Module 2:

Vue d'ensemble de C

Vue d'ensemble

- Structure d'un programme C #
- Opérations d'entrée/sorties de base
- Pratiques recommandées
- Compilation, exécution et débogage

Structure d'un programme C

- Hello World
- Une Classe
- La méthode principale
- La directive “Using” et l'espace de noms “System”
- Démonstration: Utilisation de Visual Studio pour créer un programme C #

Hello, World

```
using System;

class Hello
{
  public static void Main()
  {
    Console.WriteLine("Hello, World");
  }
}
```

La Classe

- Une application C# est une collection de classes, structures et de types
- Une classe est un ensemble de données et methods
- Syntaxe :

```
Class nom
{
        ...
}
```

- Une application C# peut être constituée de plusieurs fichiers
- Une classe s'écrit dans un seul fichier.

La principale méthode : “Main”

- Lors de l'écriture du “Main”, vous devez:
  - Utilisez un "M“ majuscule : "Main"
  - Le “**Main**” est le point d'entrée du programme
  - Déclarer **Main** comme ceci : **public static void Main**
- Il ne peut y avoir qu’une seule méthode static void Main dans une application.
- Lorsque “Main” se termine, l'application se ferme.

La directive "Using" et l'espace de noms "System"

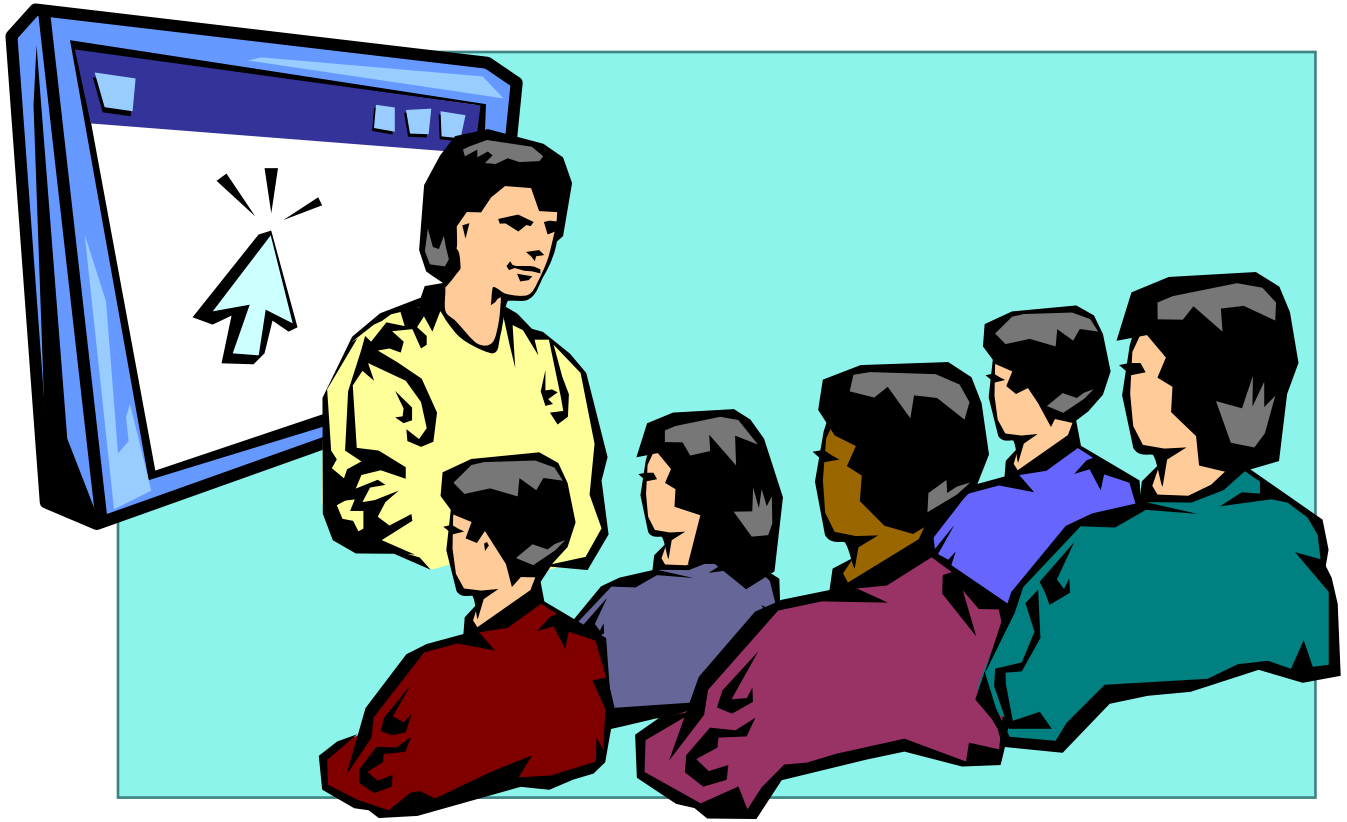
- Le Framework .NET fournit de nombreuses classes d'utilitaires
  - Organisé en espaces de noms
- System est le namespace *(l'espace de noms)* principal du Framework .Net
- Une classe se définit par son espace de noms

```
System.Console.WriteLine("Bonjour le monde");
```

