

# Les bases de C#

Programmation « procédurale » en C#

# Généralités sur la programmation

## Algorithmie

- Solution « informatique » relative à un problème
- Suite d'actions (instructions) appliquées sur des données
- 3 étapes principales :
  1. saisie (réception) des données
  2. Traitements
  3. restitution (application) des résultats

## Programme

- Transcription d'un algorithme avec une syntaxe prédéfinie
- C# - Proche de Java / Delphi (créé par le père de Delphi - [http://fr.wikipedia.org/wiki/C\\_sharp](http://fr.wikipedia.org/wiki/C_sharp))
- Même principes fondamentaux que les autres langages à objets (Delphi, Java, etc.)
- Visual C# introduit des fonctionnalités supplémentaires : la programmation visuelle et la programmation événementielle

# Mode d'exécution d'un programme

Langage interprété : + portabilité application ; - lenteur (R, VBA, ...)

Langage compilé : + rapidité ; - pas portable

(solution possible : write once, compile anywhere ; ex. Lazarus)

Langage pseudo-compilé : + portabilité plate-forme ; - lenteur (?)

(principe : write once, run anywhere ; ex. Java)

Comment se situent C# et la plate-forme .NET ?

# Etapes de la conception d'un programme (Génie Logiciel)

1. **Fixer les objectifs et détermination des besoins** : que doit faire le logiciel, dans quel cadre va-t-il servir, quels seront les utilisateurs types ? On le rédige souvent avec le commanditaire du logiciel (Remarque : commanditaire = maître d'ouvrage ; réalisateur = maître d'œuvre)
2. **Conception et spécifications** : quels sont les fonctionnalités du logiciel, avec quelle interface ?
3. **Programmation** : modélisation et codage
4. **Tests** : obtient-on les résultats attendus, les calculs sont corrects, y a-t-il plantage et dans quelles circonstances ? (tests unitaires, tests d'intégration, etc.)
5. **Déploiement** : installer le chez le client (vérification des configurations, installation de l'exécutable et des fichiers annexes, etc.)
6. **Maintenance** : corrective, traquer les bugs et les corriger (patches) ; évolutive (ajouter des fonctionnalités nouvelles au logiciel : soit sur l'ergonomie, soit en ajoutant de nouvelles procédures)

# Notre environnement de développement

## L'EDI SHARPDEVELOP

The image shows a screenshot of the SharpDevelop website and its IDE interface. The website header includes the text "develop" and "ic#code". The navigation menu contains links for Home, Download, Changes, Features, Forum, Wiki, Tech Notes, Bug Tracker, Team Blogs, Contribute, and Feature Tour. The main content area is titled "The Open Source Development Environment for .NET" and contains a description of SharpDevelop as a free IDE for C#, VB.NET, and Boo projects. It also includes links for Announcements, What's New, News History, Public Relations, and Contact Us. A sidebar on the left features a "Dissecting A C# Application" article and a "SOURCEFORGE.NET" logo. The IDE window, titled "Hello SharpDevelop - SharpDevelop", shows a menu bar (File, Edit, View, Project, Build, Debug, Search, Format, Tools, Window, Help) and a toolbar. The main workspace displays a project named "SharpDevelop2 Rocks!" with a tree view showing "Affirmation" and "View Code". A context menu is open over the "View Code" item, listing options like "Bring to Front", "Send to Back", "Align to Grid", "Show tab order", "Lock Controls", "Cut", "Copy", "Paste", "Delete", and "Properties". The Properties window on the right shows the "AffirmButton" control with various properties such as "UseWaitCursor", "Behavior", "Data", and "Design". The status bar at the bottom indicates "ln 2 col 2 ch 2 INS".

develop ic#code

Home | Download | Changes | Features | Forum | Wiki | Tech Notes | Bug Tracker | Team Blogs | Contribute | Feature Tour

### The Open Source Development Environment for .NET

[Announcements](#)  
[What's New](#)  
[News History](#)  
[Public Relations](#)  
[Contact Us](#)

**#develop** (short for SharpDevelop) is a free IDE for C#, VB.NET and Boo projects on Microsoft's .NET platform. It is open-source, and you can download both sourcecode and executables from this site. In addition, you can find the latest information and changes on #develop, as well as get in touch with the team in the [forum](#).

We are going to add more content (and new releases) over the coming months, so sign up for our [announcement list](#)! Also, we want to invite you to participate in the [forums](#), discussing SharpDevelop - its features, feature requests and of course [bugs](#).

