Ministère de l'enseignement supérieur et de la recherche scientifique
Université Abderrahmane MIRA de Bejaia

Faculté de Technologie
Département de Technologie

Structure des ordinateurs et applications

À propos du cours

- **Crédits**: 02
- **Coefficients**:02

Notion d'algorithme et de programme

Structures de contrôle répétitives

structures de contrôle répétitives

- Les **instructions itératives** (ou **structures de contrôle répétitives**) sont des instructions qui permettent de **répéter plusieurs fois un bloc de code** en fonction d'une condition. il existe trois types principaux de boucles :
  - Boucle Pour (For);
  - Boucle Tant-que (While);
  - Boucle Répéter (Repeat).

Boucle Pour (For)

Boucle Pour (For)

La structure de contrôle répétitive ***pour*** (mot-clé for en C) permet d'exécuter un bloc d'instructions selon un compteur défini à l'avance.

La syntaxe de la boucle pour

Algorithme	En langage C

Pour <indice> ← <vi> à <vf>
faire
<instruction(s)> ;
finPour;
	for (initialisation; condition;
	incrémentation) {
	// instructions;
	}

Remarque : En **C**, **l'incrémentation se fait généralement sur des variables entières**, mais elle **peut aussi se faire sur des réels (float)**.

Remarque

- La boucle for est surtout utilisée quand on connaît à l'avance le nombre de répétitions;
- Oublier une partie peut rendre la boucle infinie (initialisation; condition; incrémentation);
- La condition est testée AVANT chaque tour, Si elle est fausse dès le début → la boucle ne s'exécute jamais;
- Le point-virgule est obligatoire entre les parties for (int i = 1**;** i < 10**;** i++)
- Ne jamais mettre ; juste après le forSinon la boucle devient vide; for (i = 0; i < 10; i++)**;**
- Dans le langage C, le compteur n'est pas obligatoirement de type entier (int).Il peut être d'un autre type;

Exemple

Écrire un programme qui affiche les nombres de 1 à 10.

Algorithme nombre ;
Variables i : entier;
Début
Pour i ← 1 à 10 faire
Ecrire(i);
Fin-pour;
Fin.

En langage C

```
#include<stdio.h>
int main(){
for (int i = 1; i <= 10; i++)
{
printf("%d\n", i);
}
return 0 ;}
```

Boucle Tant-que (While)

La boucle tantque **(while)** en langage C) La boucle tant-que répète un bloc d'instructions tant qu'une condition est vraie. La condition est testée avant chaque itération (répétitions).

La syntaxe de la boucle tantque

Algorithme	En langage C

tant-que <condition> faire
<instruction(s)> ;
fin tant-que;
	while (condition) {
	// instructions ;
	}

Remarques :

1. Si la condition est fausse dès le départ, le bloc ne s'exécute jamais.

2. Toujours s'assurer que la condition soit susceptible de devenir fausse, sinon on risque une boucle infinie

Remarques :

- On utilise la boucle while lorsque l’on veut répéter un bloc d’instructions tant qu’une condition est vraie. Si la condition est fausse dès le départ, le bloc ne s'exécute jamais;
- Si la condition ne devient jamais fausse, la boucle tourne à l’infini. while (x >= 0) {
- On l’utilise souvent lorsque le nombre d’itérations n’est pas connu à l’avance.

Exemple

Écrire un programme qui affiche les nombres en commençant par 1 et continue l'affichage tant que la valeur ne dépasse pas 10.

Algorithme nombre **;**
Variables i : entier;
Début
i ← 1 ;
Tant-que (i <= 10) **faire**
Ecrire(i);
i ← i+1 ;
Fin-Tant-Que ;
Fin.

```
#include<stdio.h>
int main(){
int i = 1;
while (i <= 10) {
printf("%d\n", i);
i++;
}
return 0 ;}
```

Boucle Répéter (Repeat)

Boucle répéter…jusqu'à: (do…. while) en langage C) Contrairement à la boucle « Tant-que », la boucle « Répéter » exécute d'abord le bloc d'instructions, puis teste la condition. Elle s'exécute donc au moins une fois.

La syntaxe de la boucle répéter

Algorithme	En langage C

repéter
<instruction(s)> ;
Jusqu'à <condition> ;
	do {
	// instructions ;
	} while (condition) ;

Remarque :

1. La boucle Répéter est à utiliser lorsqu'on veut que le traitement se fasse **au moins une fois**, peu importe la condition.

2. En C, *while* et *do.. while* ont la même condition contrairement à l'algorithmique.

- On utilise la boucle do...while lorsqu'on veut exécuter un bloc d'instructions au moins une fois, puis répéter tant qu'une condition est vraie;
- Contrairement à while, la boucle do...while s'exécute toujours au moins une fois, car la condition est testée après le bloc.
- Attention à la syntaxeEn C, le while final se termine par un point-virgule ; Sans le ; le compilateur ne reconnaît pas la fin de l'instruction do...while et ça provoque une erreur.

  - while : Condition → Bloc → Retour à Condition;
  - do...while : Bloc → Condition → Retour à Bloc;

Indices pour while :"Répéter tant que …" ou "Tant que la condition est vraie;
Indices pour do...while :"Exécuter au moins une fois, puis répéter tant que …" ou "Demander une saisie et executer ensuite …"

Exemple 3:

Écrire un programme en langage C permettant d'afficher les nombres entiers successifs à partir de 1. L'affichage doit être répété jusqu'à ce que le nombre devienne strictement supérieur à 10.

Algorithme nombre **;**
Variables i : entier;
Début
i ← 1 ;
Répéter
Ecrire(i);
i ← i+1 ;
Jusqu'à (i > 10) **;**
Fin.

```
#include <stdio.h>
int main() {
    int i = 1;

    do {
        printf("%d\n", i);
        i = i + 1;
    } while (i <= 10);

    return 0;
}
```

```
Algorithme Exo2 ;
Variables
N, Sn, i : entier;
Début
Lire(N) ;
Sn ← 0 ;
Pour i ← 1 à N faire
Si (i mod 2 = 1) alors
Sn ← Sn + i ;
FinSi ;
FinPour ;
Écrire('La somme est : ', Sn);
Fin.
```

1. Traduire l'algorithme donné en un programme C;
2. Réécrire le programme C en remplaçant la boucle For par la boucle do while et la boucle while.
3. Modifier l'algorithme pour calculer et afficher la somme « Sn » des nombres qui ne sont pas des diviseurs de N.

```
Algorithme Exo2 ;
Variables
N, i, S1, S2 : entier
Début
Lire(N) ;
S1 ← 0 ;
S2 ← 0 ;
Pour i ← 1 à N faire
    Si (i ≤ N / 2) alors
        S1 ← S1 + i ;
    Sinon
        S2 ← S2 + i ;
    FinSi ;
FinPour ;
Écrire('S1= ', S1) ;
Écrire('S2= ', S2) ;
Fin.
```

1. Traduire l'algorithme donné en un programme C;
2. Réécrire le programme C en remplaçant la boucle for par la boucle do while et la boucle while.

- **Écrire un programme c qui permet d'afficher tous les nombres entiers supérieurs ou égale à 1 et inférieurs à m (entier positif) et qui sont divisible sur un nombre entier n.**

- **Écrire un programme qui lit un entier positif N et :**
  - **Compte combien de nombres pairs il y a entre 1 et N ;**
  - **Compte combien de nombres impairs il y a entre 1 et N ;**
  - **Affiche ces deux valeurs.**