- La directive using permet d'ajouter un namespace et donc de ne plus devoir le spécifier à chaque utilisation le namespace

```
using System;
...
Console.WriteLine ("Bonjour le monde");
```

Démonstration:
Utilisation de Visual Studio pour créer un programme C



Opérations d'entrée/sortie de base

- La classe Console
- Les méthodes “Write” et “WriteLine”
- Les méthodes “Read” et “ReadLine”

La classe Console

- Permet d'accéder à l'entrée standard, la sortie standard, et les flux d'erreur standard :
  - Entrée standard - clavier
  - Sortie standard - écran
  - Erreur standard – écran
- N'a de sens que pour les applications console
- *Pour info: Tous les flux peuvent être redirigés*

Les méthodes Write et WriteLine

- Console.Write et Console.WriteLine écrivent des informations sur l'écran de la console
  - **WriteLine** ajoute un retour à la ligne à la fin.

```
System.Console.WriteLine("Bonjour le monde");
```

- Une chaîne formatée et les paramètres correspondant peuvent être utilisés :

```
System.Console.WriteLine("Bonjour {0}, Comment allez vous?", Name);
```

Les méthodes Read et ReadLine

- Console.Read et Console.ReadLine lisent l'entrée de l'utilisateur
  - **Read** lit le caractère suivant
  - **ReadLine** lit la ligne d'entrée entière

```
str = System.Console.ReadLine();
```

Pratiques recommandées

- Applications commentées
- Génération de la documentation XML
- Démonstration: création et affichage de la documentation XML
- Gestion des exceptions

Applications Commentées

- Les commentaires sont importants
  - Une application bien commentée permet à un développeur de comprendre la structure de la demande

- Commentaires sur une ligne

```
/ / Obtenir le nom de l'utilisateur
Console.WriteLine ("Quel est votre nom?");
name = Console.ReadLine ();
```

