Révisions notions A1

ESILV - DÉPARTEMENT IBO

Notions à revoir

- Méthodes
- Boucles
- Tableaux

Les boucles

Boucles de répétitions

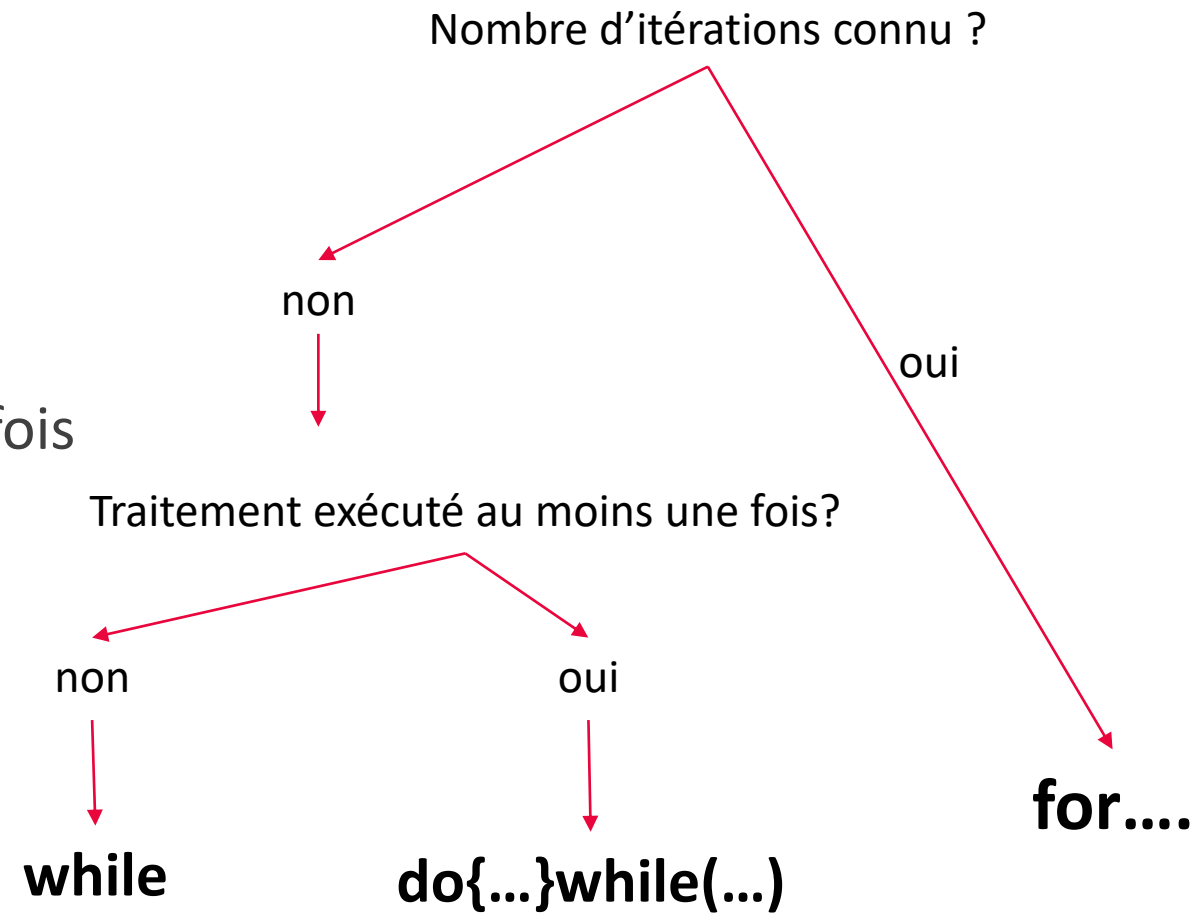
- Structure **itérative**

  for(…){….}

- Structures **événementielles**

  while(conditions){…} ➔ Au minimum 0 fois

  do{…}while(conditions); ➔ Au minimum 1 fois



Exercices

Exercice 1

Ecrire un programme en c# qui :

1. Demande à l'utilisateur de saisir un entier strictement positif « n »
2. Demande à l'utilisateur de saisir « n » entiers, calcule la moyenne de ces entiers. **On ne parle pas de tableau.**
3. Affiche la moyenne des nombres saisis.