Dissecting A C# Application  
Inside SharpDevelop

SOURCEFORGE.NET

Hello SharpDevelop - SharpDevelop

File Edit View Project Build Debug Search Format Tools Window Help

Tools

- ASCII Table
- C# Documentation Tags
- Licenses
- XSL-T
- General
- Clipboard Ring
- Windows Forms
  - Pointer
  - Button
  - CheckBox
  - ComboBox
  - Label
  - RadioButton
  - TextBox
  - CheckedListBox
  - DateTimePicker
  - DomainUpDown
  - FlowLayoutPanel
- Data
- Components
- Custom Components

Projects Tools

Source Design

Output

ln 2 col 2 ch 2 INS

# Création d'une application console sous SharpDevelop

The screenshot displays the SharpDevelop IDE interface for a project named 'AppliConsole'. The 'Projects' pane on the left shows a tree view with 'Solution AppliConsole' at the top, followed by 'AppliConsole', 'References', 'Properties', 'app.config', and 'Program.cs'. Red arrows point from text boxes to these elements: 'Solution' points to the solution folder, 'Projet' points to the 'AppliConsole' project folder, and 'Classe « principale »' points to the 'Program.cs' file.

The main editor window shows the source code for 'Program.cs':

```
1  /*  
2  * Created by SharpDevelop.  
3  * Un exemple d'application console  
4  */  
5  using System;  
6  
7  namespace AppliConsole  
8  {  
9      class Program  
10     {  
11     public static void Main(string[] args)  
12     {  
13         Console.WriteLine("Hello World!");  
14  
15         // TODO: Implement Functionality Here  
16  
17         Console.Write("Press any key to continue . . . ");  
18         Console.ReadKey(true);  
19     }  
20 }  
21 }
```

A red bracket on the right side of the code block highlights the 'Main' method, with a text box labeled 'Programme (procédure) principale' pointing to it.

The 'Properties' pane on the right shows the 'Misc' section with the following values:

Property	Value
Build action	Compile
Copy to output direc	Never
Custom Tool	
File name	C:\Users\Maison\Document

The 'Output' pane at the bottom shows 'Debug' mode. The status bar at the bottom indicates 'Build finished successfully.' and 'In 3 col 36 ch 36'.

# Aide en ligne - MSDN

On ne peut pas programmer sans aide !!!

Recherche par mots-clés très pertinent

Accueil Library Formation Téléchargements Support technique Communauté Connexion | France - Français

Rechercher sur MSDN avec Bing

MSDN Library

- Langages et outils de développement
- Visual Studio 2005
- Documentation Visual Studio
- Visual C#
- Référence C#
- Mots clés C#
- Types
  - Types référence
  - class
  - delegate
  - interface
  - object
  - string**

## string (Référence C#)

Visual Studio 2005 | Autres versions | 3 sur 5 ont trouvé cela utile - Évaluez ce sujet

Le type **string** représente une chaîne de zéro ou plusieurs caractères Unicode. **string** est un alias de **String** dans le .NET Framework.

Bien que **string** soit un type référence, les opérateurs d'égalité (== et !=) sont définis pour comparer les valeurs d'objets **string**, pas de références. Le test d'égalité des chaînes est ainsi plus intuitif. Par exemple :

```
string a = "hello";
string b = "h";
// Append to contents of 'b'
b += "ello";
Console.WriteLine(a == b);
Console.WriteLine((object)a == (object)b);
```

Cela affiche "True", puis "False" parce que le contenu des chaînes est équivalent, mais a et b ne font pas référence à la même instance de chaîne.

L'opérateur + concatène les chaînes :

```
string a = "good " + "morning";
```

Cela crée un objet de chaîne qui contient "good morning".

Les chaînes sont *immuables* – le contenu d'un objet chaîne ne peut pas être modifié une fois que l'objet a été créé, même si la syntaxe peut laisser croire que la modification est possible. Par exemple, lorsque

## Quelques types simples

Alias vers un type .NET (objet); int = Int32, double = Double, string = String, etc.

Entier : **int** ; opérateurs : +, -, \*, /, %

Réel : **double**, op : +, -, \*, /

Booléen : **bool** (**true**, **false**); op : !, &&, ||

Chaîne de caractères : **string** ; op : méthodes associées au type String [ex. Substring(), etc.]