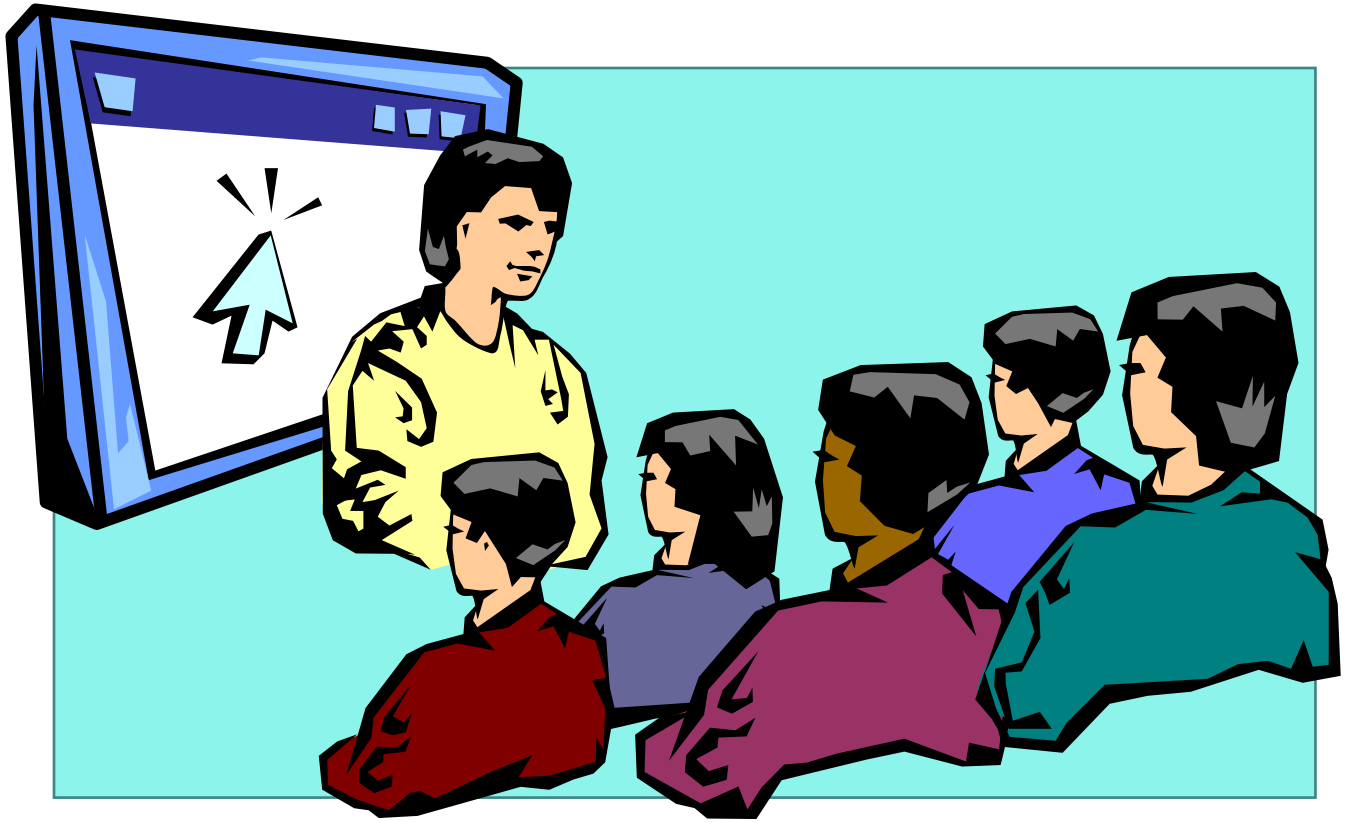
- Commentaires sur plusieurs lignes

```
/ * Trouver la racine supérieure de l'
   équation quadratique * /
x = (...);
```

Génération de la documentation XML

```
/// <summary> La classe Hello imprime un message
/// d'accueil sur l'écran
/// </summary>
Classe Hello
{
  /// <remarks> Nous utilisons la console pour afficher
  /// un message.
  /// Pour plus d'informations sur WriteLine, voir
  /// <seealso cref="System.Console.WriteLine"/>
  /// </remarks>
  public static void Main ()
  {
    Console.WriteLine ("Bonjour le monde");
  }
}
```

Démonstration:
Création et Affichage de la documentation XML



Gestion des exceptions

```
using System;
public class Hello
{
  public static void Main (string [] args)
  {
     try {
         Console.WriteLine (args [0]);
     }
     catch (Exception e) {
         Console.WriteLine ("Exception à
        ➥{0} ", e.StackTrace);
     }
  }
}
```

Compilation, exécution et débogage

- Invoquer le compilateur
- Exécution de l'application
- Démonstration: compilation et exécution d'un programme C #
- Débogage
- Démonstration: Utilisation du débogueur Visual Studio
- Les outils du SDK
- Démonstration: utilisation ILDASM

