AJC-FORMATION

LE LANGAGE FONDAMENTAL

- Value Types
- Variable
- Constante
- Énumération
- Méthodes
- Opérateur
- Condition
- Boucle

Value Type

Types Valeurs

- Langage Strongly Typed : il faut définir les types avant de les utiliser (pas de type ‘variant’)
- Les différents Types :
  - Type Intrinsic : type propre au langage
  - Type Définit par Utilisateur : différent de Type Intrinsic
  - Type Valeur : allocation de mémoire dans le Stack
  - Type Référence : Heap manage
  - Pointer Type : Unmanaged, utilisation de Pointeurs
  - Chaque Type Intrinsec a son équivalent dans le CLS afin de permettre le passage entre les langages

Type Valeur

- Tous les ‘type intrinsec’ sont des Type Valeur à l’exception de ‘String ’et ’Object’

- Tous les type ‘Définis par Utilisateur’ sont des ‘Type Référence ’ à l’exception de ‘structure ’et ‘enum’

Type Valeur

Type	Bytes	.net type	description
byte	1	Byte	
bool	1	Boolean	
char	2	Char	
short	2	Int16	
int	4	Int32	

Type Valeur

uint	4	Uint32	
float	4	Single	i = 0.333F
double	8	Double	i = 0.333
decimal	8	Decimal	i = 0.333M
long	8	Int64	
ulong	8	Uint64	

Boolean

- 2 valeurs possibles true/false
- bool b = false; //OK
- bool b = 0; //compile-time error!

String

- String Literals verbatim:

```
string a = "hello world";        // hello world
string b = @"hello world";     // hello world
string c = "hello \t world";     // hello    world
```

Implicit / Explicit Conversions

- Implicit Conversion

```
short s = 3;
int i = s;  //implicit conversion
```

- Explicit Conversion

```
int i = 3;
short s = i;              //compile-time error!
short s = (short)i;  //explicit conversion
```

conversion

- La classe ‘Convert’ des FCL possede des methodes statiques qui permettent les conversions comme:

  int a=Convert.ToInt32(“2”)

  Convert.ToDouble() …

Console.WriteLine

Console.WriteLine()

Permet d’afficher a l’écran (la console)

Console.Write(‘’ bonjour’’);

Console.WriteLine(“bonjour Mr {0} {1} “ + ‘’a bientôt ’’, ’’Mr’’,’’Dupond’’);

Console.Readline()

- Permet de lire une valeur String à partir de l'écran

```
Console.Write("Saisir votre nom: ");
string userName = Console.ReadLine();
Console.WriteLine("Bonjour {0} bonne journée",
    userName);
```

Variable et Constantes

Variables

- Elle sera stockée dans le Stack
- A un nom qui sera différent de mots clefs du framework
- Il s'agit de variable de méthode (locale)
- Il faut obligatoirement lui donner une valeur avant de l'utiliser
- Pas de valeur par défaut

Variables

- Définition et ensuite utilisation

```
using System;
class Entree {
    static void Main() {
    int i = 5;  //définition et  initialisation   Console.WriteLine("i
    = {0}", i);
    } }
```

Variables

- Mauvais exemple

```
using System;
class Entree{
 static void Main() {
 int i;          //definition pas d'initilalisation
  Console.WriteLine("i = {0}", i);

} }
```

Constante

- Une constante est une variable qui ne change jamais
- Symbolic Constant : une constante avec un nom

ex: const int boilingPoint = 100;

Les constantes

- Exemple problématique

```
using System;
class Constants {
  static void Main() {
  const int boilingPoint = 100; //définition
  boilingPoint = 212;           //erreur!
  } }
```

Enumeration

- C’est une série de constantes
  Par exemple:(ça fait long et pas lisible!)
  const int insideTheFreezer = -18;
  const int freezingPoint = 0;
  const int waterAnomaly = 4;
  const int coldEvenings = 10;
  const int hotDays = 35;
  const int boilingPoint = 100;

Énumération

- La solution

```
enum Temperatures {
insideTheFreezer = -18,
freezingPoint = 0,
waterAnomaly = 4,
coldEvenings = 10,
hotDays = 35,
boilingPoint = 100 };
```

énumération

- Par défaut le type est integer
- enum Temperatures : int {
  insideTheFreezer = -18,
  freezingPoint = 0,
  waterAnomaly = 4,
  coldEvenings = 10,
  hotDays = 35,
  boilingPoint = 100 };

Enumération

Exemple d'utilisation

```
Console.WriteLine(''temperature normale  "
+(int)Temperatures.waterAnomaly +
" degrés Celsius");
```

Opérateurs

- Opérateurs mathématiques:

+ - * / %

Ex: int a = 4 + 3;

- += -= *= /=

Ex:a += 3; →a=7

- Increment and Decrement Operators

a++; a-- //postfix increment

++a; --a //prefix increment

Operator

- Exemple
- void increment() {
  int a = 7;
  int b;
  b = a++;
  Console.WriteLine(“Postfix – a: {0}, b: {1}”, a, b);
  b = ++a;
  Console.WriteLine(“Postfix – a: {0}, b: {1}”, a, b); }

