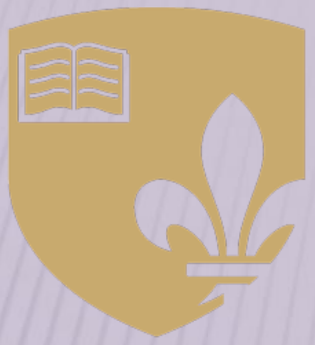


COLLEGE DE PARIS SUPÉRIEUR

DAKARTECH

PROGRAMMATIONS C

Dr Pape Abdoulaye **BARRO**
Enseignant – Chercheur
Spécialiste en Télémétrie & Systèmes Intelligents

STRUCTURES ITÉRATIVES

PROGRAMMATION C
STRUCTURES ITÉRATIVES

- Une itération consiste en la répétition d'un bloc d'instructions jusqu'à ce qu'une certaine condition soit vérifiée.
- Il existe 2 sortes d'itérations:
  - Le nombre de répétitions est connu dés le départ

    Exemple : Calcul de 1+2+3+…+n
  - Le nombre de répétitions est méconnu

    Exemple : Recherche de PGCD de deux entiers

PROGRAMMATION C
STRUCTURES ITÉRATIVES

Considérons le problème suivant:

On se propose d'afficher tous les entiers naturels strictement plus petits que 100. Il est évident que, même s'il est possible de résoudre ce problème en écrivant toutes les instructions comme suit :

```
printf(‘’1”)
printf(‘’2”)
..
printf(‘’99”)
```

Cette solution reste non désirée, d'où la nécessité de trouver une alternative. C'est là que la notion de boucle va prendre toute son importance.

PROGRAMMATION C
STRUCTURES ITÉRATIVES: LA STRUCTURE FOR

On utilise for si le nombre d'itération est connu d'avance.

- **syntaxe:**

```
for (initialisation;  condition ; incrémentation){
        // instructions
}
```

Exemple:

```
int i, som=0, n=100;
…
for(i=0; i<n; i++){
        som=som+i;
}
…
```

PROGRAMMATION C
STRUCTURES ITÉRATIVES: FOR

Exemple:

```
#include<stdio.h>

int main(){
  //Ce programme permet de calculer la factorielle d'un entier naturel
  int n, i, resultat;
  printf("Entrez un entier naturel");
  scanf("%d", &n);
  resultat = 1;
  for (i=2; i<=n; i++){
      resultat = resultat *i;
  }
  printf("%d!=%d", n, resultat );
  return 0;
}
```

PROGRAMMATION C
STRUCTURES ITÉRATIVES:FOR

- **Exemple**:
  Écrire un programme qui permet de calculer la somme S=1+2+3+4+….+ N.
  où N est saisi au clavier par l'utilisateur.

- **Solution**:

```
#include<stdio.h>
int main(){
        int i,S,N;
        S = 0;
        printf("Donner un entier");
        scanf("%", &N);
        for(i=1; i<=N; i++){
            S = S + i ;
        }
        printf("La somme est: %d", S);
      return 0;
}
```

PROGRAMMATION C

STRUCTURES ITÉRATIVES: LA STRUCTURE DO…WHILE

- Les instructions contenues dans la boucle do…while s'exécutent au moins une seule fois avant de tester la condition.
- La syntaxe est la suivante:

```
do{
        //instructions
}while(condition);
```

+ **Exemple:**

```
int i=0;
do{
   printf("Bonjour la classe %d fois ", (i+i));
   i++;
} while(i<4);
```

+ **Remarque**: Les variables sur lesquelles porte la condition doivent toujours être initialisée.

PROGRAMMATION C
STRUCTURES ITÉRATIVES: DO...WHILE

Exemple:

```
#include<stdio.h>
int main(){
        //Cet programme permet de calculer la factorielle d'un entier naturel
        int n, i, result;
        printf(''Entrez un entier naturel");
        scanf("%d", &n);
        result = 1;
        i= 1;
        do{
                result = result *i;
                i= i+1;
        }while(i<=n);
        printf("%d! =%d", n,result);
        return 0;
}
```

PROGRAMMATION C
STRUCTURES ITÉRATIVES: DO...WHILE

- **Exemple**:

  Écrire un programme qui affiche la table de multiplication de 8.

- **Solution**:

```
#include<stdio.h>
int main(){
        int i=0
        do{
            printf("8*%d=%d", i, i*8);
            i = i+1;
        } while ( i < 10 );
    return 0;
}
```

PROGRAMMATION C
STRUCTURES ITÉRATIVES: LA STRUCTURE WHILE

- Avec la boucle while, la condition d'entrée doit être définie au préalable sinon, en implémentant votre programme, vous risquez d'avoir des comportements étranges.

