TP C#2 :

1 Consignes de rendu

A la fin de ce TP, vous devrez rendre une archive respectant l'architecture suivante :

```
rendu-tpcs2-login_x.zip
|-- rendu-tpcs2-login_x/
    |-- AUTHORS
    |-- tpcs2.sln
    |-- tpcs2
        |-- Tout sauf bin/ et obj/
```

Bien entendu, vous devez remplacer "`login_x`" par votre propre login.

N'oubliez pas de vérifier les points suivants avant de rendre :

- Le fichier `AUTHORS` doit être au format habituel (une *, une espace, votre login et un retour à la ligne).
- Pas de dossiers `bin` ou `obj` dans le projet.
- **Le code doit compiler !**

2 Introduction

2.1 Objectifs

Lors de ce TP nous allons aborder les notions suivantes :

- L'utilisation de la console en C#
- L'invite de commandes Windows
- Powershell
- La console sur Unix
- La fonction Main

3 Cours

3.1 La console

En bons utilisateurs de Windows vous aimez cliquer sur des boutons avec votre souris dans votre jolie interface graphique. Mais vous ne savez peut-être pas qu'il existe un autre monde, plus rapide, plus sombre, qui fera croire à toute personne extérieure que vous êtes un hacker confirmé : la console.
En effet, pendant la préhistoire [1], les informaticiens n'avaient à leur disposition qu'une interface en ligne de commande dans un écran affichant 80 colonnes de texte.
Malgré l'utilisation massive des interfaces graphiques (GUI = Graphical User Interface) de nos jours, la console et ses commandes textuelles résistent encore et toujours grâce à leur efficacité.

1. Au moins... 1970 !

3.2 L'invite de commandes Windows

Chaque version de Windows dispose d'une console héritée de l'époque de MS-DOS et accessible en lançant **cmd.exe** (avec Win+R par exemple) : l'invite de commandes.
Vous pouvez dans celle-ci exécuter tout un tas d'actions mais aussi les regrouper dans des scripts afin d'automatiser des tâches facilement. [2]
Voici les principales commandes :

dir
: Liste les fichiers du dossier courant.

cd
: Change le dossier courant.

copy
: Copie un fichier à un autre endroit.

mov
: Déplace un fichier.

print
: Affiche un fichier.

Il est également possible de lancer n'importe quel exécutable simplement en tapant son nom (précédé du chemin pour y accéder s'il n'est pas dans le dossier courant).
Cette interface est toutefois assez limitée et n'est aujourd'hui conservée que pour des raisons de compatibilité.

3.3 Powershell

Comme son nom l'indique (un shell est un interpréteur de commandes) il s'agit d'une nouvelle version de la console Windows introduite dans Windows 7.
Son fonctionnement se rapproche plus des équivalents Unix avec des noms de commandes proches et l'apparitions de fonctionnalités évolués telles que les redirections de flux. [3]
Les commandes expliquées précédemment deviennent celles-ci :

ls
: Liste les fichiers du dossier courant.

cd
: Change le dossier courant.

cp
: Copie un fichier à un autre endroit.

mv
: Déplace un fichier.

cat
: Affiche un fichier.

On note également l'apparition de **man** qui permet d'obtenir toutes les informations nécessaires à l'utilisation d'une commande en tapant man suivi de celle ci.

2. Et oui, un .bat ne sert pas qu'à détruire votre ordinateur.
3. http ://fr.wikibooks.org/wiki/Programmation_Bash/Flux_et_redirections

3.4 La console sur Unix

À partir de l'année prochaine (ou dès maintenant si vous voulez !) vous allez utiliser des systèmes d'exploitation Unix, dans ces derniers l'interface en ligne de commandes est encore très présente et est de ce fait très développée afin de pouvoir n'utiliser qu'elle sans être limité. C'est l'une des raisons qui fait que la majorité des serveurs tourne sous des OS de ce genre car lors d'un accès à distance seul un TTY [4] est accessible.
On ne détaillera pas ici toutes les merveilles que vous pouvez y trouver mais si vous êtes intéressés n'hésitez pas à vous renseigner sur internet où à demander à vos ACDC.

3.5 Utilisation en C#

La manipulation de la console en C# s'effectue au travers de la classe... Console !
Sa documentation est disponible dans la bible du C# : MSDN. Vous pouvez y accéder ici : http ://msdn.microsoft.com/en-us/library/system.console
Votre premier travail dans ce TP sera d'aller rechercher (pas forcément dans la page donnée au dessus) au moins les fonctions suivantes afin de comprendre leur utilisation :

1. Console.Write
2. Console.WriteLine
3. Console.Read
4. Console.ReadLine
5. Convert.ToInt32
6. Convert.ToFloat

3.6 La fonction Main

Lorsque vous compilerez et exécuterez votre programme, la première et seule fonction appelée sera la fonction `Main()`.
Cette fonction est le point d'entrée de votre programme, c'est donc à l'intérieur de celle-ci que vous pouvez faire appel à vos autres fonctions afin de les tester.