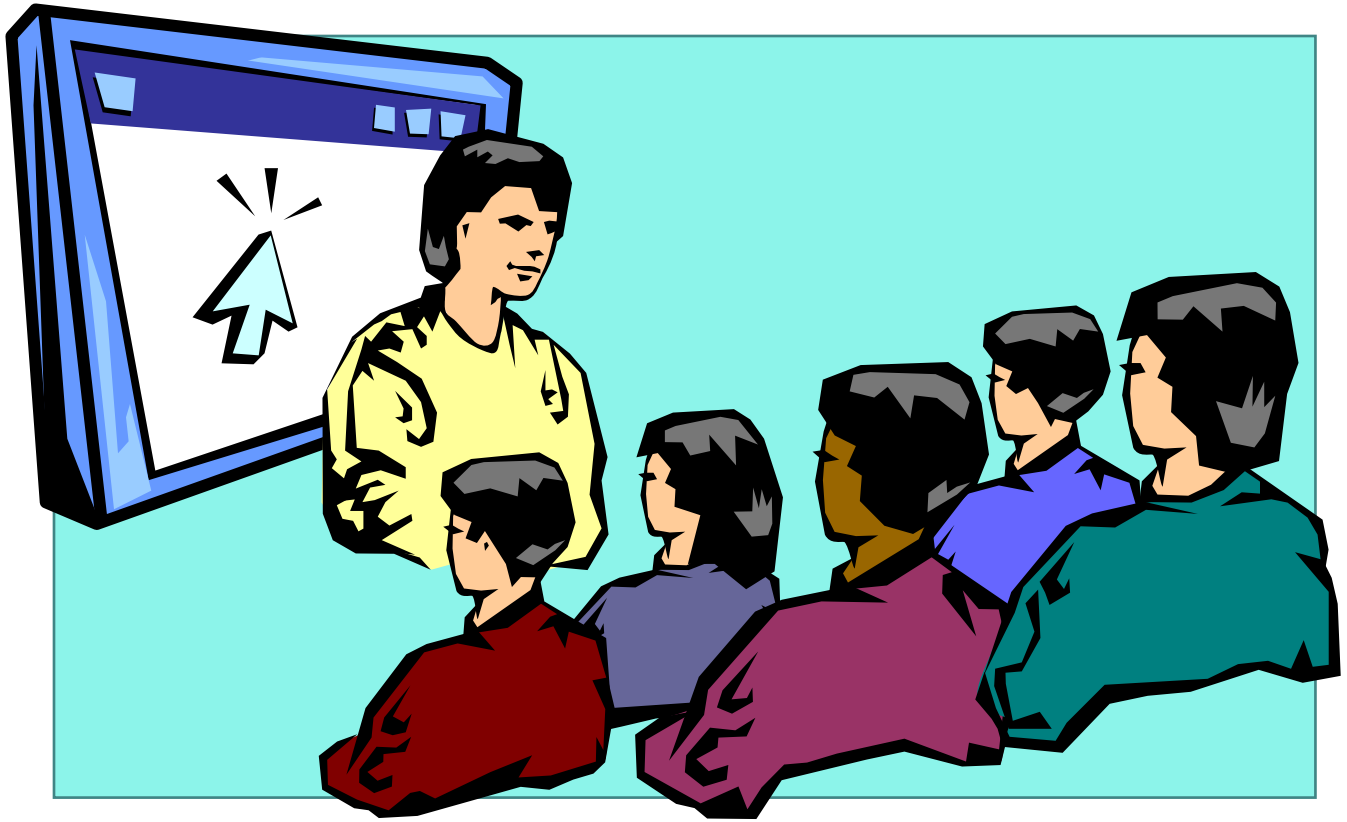
Invoquer le compilateur

- Commutateurs du compilateur principaux
- Compilation depuis la ligne de commande
- Compilation depuis Visual Studio
- Localisation des erreurs

Exécution de l'application

- Exécution depuis la ligne de commande
  - Tapez le nom de l'application
- Exécution depuis Visual Studio
  - Cliquez **Exécuter sans débogage** sur le menu **Debug**