Exemple :

n = 5

Les nombres saisis au fur et à mesure sont : 6, 7, 8, 5, 4

Le programme affiche ➔ La moyenne des 5 nombres saisis est 6

Exercice 2

Ecrire un programme en c# qui :

- Afficher et compter tous les nombres multiples de 3 et 7 entre 1 et 1000
- Affiche le nombre total calculé précédemment.

Exercice 3

Ecrire un programme en c# qui :

- Demande à l'utilisateur le nombre d'élèves d'une classe
- Puis, de saisir les notes de ces élèves. **On ne parle pas de tableau.**
- Affiche :
  - La note maximale
  - La note minimale
  - La moyenne de la classe

Exercice 4

Ecrire un programme en c# qui :

1. Demande à l'utilisateur de saisir un entier strictement positif « x »
2. Demande à l'utilisateur de saisir « x » entiers positifs. On comptabilise le nombre d'entiers premiers parmi les « x » entiers. Pour rappel, un entier est premier s'il est divisible que par 1 et par lui-même. **On ne parle pas de tableau.**
3. Affiche le nombre d'entiers premiers saisis.

Exemple :

x = 6

Les entiers saisis sont : 8, 7, 12, 5, 3, 20

Le programme affiche ➔ 3 nombres premiers ont été saisis.

Les méthodes

Syntaxe - Méthode

```
Signature de la méthode {
    Bloc de code de la méthode (ensemble d'instructions)
}
```

- Les noms des méthodes en C# doivent :
  - Commencer par une lettre majuscule, et si plusieurs mots sont accolés, mettre une majuscule à chaque première lettre des mots pour visualiser leur séparation.
  - Pas de caractères spéciaux, ni d'espace!

```
static typeRetour nomFonction(type1 param1, type2 param2, ..., typek paramk){

    //corps de la méthode
    return valeur; //si la méthode retourne un résultat
}
```

Exercices

Exercice 1

Ecrire une méthode ***static int SaisieNbPositif()*** qui demande à l'utilisateur de saisir un nombre entier positif et le retourne en résultat de sortie. L'utilisateur peut se tromper plusieurs fois.

Faire appel à la méthode dans le Main et afficher un message significatif à l'utilisateur.

Exercice 2

Ecrire une méthode ***static bool EstPositif(int n)*** qui prend en entrée un entier et vérifier s'il est positif ou pas.

Faire appel à la méthode dans le Main et afficher un message significatif à l'utilisateur.

Exercice 3

Ecrire une méthode ***static bool EstPair(int n)*** qui prend en entrée un entier positif et vérifier s'il est pair ou pas.

Faire appel à la méthode dans le Main et afficher un message significatif à l'utilisateur. Utilisation de la méthode de l'exercice 1 nécessaire (***SaisieNbPositif())***

Exercice 4

Ecrire une méthode ***int NombreOccurrences(string s, char l)*** qui calcule et retourne le nombre de d'apparitions du caractère *l* dans la chaine *s* . 0 si le caractère *l* n'apparait jamais dans s.

Testez votre code à l'aide d'une méthode ***void TestNombreOccurrences()*** qui :

déclare un caractère l ainsi qu'une variable chaine de caractère s,

demande à l'utilisateur d'entrée une chaine de caractères, et une lettre correspondant à la lettre à chercher,

appelle la fonction ***NombreOccurrences*** puis affiche un message significatif à l'utilisateur

Exercice 5

Ecrire une méthode ***string RemplaceLettre(string s, char l, char c)*** qui :

Remplace la première apparition de la lettre « l » dans la chaine « s » par la lettre « c ». La méthode retourne la nouvelle chaine modifiée.

Vous aurez besoin d'utiliser la fonction « **Replace** » existante en c# permettant de remplacer une lettre par une autre dans une chaine. Exemple : s = "bonjour", l='o', c = 'p' lorsque l'on fait : string x = s.Replace(l, c) ; x va contenir la nouvelle chaine "bpnjpur".

Testez votre code à l'aide d'une méthode ***void TestRemplaceLettre()*** qui :

- déclare deux variables caractères l et c ainsi qu'une variable chaine de caractère s,
- demande à l'utilisateur d'entrée une chaine de caractères, et deux lettres correspondant à la lettre à remplacer et la lettre de remplacement,
- appelle la fonction *RemplaceLettre* puis affiche la nouvelle chaine après remplacement

Exercice 6

Ecrire une méthode ***int RemplaceCompte(ref string s, char l, char c)*** qui :

Remplace toute apparition de la lettre « l » dans la chaine « s » par la lettre « c ». La méthode calcule et retourne le nombre de remplacement du caractère *l* dans la chaine *s* par le caractère *c*. 0 si le caractère *l* n'apparait jamais dans s et donc ne peut pas être remplacé.