4. http ://en.wikipedia.org/wiki/Teleprinter

4 Exercices

4.1 Avant de commencer

Au cours de ces exercices vous devrez souvent afficher du texte en console, sans indication contraire tout affichage doit se terminer par un retour à la ligne.
Vous remarquerez aussi que si vous lancez votre programme avec Visual Studio, il va apparaitre et disparaitre bien trop rapidement pour que vous puissiez y voir quoi que ce soit. Pour remédier à ce problème vous pouvez soit ajouter un appel à `Console.Read()` à la fin de votre `Main` afin de rester ouvert en attendant une entrée de l'utilisateur ou simplement lancer votre programme depuis une invite de commande Windows.

Il est temps de rentrer dans le vif du sujet, créez un nouveau projet console dans Visual Studio en faisant File/New/Project/Console Application avec pour nom "tpcs02" et lancez-vous dans les exercices suivants !

4.2 Exercice 0 : Hello World

Vous devez écrire la fonction `HelloWorld` qui affiche dans la console le texte "Hello World !".
Prototype :

```
public static void HelloWorld()
```

Exemple :

```
> test_helloworld.exe
Hello World!
```

4.3 Exercice 1 : Ohé ! Écho... cho... o...

Écrivez la fonction `Echo` qui lit une ligne tapée par l'utilisateur et l'affiche à la suite.
Prototype :

```
static void Echo()
```

Exemple :

```
> test_echo.exe
test
test
```

4.4 Exercice 2 : Esrever

Écrivez la fonction `Reverse` qui lit une ligne sur la console et l'affiche à l'envers à la suite. Cette fonction devra être impérativement récursive. [5]
PROTIP : Soit une string s, `s[i]` renvoie le caractère d'index i (attention le premier est à 0) et `s.Length` renvoie la taille de s.
La fonction `Reverse` peut appeler une autre fonction chargée de la récursivité.

5. Et non récursivement impérative.

Prototype :

```
static void Reverse()
```

Exemple :

```
> test_reverse.exe
test
tset
```

4.5 Exercice 3 : Triforce

Écrivez la fonction `Triforce` qui affiche une triforce en jaune sur fond bleu et change le titre de la fenêtre en "Triforce".
Vous devez également appeler `Console.ResetColor()` à la fin pour ne pas conserver vos changements.
PROTIP : Ces modifications ne sont pas faites par des fonctions mais en modifiant des propriétés de `Console` (cherchez "Console Class" sur MSDN). Renseignez-vous également sur `ConsoleColor`.

Prototype :

```
static void Triforce()
```

Exemple :

```
> test_triforce.exe
   /\
  /__\
 /\  /\
/__\/__\
```

Bonus : n'utilisez qu'un seul `Write`, parce qu'en une ligne c'est mieux.

4.6 Exercice 4 : Dans ton QCM

Écrivez la fonction `QCM` qui pose une question à l'utilisateur et affiche :
- "Good job bro!" si la réponse est correcte.
- "You lose... The answer was X." si la réponse est fausse, en remplaçant X par le numéro de la bonne réponse.
- "So counting is too hard, n00b..." si l'entrée est invalide.

Les Paramètres `ansX` contiennent seulement le texte de l'option, c'est à vous d'afficher le "X) " au début de chacune.
Le paramètre `answer` contient le numéro de la bonne réponse.
Vous n'avez pas à gérer le cas où l'utilisateur n'entrerait pas un nombre.
PROTIP : Pour concaténer deux strings vous pouvez utiliser l'opérateur `+`.

Prototype :

```
static void QCM(string question,
                string ans1,
                string ans2,
                string ans3,
                string ans4,
                int answer)
```

Exemple :

```
> test_QCM.exe
Quelle est la différence entre un pigeon ?
1) Les deux pattes, surtout la gauche
2) Oui
3) Obiwan Kenobi
4) La réponse D
2
You lose... The answer was 1.
```

Bonus : gérer les entrées incorrectes (cherchez comment utiliser `try ... catch`).

4.7 Exercice 5 : ___ _____ .__. ... .

Écrivez la fonction `Morse` qui produit le son correspondant au message donné en entrée.
Ce message sera constitué des caractères '_', '.' et ' ' uniquement.
Un '_' correspond à un bip de 450ms, un '.' à un bip de 150ms et ' ' à une pause de 450ms.
Les sons doivent avoir une fréquence de 900Hz.
Cette fonction doit comme le reste du TP être récursive.
PROTIP : vous pouvez utiliser `Console.Beep(f, n)` pour produire un son de fréquence f Hz pendant n millisecondes et `System.Threading.Thread.Sleep(n)` pour mettre le programme en pause pendant n millisecondes.

Prototype :

```
static void Morse()
```

Exemple :

```
> test_morse.exe
·___ ·____· ·_ ·· __ · __ · ··· ·_ _·_· _·· _·_·
bip biiip biiip... (rien d'affiché)
```

4.8 Bonus 1 : Dessine-moi un poney

Écrivez la fonction `DrawPony` qui dessine le plus joli poney en ASCII art avec des couleurs et tout ce qui peut vous plaire.

Prototype :

```
static void DrawPony()
```

Bonusception : animez votre poney !

4.9 Bonus ultime : sl

Écrivez la fonction `Sl` qui reproduit le comportement de la commande **sl** sour Unix.
PROTIP : Google est votre ami.
Bonusception : gérez les arguments en ligne de commande.
(regardez du côté de `string[] args`, le paramètre de `Main`)

It's dangerous to code alone !