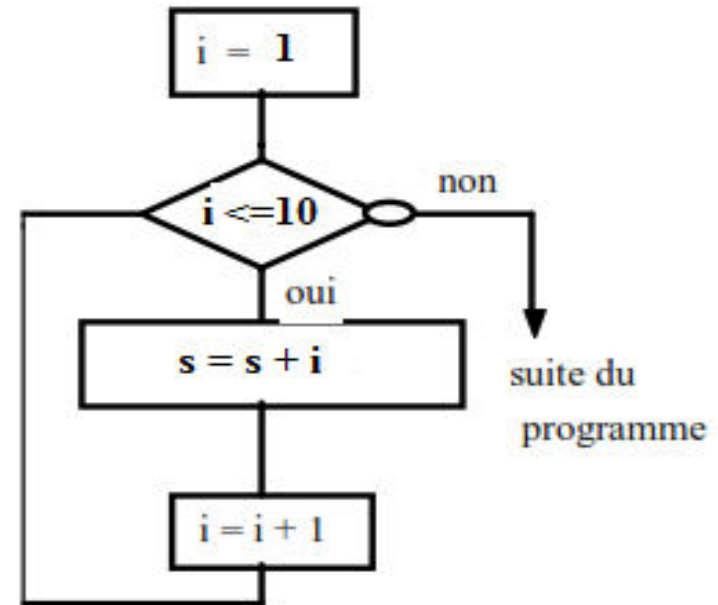
5. Structures et instructions de contrôle en C.

5.2. Les instructions structurés(Boucles):

c) Boucle for

Exemple: $s = 1 + 2 + 3 + \dots + 10$

```
int main() {  
    int s = 0 ;  
    int i;  
    for(i=1; i<=10; i++)  
    {  
        s = s + i;  
    }  
    printf ("la somme est:%d \n " ,s);  
}
```



5. Structures et instructions de contrôle en C.

5.2. Les instructions structurés(Boucles):

c) Boucles for et while

Cet expression de **for** est équivalent à:

```
initialisation;
```

```
while( condition)
```

```
{ bloc d'instructions
```

```
    incrémeDec;
```

```
}
```

```
for (initialisation ; condition ; incrémeDec  
{  
    bloc d'instructions  
}
```

```
initialisation ;  
do {  
    bloc d'instructions  
    incrémeDec  
} while( condition );
```

incrémeDec : incrémentation ou décrémentation

RQ: Dans la boucle **while** on vérifie la condition avant d'exécuter la liste d'instructions, tandis que dans la boucle **do.. while** on exécute la liste d'instructions **avant** de vérifier la condition.

5. Structures et instructions de contrôle en C.

5.2. Les instructions structurés(Boucles):

c) Boucles for

Exemple 2 : Ecrire un programme qui lit un entier n au clavier et affiche ses diviseurs.

```
#include <stdio.h>
int main() {
    int n,i ;
    printf("Entrez n: ");
    scanf("%d", &n);
    printf ("Les diviseurs de %d sont: \n", n);
    for(i=1 ; i<=n ; i++)
        { if (n%i == 0)
            printf("%d \n", i );
        }
}
```

5. Structures et instructions de contrôle en C.

5.2. Les instructions structurés(Boucles):

Exemple 3 : calculer le factoriel d'un entier n? $1*2*3*4*...*n$

```
#include <stdio.h>
int main() {
    int n, i , fact=1;
    for(i=1; i<=n;i++) fact*=i; /*
    fact=fact*i*/
    printf(" !%d = %d ",n , fact);
}
```

Ecrire un programme qui affiche les nombres paires<100

- Avec la boucle for
- Avec la boucle while
- Avec la boucle do

..while

```
#include <stdio.h>
int main() {
    int n, i , fact=1;
    i=1;
    while(i<=n){
        fact*=i;
        i++;
    }printf(" !%d = %d ",n , fact);
}
```

```
#include <stdio.h>
void main() {
    int n, i , fact=1;
    i=1;
    do{
        fact*=i;
        i++;
    } while(i<=n);
    printf(" !%d = %d ",n , fact);
}
```

5. Structures et instructions de contrôle en C.

5.3. Les instructions de branchement inconditionnel

a) L'instruction **break** :

Elle sert à interrompre le **déroulement de la boucle** en cours d'exécution à l'instruction qui suit cette boucle.

b) L'instruction **continue**:

Elle permet d'interrompre **l'itération courante** de la boucle et de passer à l'itération suivante.

5. Structures et instructions de contrôle en C.

5.3. Les instructions de branchement inconditionnel

- a) L'instruction break peut être utilisée dans une boucle (for, while, ou do .. while). Elle permet d'arrêter le déroulement de la boucle et le passage à la première instruction qui la suit.

En cas de boucles imbriquées, break ne met fin qu' à la boucle la plus interne

```
void main
```

```
{ int i,j;
```

```
  for(i=0;i<4;i++)
```

```
    for (j=0;j<4;j++)
```

```
      { if(j==1) break;
```

```
        printf("i=%d,j=%d\n ",i,j);
```

```
      }
```

```
  }
```

```
résultat:  i=0 , j=0  
           i=1 , j=0  
           i=2 , j=0  
           i=3 , j=0.
```

5. Structures et instructions de contrôle en C.

5.3. Les instructions de branchement inconditionnel

L'instruction continue: peut être utilisée dans une boucle (for, while, ou do .. while). Elle permet l'**abandon** de l'itération courante et le **passage** à l'itération suivante

```
int main() {
    int i;
    for(i=1; i<5; i++)
    { printf("début itération %d\n " ,i);
      if(i<3) continue;
      printf(" fin itération %d\n " ,i);
    }
    return 0;
}
```

résultat: début itération 1
début itération 2
début itération 3
fin itération 3
début itération 4
fin itération

5. Structures et instructions de contrôle en C.

5.3. Les instructions de branchement inconditionnel

Break;



```
int i, j=1;
char a;
for (i = -10; i <= 10; i++){
```

```
while(j!=0) /*boucle infinie
*/ {
    a=getchar();
    if(a == 'x')
        break;
}
```

Si x est tapée au clavier

Continue;



```
for (i = -10; i <= 10; i++)
{
    if (i == 0)
        continue;
    /* pour éviter la division par
zéro*/
    printf(" %f", 1 / i);
}
```

return (expression);
permet de sortir de la fonction qui la contient