- La syntaxe est la suivante:

```
while (condition) {
        //instruction
}
```

- **Remarque** :

```
int i=0;
while(i<4){
   printf("Bonjour la classe %d fois ", (i+i));
   i++;
}
```

PROGRAMMATION C
STRUCTURES ITÉRATIVES: WHILE

Exemple:

```
#include<stdio.h>
int main(){
        //Cet programme permet de calculer la factorielle d'un entier naturel
        int n, i, result;
        printf("Entrez un entier naturel");
        scanf("%d", &n);
        result = 1;
        i = 1;
        while (i<=n) {
                result = result *i;
                i= i+1;
        }
        printf("%d != %d", i, result);
        return 0;
}
```

PROGRAMMATION C
STRUCTURES ITÉRATIVES: WHILE

- **Exemple**:

  Écrire un programme permettant de calculer la somme S=1+2+3+...+ N, où N saisi par l'utilisateur.

- **Solution**:

```
#include<stdio.h>
int main(){
        int i=1,S=0,N;
        printf("Donner un entier:");
        scanf ("%d", &N);
        while(i <= N ){
                S = S + i;
                i =  i + 1;
        }
        printf("La somme de 1 à %d est %d", N, S);
      return 0;
}
```

PROGRAMMATION C
STRUCTURES ITÉRATIVES

QUELLE BOUCLE CHOISIR ?

Chaque boucle a un contexte dans lequel son utilisation est plus adéquate bien qu'il soit possible d'utiliser l'une comme l'autre dans certains contextes.

- **for**: Lorsque le nombre d'itérations est connu;
- **do… while**: Lorsque nous sommes certains d'exécuter le bloc d'instruction au moins une fois;
- **while**: Lors que le bloc d'instruction peut ne pas du tout être exécuté.

PROGRAMMATION C

STRUCTURES ITÉRATIVES: BRANCHEMENT CONDITIONNEL

- **break:** achève immédiatement la boucle ou la conditionnelle qui l'encadre.
- **continue:** ignore le reste des instructions et passe tout de suite à l'itération suivante.

```
int i;
for(i=0;i<10;i++){
  if(i%2==0){
    continue;
  }
 if(i>7){
    break;
  }
   printf("->%d\n",i);
}
```

```
->1
->3
->5
->7
```

PROGRAMMATION C

STRUCTURES ITÉRATIVES: BRANCHEMENT CONDITIONNEL

- **Que produit le programme suivant ?**

```
int i;
int j;
for(j=0;j<2;j++){
  for(i=0;i<10;i++){
    if(i%2==0){
      continue;
    }
    if(i>3){
      break;
    }
     printf("->%d\n",i);
  }
}
```

```
->1
->3
->1
->3
```

PROGRAMMATION C

STRUCTURES ITÉRATIVES: BRANCHEMENT CONDITIONNEL

return: A une double casquette, il permet de :

- Achever immédiatement l'exécution de la méthode qui l'embarque.
- Préciser la valeur retournée par une fonction

```
int i;
for(i=0;i<10;i++){
  if(i%2==0){
    continue;
  }
 if(i>7){
    return;
  }
   printf("->%d\n",i);
}
```

```
->1
->3
->5
->7
```

PROGRAMMATION C

STRUCTURES ITÉRATIVES: BRANCHEMENT CONDITIONNEL

- **Que produit le programme suivant ?**

```
int i;
int j;
for(j=0;j<2;j++){
  for(i=0;i<10;i++){
    if(i%2==0){
      continue;
    }
    if(i>3){
      return;
    }
     printf("->%d\n",i);
  }
}
```

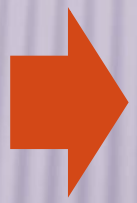
```
->1
->3
```

PROGRAMMATION C

STRUCTURES ITÉRATIVES: BRANCHEMENT CONDITIONNEL

- **goto**: Elle permet classiquement le branchement en un emplacement quelconque du programme.
- Cette endroit doit être nommé.

```
int i;
for(i=0;i<10;i++){
  if(i%2==0){
    continue;
  }
 if(i>7){
    goto out;
  }
   printf("->%d\n",i);
}
out:printf(""); //ou out:; ou out: {};
```



```
->1
->3
->5
->7
```

CAS PRATIQUES N°3

- **Application 16 :**

Écrire un programme permettant de calculer la somme des nombres impairs de 1 à n.

- **Application 17:**

Écrire un programme permettant de calculer la moyenne des n premiers entiers. n étant strictement positif.

- **Application 18:**

Écrire un programme permettant de lire 20 nombres au clavier et d'afficher le carré des nombres pairs uniquement. Attention, on ne mémorisera pas les 20 valeurs saisies.

- **Application 19:**

Écrire un programme qui demande un nombre à l'utilisateur, puis vérifie et affiche que les nombres paires.

Le programme s'arrête lorsque l'utilisateur donne -1.

- **Application 20:**

Écrire le programme permettant de lire puis d'afficher une valeur comprise entre 1 et 31; on recommencera la saisie jusqu'à ce que la valeur soit bien dans les bornes imposées.

Affaires à suivre