Vous aurez besoin d'utiliser la fonction « **Replace** » existante en c# permettant de remplacer une lettre par une autre dans une chaine. Exemple : s = "bonjour", l='o', c = 'p' lorsque l'on fait : string x = s.Replace(l, c) ; x va contenir la nouvelle chaine "bpnjpur".

Testez votre code à l'aide d'une méthode ***void TestRemplaceCompte()*** qui :

- déclare deux variables caractères l et c ainsi qu'une variable chaine de caractère s,
- demande à l'utilisateur d'entrée une chaine de caractères, et deux lettres correspondant à la lettre à remplacer et la lettre de remplacement,
- appelle la fonction *RemplaceCompte* puis affiche la nouvelle chaine après remplacement et le nombre de remplacement.

Exercice 7

Reprendre l'exercice 4 de la section « Boucles » et le décomposer en méthodes pour une meilleure lisibilité du code.

Les tableaux

Tableau

- Un tableau est une **variable** (espace mémoire, type et nom) **particulière** permettant **d'accéder indirectement** à l'**espace mémoire alloué à un ensemble d'éléments** (dit « tableau ») de même type que cette variable.

- Syntaxe :

```
type[]   nomDuTableau;
```

➔ Le tableau n'existe toujours pas ➔ pas encore d'espace mémoire alloué

➔ La valeur est **null**

- Exemple :

```
//déclaration d'un tableau, qui pourra « contenir » des entiers
int[] tab;
```

Dimensionnement d'un tableau

- **Allocation mémoire** ➔ réserver en mémoire la taille occupée par le tableau
  - Espace mémoire = nombre de cases x nombre d'octets pour le type
  - Un tableau est **de taille fixe**

```
type[]   nomDuTableau = null ;
tab = new type[taille]; //allocation mémoire
```

- Exemple:

```
int[] tab= null ; //déclaration
tab = new int[6]; //allocation mémoire de 6 cases pouvant
chacune contenir la valeur d'un entier
```

Création d'un tableau

- Syntaxe : **Déclaration + Allocation mémoire**

type[] nomDuTableau = **new type[taille];**

- Exemple:

```
int[] tab = new int[6];
```

Accès aux éléments d'un tableau

- Accès en **lecture** ou en **écriture** à une case via un index
  - ✓ En c#, le **premier élément est indexé par 0.**
- Syntaxe : accès à la (index-1) ième case

```
nomDuTableau[index];
```

- Exemple:

```
int[] tab = new int[6];          // création d'un tableau d'entiers
tab[0] = 5;                      // accès en écriture au 1er élément
Console.WritelLine( tab[0] );    // accès en lecture au 1er élément
tab[1] = 13;                     // accès au 2ème élément
// ….
```

Exercices

Exercice 1

Compléter le programme suivant pour afficher les éléments du tableau en utilisant :

- une boucle for
- une boucle foreach

Quelle est la différence?

```
 static void Exercice1()
        {
            int[] tableau = { 15, 1, 23, 18, 65,
33, 9, 7, 24, 45, 23, 89 };


// …… A COMPLETER ….




        }
```

Exercice 2

Compléter le programme suivant pour afficher en **sens inverse** les éléments du tableau.

C’est-à-dire afficher les éléments de la dernière position à la première.

```
 static void Exercice2()
        {
            int[] tableau = { 15, 1, 23, 18, 65, 33, 9, 7, 24, 45, 23, 89 };


// …… A COMPLETER ….



        }
```

Exercice 3

Compléter le programme suivant afficher un élément sur deux du tableau en commençant par :

- le premier élément
- le dernier élément

```
 static void Exercice3() {
    int[] tableau = { 15, 1, 23, 18, 65, 33, 9, 7, 24,
45, 23, 89 };


    // …… A COMPLETER ….




        }
```

Exercice 4

Reprendre les exercices 3 et 4 de la section « Les boucles » en utilisant des tableaux d'entiers.

SCHOODING