Remarque 1 : TRANSTYPAGE (type vers chaîne et inversement)

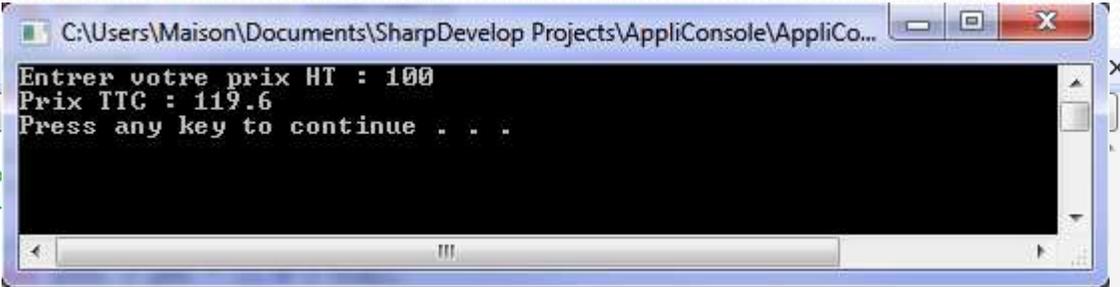
- nom\_de\_variable.**ToString**()
- type\_destination.**Parse**(chaîne de caractères)

Remarque 2 : Opérateurs de comparaison

- Compare des éléments de même type, et renvoie un booléen
- **==, !=, >, >=, <, <=**

# Déclaration, affectation, entrées / sorties

```
Program.cs
AppliConsole.Program
1  /*
2  * Created by SharpDevelop
3  * Un exemple d'applicati
4  */
5  using System;
6
7  namespace AppliConsole
8  {
9      class Program
10     {
11         public static void Main(string[] args)
12         {
13
14             //constante taux de TVA
15             const double tva = 0.196;
16
17             //invitation console
18             Console.WriteLine("Entrer votre prix HT : ");
19
20             //chaîne saisie à la console, représente le PHT
21             string str_pht = Console.ReadLine();
22
23             //transtypage
24             double pht = double.Parse(str_pht);
25
26             //marche aussi
27             //double pht = double.Parse(Console.ReadLine());
28
29             //calcul
30             double pttc = pht * (1.0 + tva);
31
32             //affichage
33             Console.WriteLine("Prix TTC : " + pttc.ToString());
34             Console.WriteLine("Press any key to continue . . . ");
35             Console.ReadKey(true);
36         }
37     }
38 }
39 }
```



Déclaration de constante + affectation

E/S : sortie écran

Déclaration de variable + E/S : saisie console + affectation

Déclaration de variable + Affectation + Transtypage

Déclaration de variable + Calcul + Affectation

Concaténation + affichage

# Structures algorithmiques : branchement conditionnel

Condition : booléen  
Opération de comparaison  
Peut être complexe (combinaison de conditions)

if (condition)  
instruction si condition vraie;  
else  
instruction si condition fausse;

La partie « else » est facultative

```
if (condition)
{
    bloc d'instructions si condition vraie;
    ...
}
else
{
    bloc d'instructions si condition fausse;
    ...
}
```

```
namespace AppliConsole
{
    class Program
    {
        public static void Main(string[] args)
        {
            ...
            Console.WriteLine("Entrez votre age : ");
            int age = int.Parse(Console.ReadLine());

            if (age > 25)
                Console.WriteLine("sénile");
            else
                Console.WriteLine("jeunot");

            Console.WriteLine("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

# Structures algorithmiques : branchement multiple

Entier, caractère ou chaîne de caractères

```
switch (variable)
{
    case valeur_1:
        instruction(s) pour valeur_1;
        break;
    case valeur_2:
        instructions(s) pour valeur_2;
        break;
    ...
    default :
        instruction(s) pour cas par défaut;
        break;
}
```

La partie « default » est facultative.  
« break » fait sortir de la structure  
Même dans « default » il faut mettre break

```
Console.Write("Type de voiture (berline, luxe, utilitaire) : ");
string typeVehicule = Console.ReadLine();

double taxe = 0;

switch(typeVehicule)
{
    case "luxe" :
        taxe = 100.0;
        break;
    case "utilitaire" :
        taxe = 50.0;
        break;
    default:
        taxe = 75.0;
        break;
}

Console.WriteLine("Taxe : " + taxe.ToString());
```

```
Console.Write("Type de voiture (berline, initiale, luxe, utilitaire) : ");
string typeVehicule = Console.ReadLine();

double taxe = 0;

switch(typeVehicule)
{
    case "luxe" :
    case "initiale":
        taxe = 100.0;
        break;
    case "utilitaire" :
        taxe = 50.0;
        break;
    default:
        taxe = 75.0;
        break;
}

