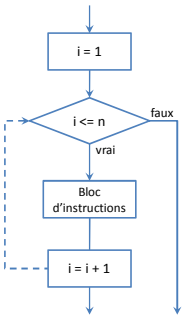
Boucle *for*

- **Instruction de contrôle** à l'instar de *while* et *do…while*
- Dans une seule instruction :
  - Affectation : i=1
  - Test : i <= n
  - Incrément : i = i + 1



Boucle *for*

```
1  #include <stdio.h>
2
3  /*  Factoriel avec for  */
4
5  int main () {
6
7     int i, n;
8     int factoriel;
9
10     /* point d'observation 1 */
11     printf ("Entrez un nb entier : ");
12     scanf ("%d", &n);
13
14     factoriel = 1;
15
16     /* point d'observation 2 */
17     for (i=1; i<= n; i=i+1) {
18        factoriel = factoriel * i;
19        /* point d'observation 3 */
20     }
21
22     /* point d'observation 4 */
23     printf ("factoriel de %d est %d \n", n, factoriel);
24  }
25
```

for (i=1 ; i <= n ; i = i+1)

Trace d'exécution

	i	n	factoriel
PO 1	?	?	?
PO 2	?	3	1
PO 3	1	3	1
PO 3	2	3	2
PO 3	3	3	6
PO 4	4	3	6

Boucle *for*

- Format :

for (*affectation* ; *test* ; *incrément*)
{
    bloc d'instructions ;
}

Affectation test Incrément

```
for (i=1; i<= n; i=i+1) {
   factoriel = factoriel * i;
}
```

Boucles imbriquées

- Une boucle peut contenir une autre…
- Exemples :

```
while (test 1) {
   for (affectation;  test 2; incr)
        {  instructions for;  }
   autres instruction while;
}
```

```
for (affectation;  test 2; incr) {
   while (test 1)
      {  instructions while;  }
   autres instruction for;
}
```

```
while (test 1) {
   while (test 2)
        {  instructions while;  }
   autres instruction while;
}
```

```c
#include <stdio.h>

/*  CM 5: afficher le tableau de multiplication */

int main () {
   int i,j;

   for (i=1; i <= 10; i++) {
      printf ("%d : ", i);
      for (j=1; j <= 10 ; j++) {
         printf ("\t %d", (i*j));
      }
      putchar ('\n');
   }
}
```

à chaque interaction du premier *for*, on exécute tout le second *for*.

```
kirsch@vbxlinux:~/code$ ./cm5-loopimbr
1 :     1       2       3       4       5       6       7       8       9       10
2 :     2       4       6       8       10      12      14      16      18      20
3 :     3       6       9       12      15      18      21      24      27      30
4 :     4       8       12      16      20      24      28      32      36      40
5 :     5       10      15      20      25      30      35      40      45      50
6 :     6       12      18      24      30      36      42      48      54      60
7 :     7       14      21      28      35      42      49      56      63      70
8 :     8       16      24      32      40      48      56      64      72      80
9 :     9       18      27      36      45      54      63      72      81      90
10 :    10      20      30      40      50      60      70      80      90      100
```

E / S non-formaté

```c
#include <stdio.h>

/*  CM 5: afficher le tableau de multiplication */

int main () {
   int i,j;

   for (i=1; i <= 10; i++) {
      printf ("%d : ", i);
      for (j=1; j <= 10 ; j
         printf ("\t %d",
      }
      putchar ('\n');
   }
}
```

Fonctions de la bibliothèque **stdio.h**

- E/S pour 1 caractère
- Sortie :
  `putchar (char)`
- Entrée :
  `char = getchar()`

Attention à l'indentation

```c
#include <stdio.h>

/*  CM 5: afficher le tableau de multiplication */

int main () {
int i,j;

   for (i=1; i <= 10; i++) {
printf ("%d : ", i);
      for (j=1; j <= 10 ; j++) {
printf ("\t %d", (i*j));
      }
putchar ('\n');
}
}
```

Indentation !!
But : Rendre le code compréhensible

Opérateurs unaires ++ et --

- Instructions très souvent utilisée : i = i + 1;
- Opérateur d'incrémentation **++** :

i ++ ; équivaut i = i + 1;

- Opérateur de décrémentation **--** :

i -- ; équivaut i = i - 1;

Opérateurs ++ et --

```
...
 int i, k;
 i = 0;
 k = 0

 i++;
 printf ("i=%d",i);
...
```

i ++ ; ⇒ i = i + 1;

i = 1

i++ ou ++i ?!

```
...
 i = 3;

 k = i++ - 5;

 printf ("k=%d",k);
...
```

k = i ++ - 5; ⇒
k = i - 5;
i = i + 1;

k = -2
i = 4

! i ++
++ après, incrément réalisé après !!!

Opérateurs ++ et --

```
...
 i = 1;

 k = i++;

 printf ("k=%d",k);
...
```

k = i ++ ; ⇒
k = i;
i = i + 1;

k = 1
i = 2

```
...
 i = 2;
 k = ++i;
...
```

k = 2

i++ ou ++i ?!

```
...
 i = 4;

 k = ++i + 5;

 printf ("k=%d",k);
...
```

k = ++i + 5; ⇒
i = i + 1;
k = i + 5;

k = 10
i = 5

! ++i
++ avant, incrément réalisé avant !!!

Opérateurs d'affectation élargie

- Principaux opérateurs d'affectation :

  += -= *= /= %=

```
k += i;   ⇒  k = k + i;
k -= i;   ⇒  k = k - i;
k *= i;   ⇒  k = k * i;
k /= i;   ⇒  k = k / i;
k %= i;   ⇒  k = k % i;
```

Priorité entre les opérateurs

Opérateurs unaires :	**++**	**--**	**(cast)**	
Opérateurs arithmétiques	*	/	%	
	+	-		
Opérateurs relationnels :	<	<=	>	>=
	==	!=		
Opérateurs logiques :	&&			
	||			
Opérateurs d'affectation :	=	*=	/=	%=
	+=	-=		