Les atrocités du C#

Consignes

Dans ce TP, vous ne devez rendre **uniquement** que les fichiers sources (`.cs`) de vos programmes.
La structure du rendu doit être comme ceci :

```
/rendu-tp-login_x.zip
   |-- login_x /
          |-- Bulls.cs
          |-- Motifs.cs
          |-- Training.cs
          |-- AUTHORS
          |-- README
```

`rendu-tp-login_x.zip`	nom de l'archive à rendre
`AUTHORS`	vous devez remplacer le contenu du fichier par une étoile `*`, un espace, puis votre login (ex : `login_x`) suivi d'un retour à la ligne
`README`	vous devez écrire dans ce fichier, tout commentaire sur le TP, votre travail ou bien plus généralement sur vos points forts/faibles. un fichier `README` vide sera considéré comme une archive sale (malus).

Il va de soi que toutes les occurrences de `login_x` sont à remplacer par **votre** login.

Tout le TP est à réaliser en mode console : `Fichier` ⇒ `Nouveau` ⇒ `Projet` ⇒ `Application Console`

TESTEZ VOTRE CODE !

In computer programming, a variable or scalar is a storage location and an associated symbolic name (an identifier) which contains some known or unknown quantity or information, a value. The variable name is the usual way to reference the stored value ; this separation of name and content allows the name to be used independently of the exact information it represents. The identifier in computer source code can be bound to a value during run time, and the value of the variable may thus change during the course of program execution.

– wikipedia

1 Training

Fichier à rendre : `Training.cs`

Dans cette section nous allons vous demander d'implémenter vos fonctions mathématiques préférées. **Les fonctions ne doivent pas être récursives, et vous devez respecter les prototypes donnés.**

base de construction

```
1 static void Main(string[] args)
2 {
3   // do stuff, instructions and functions calls here
4   Console.ReadLine(); // to see something (ask ACDC why)
5 }
```

fonctions

La fonction `MyFact` retourne la factorielle de `n`.

```
1 static int MyFact(int n) { /* FIXME */ }
```

La fonction `MyPow` retourne le nombre `x` à la puissance `n`.

```
1 static int MyPow(int x, int n) { /* FIXME */ }
```

La fonction `MyPgcd` retourne le pgcd de `a` et `b`.

```
1 static int MyPgcd(int a, int b) { /* FIXME */ }
```

La fonction `MyFibo` retourne le `n`-ième élément de la suite de Fibonacci.

```
1 static int MyFibo(int n) { /* FIXME */ }
```

La fonction `MySqrt` retourne l'approximation de la racine carrée d'un entier `n` en utilisant la méthode d'Héron [1]. On ne gardera qu'une précision de trois chiffres après la virgule.

```
1 static float MySqrt(int n) { /* FIXME */ }
```

La fonction `MyIsPalindrome` retourne vrai si la chaîne `str` est un palindrome, faux sinon.

```
1 static bool MyIsPalindrome(string str) { /* FIXME */ }
```

1. `http://www.google.com`

2 Motifs

Fichier à rendre : `Motifs.cs`

Comme le `Caml` n'est pas pas encore si loin, on va se refaire les motifs en affichage console. **Les fonctions ne doivent pas être récursives, et vous devez respecter les prototypes donnés.**

base de construction

```
static void Main(string[] args)
{
  // do stuff, instructions and functions calls here
  Console.ReadLine(); // to see something (ask ACDC why)
}
```

fonctions

La procédure `Square` affiche un carré de côté `n`, composé du caractère `c`.

```
static void Square (int n, char c) { /* FIXME */ }
```

La procédure `Alternate` affiche un carré de côté `n`, en alternant le caractère `c1` et `c2`.

```
static void Alternate (int n, char c1, char c2) { /* FIXME */ }
```

La procédure `Triangle` affiche un triangle de base `n`, composé du caractère `c`.

```
static void Triangle (int n, char c) { /* FIXME */ }
```

La procédure `Pyramid` affiche une pyramide de hauteur `n`, composé du caractère `c`.

```
static void Pyramid (int n, char c) { /* FIXME */ }
```

La procédure `Cross` affiche une croix de côté `n`, composé du caractère `c1` et `c2`.

```
static void Cross (int n, char c1, char c2) { /* FIXME */ }
```

exemples

```
#####       #@#@#       *          =         ?...?
#####       @#@#@       **        ===        .?.?.
#####       #@#@#       ***      =====       ..?..
#####       @#@#@       ****    =======      .?.?.
#####       #@#@#       *****  =========     ?...?

 5 #         5 # @       5 *        5 =       5 ? .
```

