TP 6 : Les structures de répétition (Boucles)

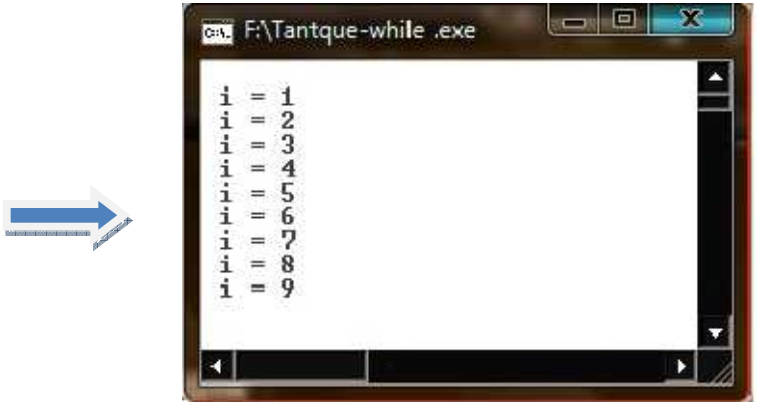
1. La structure de répétition (Tantque) :

L'instruction « **while »** dans le langage C++ correspond à l'instruction « **Tantque »** dans le langage algorithmique, elle est utilisée comme suit :

Langage algorithmique	Langage C++
Tantque (condition**) faire** <instructions> **FinTantque**	**while (**condition**)** **{** <instructions> **}**

Exemple :

```
i= 1;
while (i<=9)
{
cout<< "i = "<< i;
i=i+1;
}
```



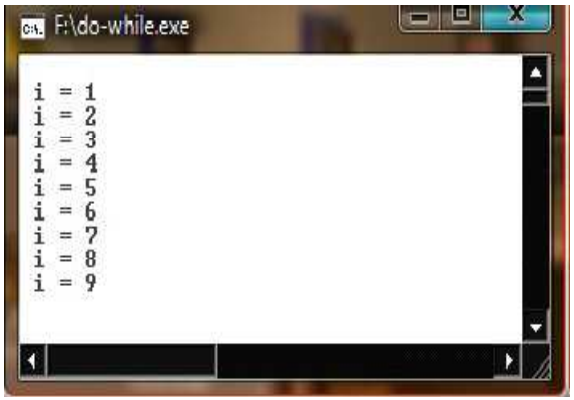
2. L'instruction do - while (Répétez – Tant que) :

Dans l'instruction **do-while** la condition est vérifiée **<u>à la fin</u>** et non pas au début, donc les instructions sont **<u>exécutées au moins une fois</u>**. L'instruction **do-while** est utilisée comme suit :

```
do
{
<instructions>
} while (condition);
```

Exemple :

```
i= 1;
do
{
cout << "i = "<< i;
i=i+1;
} while (i<=9) ;
```



<u>Remarque:</u>

L'instruction do-while dans le langage C++ **<u>diffère de l'instruction</u>** (Répéter -jusqu'à) du langage algorithmique car la condition à vérifier est **inversée**, elle correspond en quelque sorte à une instruction (**Répétez – Tantque**) qui n'est pas utilisée dans le langage algorithmique.

Exemple :

Langage algorithmique	Langage C++

i← 0;
Répétez
Ecrire(i);
i← i + 1;
 	
jusqu'à (i = 10)
	i = 0;
	do
	{
	cout<<i
	i = i+1;
	} while (i **!=** 10**);**

3. L'instruction for (pour) :

L'instruction **for** dans le langage C++ correspond à l'instruction **pour** dans le langage algorithmique, elle est utilisée comme suit :

```
for (valeur initiale; condition; pas)
{
< instructions >
}
```

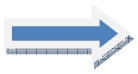
- "**valeur initiale**" représente la valeur initiale de la variable de la boucle,
- "**condition**" représente la condition d'arrêt de la boucle, et
- "**pas**" représente le pas de la boucle.

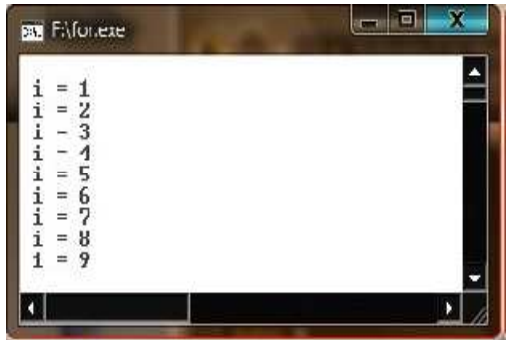
Langage algorithmique	Langage C++
Pour i **allant de** 0 a N **pas** 1 **faire** **Écrire**(i); **Fin pour ;**	**for** (i=0; i<=N; i= i+1) **{** cout << i; **}**

Exemple :

```
for ( i=1; i<=9; i=i+1)
{
 cout << "i = "<< i;


}
```





Remarque importante :

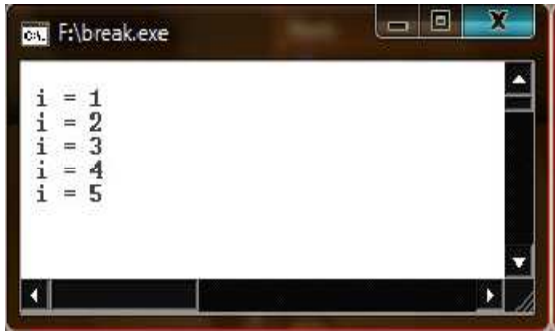
Si la boucle contient une seule instruction on peut ne pas utiliser les parenthèses '**{ }**'.