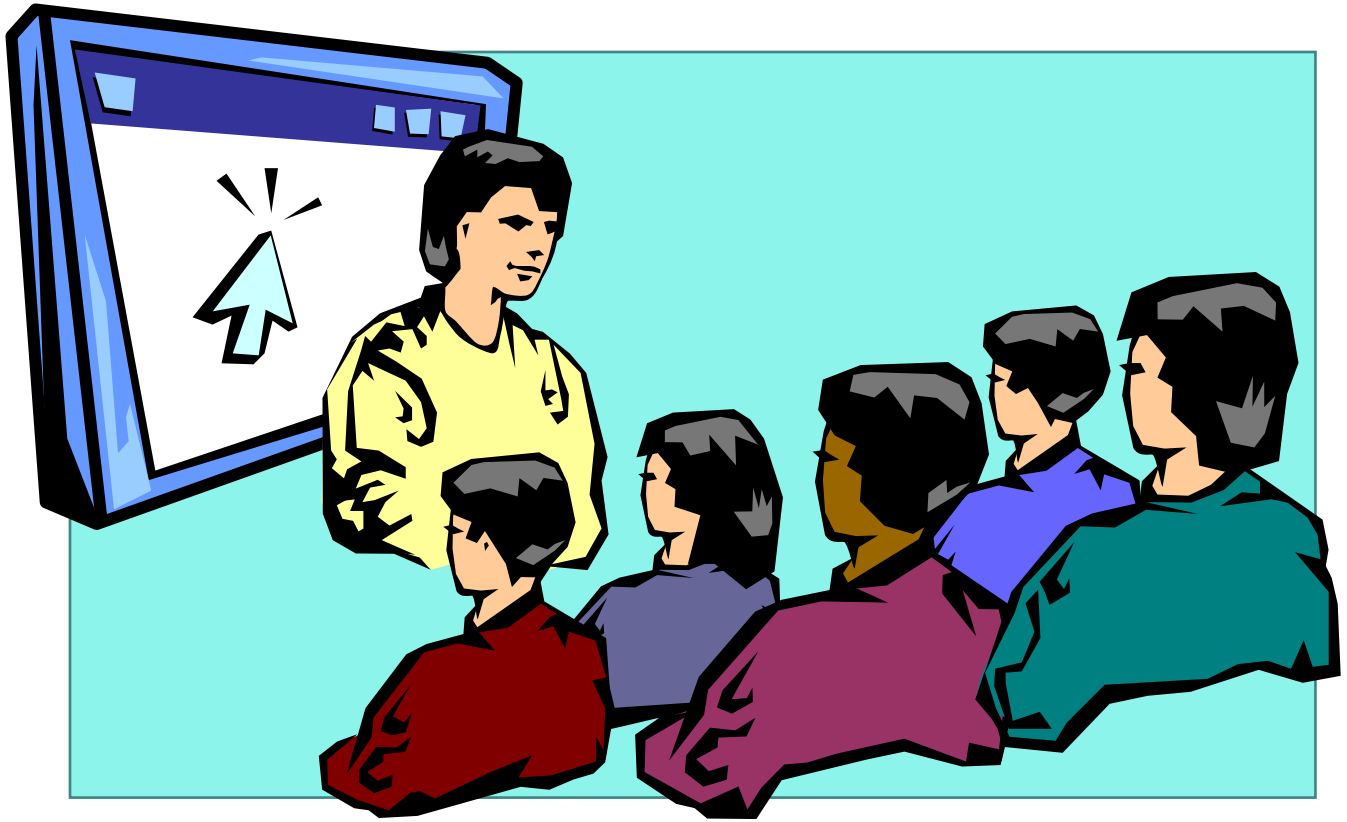
Démonstration:
Compilation et exécution d'un programme C



Débogage

- Exceptions et JIT débogage
- Le débogueur Visual Studio
  - Points d'arrêt et de réglage des montres
  - Progression dans le code
  - Examen et modification des variables