3 Bulls and Cows

Fichier à rendre : `Bulls.cs`

Pour notre premier programme nous allons implémenter l'ancêtre du jeu de Mastermind. Il s'agit d'un jeu à deux joueurs, dans lequel un joueur choisit une combinaison et l'autre joueur doit trouver cette combinaison, il dispose d'autant d'essais que nécessaire. A chaque essai le joueur ayant choisi la combinaison lui indique le nombre de bonnes valeurs bien placées dans son essai (les `Bulls`), et le nombre de bonnes valeurs mal placées (les `Cows`), afin de lui permettre d'avancer dans sa résolution.
Une fois le code trouvé, les joueurs échangent de rôle et on recommence. Une fois les deux tours terminés, le joueur qui a fait le moins d'essais gagne.

Nous allons faire la version numérique de celui-ci. Pour ce faire nous partirons sur une base de quatre chiffres (de 0 à 9). Dans un premier temps la combinaison à trouver sera en dur dans notre code, puis on demandera au joueur de la saisir, voir à l'ordinateur d'en générer une.

base de construction

```
static int code = 1337;

static void Main(string[] args)
{
  // do stuff, instructions and functions calls here
  Console.ReadLine(); // to see something (ask ACDC why)
}
```

fonctions

Écrivez une fonction qui récupère une combinaison entrée par l'utilisateur dans la console. La fonction doit afficher un texte explicite de saisie.

```
static int GetInput () { /* FIXME */ }
```

Écrivez une fonction qui indique à l'utilisateur le nombre de `Bulls` et `Cows` dans son choix. La fonction retournera `true` si le joueur a exactement la bonne combinaison, `false` sinon.

```
static bool ValidateInput (int my_try) { /* FIXME */ }
```

Écrivez une fonction qui servira de boucle de jeu. La fonction demandera une proposition à l'utilisateur tant que celui-ci ne trouve pas le code, elle l'informera également de sa progression. Une fois le code trouvé, la fonction affichera le nombre d'essais que le joueur a utilisé.

```
static void GameLoop () { /* FIXME */ }
```

Bonii

- `*` un mode deux joueurs avec pseudo
- `**` la création d'un code par l'ordinateur
- `***` une IA pour deviner le code du joueur
- `**` une version graphique en `Windows Forms`
- `****` une version 3D en `DirectX` (avec `Shaders` tant qu'à faire)

Annexes

variables

```
int a = 40;                         // an integer
a += 2;                             // add 2 to a (now equal 42)
float b = 13.37;                    // a float
char c = 'p';                       // a character
string s = "code me i'm famous" // a string
```

typage

```
char c = 'b';
int i = (int) c;                    // cast
string s = i.ToString();            // convert
```

conditions

```
==  // equality
<   // lower than
<=  // lower or equal to
>   // greater than
>=  // greater or equal to
```

```
&&  // logical and
||  // logical or
!   // logical not
```

alternatives

```
if (conditions) {
  // instructions when true
} else {
  // instructions when false
}
```

fonctions

```
<r_type> <function_name>(<p_type_1> <name_1>, ...) {
  // instructions
  return <value of rtype>
}
```

```
int multiply(int a, int b) {
  int c = 0;
  c = a * b;
  return c;
}
```

boucles

syntaxe

```
while (condition) {
  // do stuff
}
```

```
do {
  // stuff
} while (conditions)
```

```
for (init ; conditions ; increment) {
  // do stuff
}
```

exemple

```
for (int i = 0; i < 10 ; ++i) {
  // do stuff
}
```

```
int i = 0;
while (i < 10) {
  // do stuff
  ++i;
}
```

console

affichage

```
int a = 23;
Console.Write("chaine");            // display "chaine"
Console.WriteLine();                // display an end of line (\n)
Console.WriteLine(a.ToString());    // display "23"
Console.WriteLine("{0}={1}+{2}", 32, 10, 26);

Console.ForegroundColor = ConsoleColor.Magenta;
Console.ResetColor();
Console.Clear();
```

lecture

```
int i;
string str;
str = Console.ReadLine();
i = Convert.ToInt32(str);
```