4. L'instruction break :

L'instruction **break** permet de **sortir d'une boucle** (for , while, do-while) ou d'une instruction de sélection (switch-case).

Exemple :

```
for (i=1; i<=9; i=i+1)
{
if (i == 6) break;
cout << "i = "<< i;
}
```

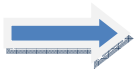


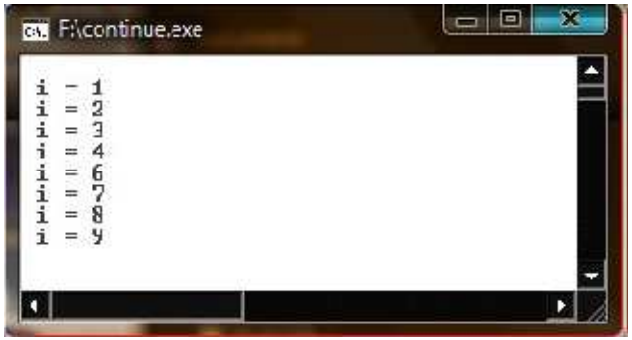
5. L'instruction continue :

L'instruction **continue** permet de **sauter les instructions suivantes dans une boucle** (for , while, do-while) et de reprendre la boucle depuis le début.

Exemple :

```
for (i=1; i<=9; i=i+1)
{
if (i == 5) continue;
cout << "i = "<< i;
}
```





Exercice 1 :

Écrire des programmes en langage C++ qui permettent de :

1) lire un nombre entier positif N et affiche le nombre de chiffres qui le compose.
2) lire un nombre entier positif N et l'inverse.
3) lire un nombre entier positif N et affiche si il est premier ou non.
4) afficher tous les nombres premiers inférieurs à 100.
5) afficher 100 nombres premiers.

==========solution==========

Exercice 1 :

1)

```
#include <iostream>
using namespace std;
// Declaration des variables
int n,i,nbr;
int main()
{
cout << " programme qui calcule le nombre de chiffres d'un nombre entier";
cout << endl;

cout << "Donnez le nombre n : ";
cin >> n;
nbr=0;
do {
      n=n/10;
      nbr=nbr+1;
}
while (n!=0);

cout<< "le nombre de chiffres="<<nbr;

getchar();
getchar();
}
```

2)

```
#include <iostream>
 using namespace std;
// Declaration des variables
int n1,n2;   // n2 est l'inverse de n1

int main()
{
cout << " programme qui calcule l'inverse d'un nombre entier";

cout << endl;
cout << "Donnez le nombre a inverser: ";
cin >> n1;
n2=0;
while (n1>0){
      n2=(n2*10)+n1%10;
      n1=n1/10;
}

cout<< "l'inverse est: "<<n2;

getchar();
getchar();
}
```

3)

```
#include <iostream>
//#include <cmath>
using namespace std;
// Declaration des variables
int n,i,nbr_div;
```

```
int main()
{
cout << " programme qui verifier si un nombre entier est premier ou non";

cout << endl;
cout << "Donnez le nombre: ";
cin >> n;
nbr_div = 0; // initialiser le nbr de diviseur à 0
for (i=2;i<=n/2;i=i+1){
      if (n%i==0)
      nbr_div=nbr_div+1;
}

if (nbr_div==0)
cout<< "le nombre "<<n<<" est premier";
else
cout<< "le nombre "<<n<<" n'est pas premier";

getchar();
getchar();
}
```

Remarques importante pour (3) :

- **pour optimiser le calcule, on a initialisé l'indice i de la boucle à 2 jusqu'à n/2.**
- **La meilleure solution est d'utiliser la boucle while , et une variable booléenne « premier » comme suit :**

```
i=2 ;
premier= true ;
While ((i<=n/2)&&(premier==true)){
```

- **if (n%i==0)**

```
      premier=false;
          i=i+1;
      }
   if (premier==true)
      cout<< "le nombre "<<n<<" est premier";
   else
      cout<< "le nombre "<<n<<" n'est pas premier";
```

4)

```
#include <iostream>
//#include <cmath>
using namespace std;
// Declaration des variables
int n,i;
bool premier;

int main()
{
cout << " les nombres premiers inferieur a 100:";

cout << endl;

for (n=1;n<=100;n=n+1)
{
premier=true;
i=2;
while((i<=n/2)&&(premier==true)){
      if (n%i==0)
      premier=false;
      i=i+1;
}
```

```
if (premier==true){
cout<< n ;
cout << endl;
}
}
getchar();
}
```

5)

```
#include <iostream>
//#include <cmath>
using namespace std;
// Declaration des variables
int n,i,nbr;
bool premier;

int main()
{
cout << " les 100 nombres premiers:";

cout << endl;
//cout << "Donnez le nombre: ";
//cin >> n;
n=1;
nbr=1; // pour 100 nombres premiers
while (nbr<=100)
{
premier=true;
i=2;
while((i<=n/2)&&(premier==true)){
      if (n% i==0)
      premier=false;
      i=i+1;
}

if (premier==true){
cout<< n ;
cout << endl;
nbr=nbr+1;
}
n=n+1;
}
getchar();
}
```