Console.WriteLine("Taxe : " + taxe.ToString());
```

# Structures algorithmiques : boucle « pour »

2 formes possibles

```
for (initialisation; test continuation; suite)
    instruction;
```

```
for (initialisation; test continuation; suite)
{
    instruction(s);
}
```

Ex. somme de termes au carré

```
Console.WriteLine("Entrez une valeur entière : ");
int n = int.Parse(Console.ReadLine());

double s = 0.0;
for (int i = 1; i <= n; i++)
    s += Math.Pow(i,2);

Console.WriteLine("S : " + s.ToString());
```

« break » peut casser la boucle (nous faire sortir directement)  
« continue » saute les instructions et passe à l'itération suivante

Attention, for peut faire plus qu'une simple boucle indicée

Ex. Trouver le plus petit « n » pair et > 0, tel que  $(2^n \geq v)$

```
Console.Write("Entrer une valeur : ");
int v = int.Parse(Console.ReadLine());

int n;

for (n = 0; Math.Pow(2,n) < v; n = n + 2)
    Console.WriteLine(n.ToString() + ":" + Math.Pow(2,n).ToString());

Console.WriteLine("n = " + n.ToString());
```

# Structures algorithmiques : boucle « tant que »

```
while (condition vraie)  
  instruction;
```

```
while (condition vraie)  
{  
  instruction(s);  
}
```

```
do  
{  
  instruction(s);  
} while (condition vraie);
```

« break » et « continue » jouent le même rôle

```
Console.Write("Entrer une valeur : ");  
int v = int.Parse(Console.ReadLine());  
  
int n = 0;  
  
while (Math.Pow(2,n) < v)  
  n += 2;  
  
Console.WriteLine("n = " + n.ToString());
```

Ex. Trouver le plus petit « n » pair et  $> 0$ , tel que  $(2^n \geq v)$

```
Console.Write("Entrer une valeur : ");  
int v = int.Parse(Console.ReadLine());  
  
int n = 0;  
  
while (true)  
{  
  n += 2;  
  
  if (Math.Pow(2,n) >= v)  
    break;  
}  
  
Console.WriteLine("n = " + n.ToString());
```

*Si on aime faire compliqué...*

# Organisation des programmes : procédures et fonctions

Objectif (noble) : Associer dans la même structure du code associé à une fonctionnalité.

Objectif (pragmatique) : Eviter les copier/coller dans le code, maximiser la réutilisation

```
type_de_retour nom_de_fonction(arguments)
{
    instructions;
    return valeur à renvoyer;
}
```

- 1 argument = type nom de paramètre
- Si plusieurs arguments, les séparer par des « , »
- Procédure : type de retour = `void` ; `return` n'est pas nécessaire dans ce cas
- `return` provoque la sortie directe de la fonction dès qu'elle est invoquée
- fonction sans argument, on doit conserver quand même les parenthèses `()` lors de la définition et lors de l'appel

```
namespace TestFonctions
{
    class Program
    {
        /// <summary>
        /// Calcul du prix TTC
        /// </summary>
        /// <param name="ht">prix ht</param>
        /// <param name="niv_tva">niveau de tva</param>
        /// <returns>prix ttc</returns>
        public static double calcTTC(double ht, double niv_tva)
        {
            double tmpTtc = ht * (1.0 + niv_tva);
            return tmpTtc; //on peut aussi effectuer le calcul ici
        }

        public static void Main(string[] args)
        {
            //constante taux de TVA
            const double tva = 0.196;

            //invitation console
            Console.WriteLine("Entrer votre prix HT : ");

            double pht = double.Parse(Console.ReadLine());

            //calcul
            double pttc = calcTTC(pht,tva);

            //affichage
            Console.WriteLine("Prix TTC : " + pttc.ToString());

            Console.WriteLine("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

# Procédures et fonctions, passages de paramètres (1)

## Passage par valeur



```
a : 10
b : 15
Press any key to continue . . .
```

```
class Program
{
    public static void echangeValeur(int p, int q)
    {
        int tmp = q;
        q = p;
        p = tmp;
    }

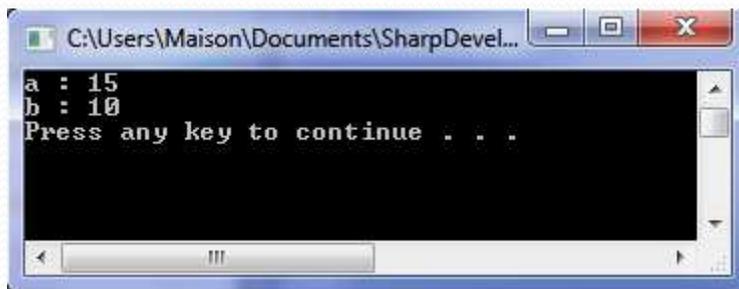
    public static void Main(string[] args)
    {
        int a = 10;
        int b = 15;

        echangeValeur(a,b);

        Console.WriteLine("a : " + a.ToString());
        Console.WriteLine("b : " + b.ToString());

        Console.Write("Press any key to continue . . . ");
        Console.ReadKey(true);
    }
}
```