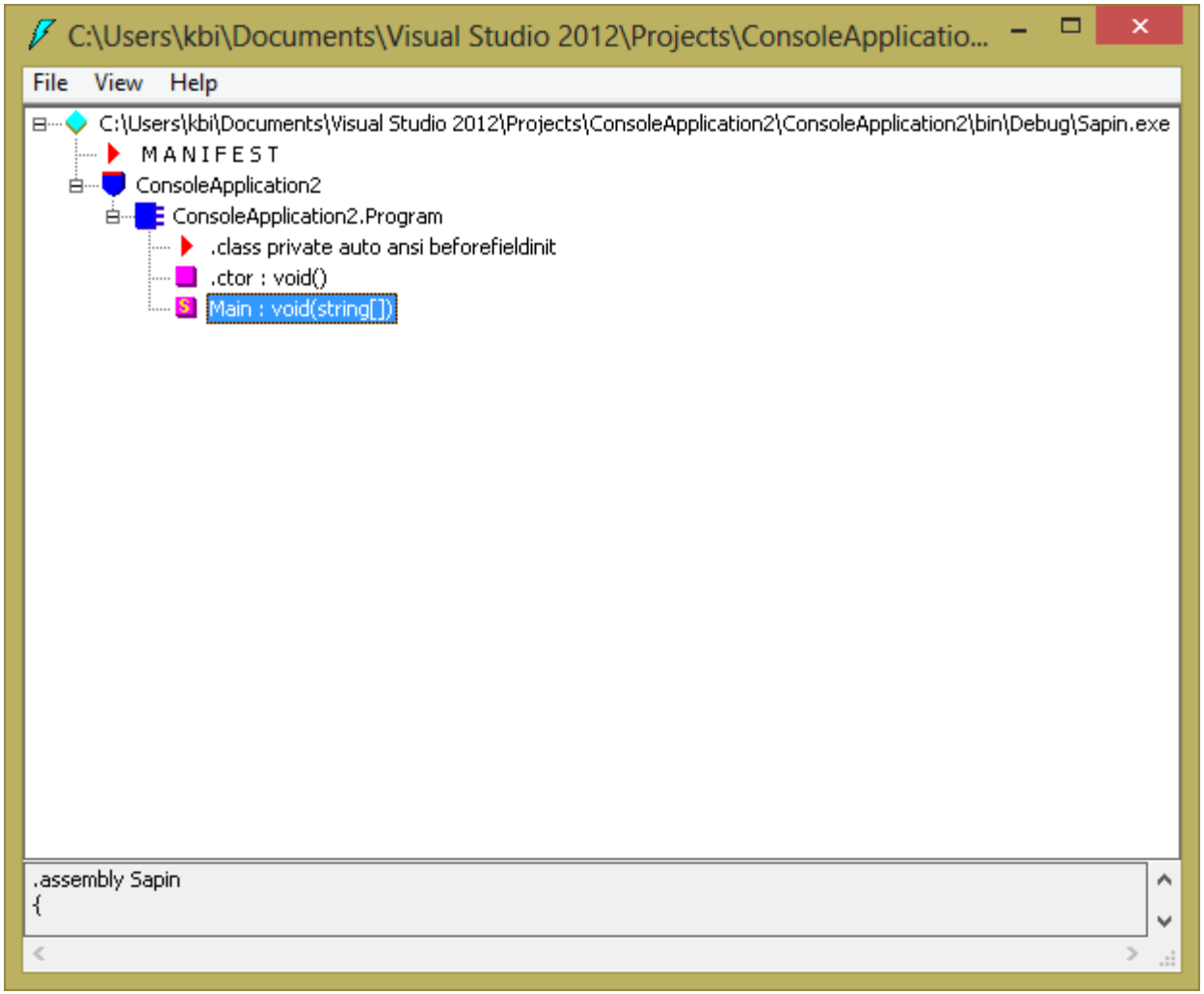
Démonstration:
Utilisation du débogueur Visual Studio



Les outils du SDK

- Outils et utilitaires généraux
- Windows Forms outils de conception et utilitaires
- Outils de sécurité et utilitaires
- Configuration, utilitaires et outils de déploiement

Démonstration: Utilisation ILDASM



Lab 2.1:
Création d'un simple programme C