Opérateur de comparaison

- const int a = 4;
  const int b = 3;
- Retourne False ou True
- a == b;
  a != b;
  a > b;
  a >= b;
  a < b;
  a <= b;

Logical Operators

- (a > b) && (b > a);
- (a > b) | | (b > a);
- !(a > b);

condition

- On note 3 facons d’ecrire une condition :

  condition if

  condition switch

  condition ternaire

Conditions if

- void Comparer(int a ,int b) {
  if (a > b)
  Console.WriteLine(“le plus grand: {0}”, a);
  else if (a < b)
  Console.WriteLine(“le plus grand: {0}”, b);
  else
  Console.WriteLine(“les 2 sont égaux”);
}

Condition switch

- Le switch se fait sur du string ou int

```
void FonctoinAnnuaire() {
string tel = Console.ReadLine();

switch (tel) {
case ‘’samu":
Console.WriteLine(« le tel est le 18!!!!");
break;
case’’ pompier":
Console.WriteLine(« le tel est le 100!!!!");
break;
```

Conditions

```
default:
    Console.WriteLine(‘’ vous n’avez choisi aucun tel!!!‘’ );
    break;
} }
```

- Le default est optionnel mais le break est obligatoire

Condition switch

- switch (tel) {

```
case “samu”:
case “Samu”: Console.WriteLine(“le tel ……”);
        break;
}
```

Condition ternaire

- Condition ternaire
- void comparer(int a,int b) {

  int max = (a > b) ? a : b; Console.WriteLine(“Max({0}, {1}) = {2}”, a, b, max); }

Les boucles

- Méthode goto

- ```
  void compteur() {
  int i = 0;
  const int MAX = 10;
  label:
  if (i < MAX) {
  i++;
  Console.Write(".");
  goto label;          }
  Console.WriteLine("\nFin!"); }
  ```
  ```

Les boucles

- Méthode while(condition)

- ```
  void compteur() {
  int i = 0;
  const int MAX = 10;
  while (i < MAX) {
  i++;
  Console.Write(".");  }
  Console.WriteLine("\nFin!"); }
  ```
  ```

Les boucles

- Méthode do while

- ```
  void compteur() {
  int i = 0;
  const int MAX = 10;
  do {
  i++;
  Console.Write(".");  }
  while (i < MAX);
  Console.WriteLine("\nFin!"); }
  ```
  ```

Les boucles

- Méthode for

- void compteur() {
  const int MAX = 10;
  for (int i = 0; i < MAX; i = i + 1)
  Console.Write(“.”);
  Console.WriteLine("\nFin!"); }

Tableaux simples

- C'est une collection d'éléments de type prédéfini
- Le nombre d'éléments est fixe et non dynamique

- La valeur par défaut est 0 pour tous les éléments

- Déclaration (+initialisation):

```
int [] Tab; Tab=new int[5];
int [] Tab=new int [5];
int [] Tab=new int[5]{1,2,3,4,5};
int [] Tab=new int[]{1,2,3,4,5};
int [] Tab={1,2,3,4,5};
```

Tableaux simples

- Initialisation :

  ```
  for(int i=0;i<5;i++)
  Tab [i]=i;
  ```

- Accéder aux éléments

  ```
  for(int i=0;i<5;i++)
  Console.WriteLine(Tab[i]);
  ```

REMARQUES

Exercice demander a l'utilisateur la taille de la collection et les membres que vous afficherez

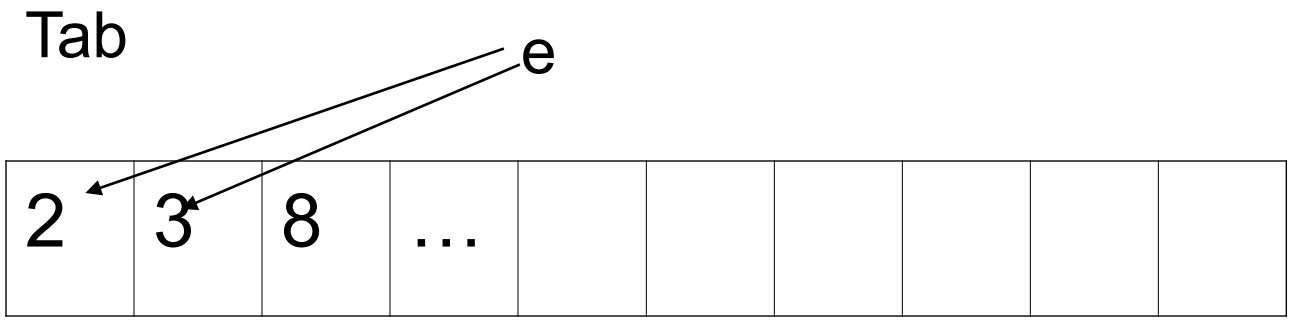
Foreach

- foreach permet de passer sur une collection d'éléments de même type de façon itérative sans en connaître la taille

- foreach(int e in Tab)

  Console.WriteLine(e);

  //e est la valeur de chaque élément et non pas un //index



L’index n'est pas nécessaire et
e est la valeur et non pas
l'index