## Passage par référence



```
a : 15
b : 10
Press any key to continue . . .
```

```
class Program
{
    public static void echangeReference(ref int p, ref int q)
    {
        int tmp = q;
        q = p;
        p = tmp;
    }

    public static void Main(string[] args)
    {
        int a = 10;
        int b = 15;

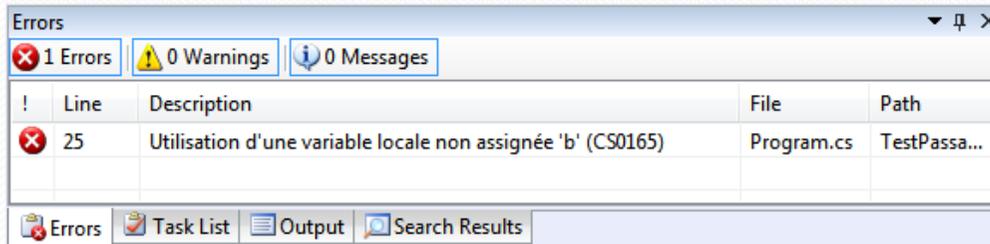
        echangeReference(ref a,ref b);

        Console.WriteLine("a : " + a.ToString());
        Console.WriteLine("b : " + b.ToString());

        Console.Write("Press any key to continue . . . ");
        Console.ReadKey(true);
    }
}
```

# Procédures et fonctions, passages de paramètres (2)

Tentative de passage par référence, le compilateur râle...



The screenshot shows the 'Errors' window in Visual Studio. It displays one error: 'Utilisation d'une variable locale non assignée 'b' (CS0165)' at line 25 of Program.cs. The error message is in French, indicating that a local variable 'b' is used without being assigned a value. The window also shows 0 warnings and 0 messages.

!	Line	Description	File	Path
✖	25	Utilisation d'une variable locale non assignée 'b' (CS0165)	Program.cs	TestPassa...

```
class Program
{
    public static void affecte(int p, ref int q)
    {
        q = p;
    }

    public static void Main(string[] args)
    {
        int a = 10;
        int b;

        affecte(a, ref b);

        Console.WriteLine("b : " + b.ToString());

        Console.Write("Press any key to continue . . . ");
        Console.ReadKey(true);
    }
}
```

```
class Program
{
    public static void affecte(int p, out int q)
    {
        q = p;
    }

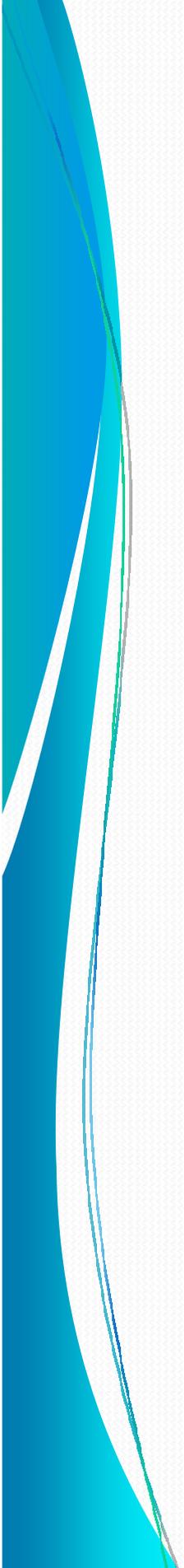
    public static void Main(string[] args)
    {
        int a = 10;
        int b;

        affecte(a, out b);

        Console.WriteLine("b : " + b.ToString());

        Console.Write("Press any key to continue . . . ");
        Console.ReadKey(true);
    }
}
```

Passage par référence sans  
valeur d'initialisation, utilisation  
du mot clé « out »



# FIN...

*Les mêmes concepts sont - à peu de choses près - présents dans tous les langages de programmation objet (Java, Delphi, C++,...)*