Bibliographie

1. C# et .NET Versions 1 à 4 par Gérard Leblanc, Eyrolles
2. Illustrated C# 2012 par Daniel Solis, Edition Apress
3. C# 7.0 in a nutshell, 0'Reilly

1 TP 1

1.1 Création d'un projet avec MonoDevelop

Un projet regroupe tous les fichiers nécessaires à la création d'une application.

— Lancez MonoDevelop. Cliquez sur "Nouvelle solution", choisissez le type "Application console" et validez avec le bouton "OK".

— Choisissez un répertoire où sera créé le projet (champ `Emplacement`) et donnez un nom au projet (champ `Nom`). Cochez éventuellement la case `Créer le répertoire pour la solution` (une solution regroupe plusieurs projets).

Pour le moment on s'intéressera surtout au fichier d'extension `cs` contenant le code C#.

1.2 Données : type et conversion

Exercice 1. Réaliser un programme qui écrit `Coucou`.

Exercice 2. Ecrire un programme qui :

— affiche : `Bonjour, appuyez sur une touche SVP`
— attend qu'on appuie sur une touche
— affiche : `Merci d'avoir appuyé sur une touche. Au revoir.`

Exercice 3. Réaliser un programme qui :

— demande à l'utilisateur de saisir un morceau de texte
— affiche le morceau de texte avec un message de remerciement
— attend que l'utilisateur appuie sur une touche avant de quitter

Exercice 4. Réaliser une petite calculatrice très simple qui :

— demande à l'utilisateur d'entrer un nombre a
— demande à l'utilisateur d'entrer un nombre b
— stocke ces valeurs dans des variables de type `int`
— effectue la somme de ces deux valeurs
— affiche l'addition complète avec son résultat
— attend que l'utilisateur appuie sur une touche avant de quitter

1.3 Les instructions de contrôle

Exercice 5. Réaliser un programme qui :
- demande à l'utilisateur d'entrer un nombre
- affiche si cette valeur est positive ou négative ou égale à 0

Exercice 6. Réaliser un programme utilisant un `switch` qui :
- demande à l'utilisateur de saisir un caractère
- affiche si celui-ci est une voyelle ou non.

Utiliser la méthode `ToLower` de la classe `string` : si `ch` est une variable de type `string` l'expression `ch.ToLower()` renvoie une copie de la chaîne `ch` en minuscules.

Exercice 7. Créer une application qui affiche un message différent en fonction du nom de l'utilisateur et du moment de la journée :
- `Bonjour XXX` pour la tranche horaire 18h-9h les lundi, mardi, mercredi, jeudi et vendredi
- `Bonsoir XXX` pour la tranche horaire 9h-18h les lundi, mardi, mercredi, jeudi
- `Bon week-end XXX` pour la tranche horaire vendredi 18h - lundi 9h

Pour récupérer l'heure courante utiliser l'instruction `DateTime.Now.Hour` qui renvoie un entier. Pour récupérer le jour de la semaine, utiliser l'instruction `DateTime.Now.DayOfWeek` qui renvoie une valeur de type `DayOfWeek` qui est l'énumération suivante :

```
enum DayOfWeek{Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}
```

Exercice 8. Ecrire un programme qui :
- demande à l'utilisateur de saisir un entier compris entre 1 et 4
- Si l'utilisateur a saisi 1, affiche `Vous avez entré 1` puis revient au début
- Si l'utilisateur a saisi 2, affiche `Vous avez entré 2` puis revient au début
- Si l'utilisateur a saisi 3, affiche `Vous avez entré 3` puis revient au début
- Si l'utilisateur a saisi 4, affiche `Le programme va se fermer` et attend que l'utilisateur appuie sur une touche avant de quitter

Exercice 9. En utilisant l'exercice précédent pour laisser le choix à l'utilisateur de la boucle utilisée, écrire un programme qui :
- demande à l'utilisateur d'entrer une valeur a et une valeur b (n'oubliez de tester que a est un nombre flottant et que b est un entier positif, sinon redemandez une valeur à l'utilisateur)
- Calcule la valeur de a^b en utilisant soit une boucle `do...while`, soit une boucle `while`, soit une boucle `for`.

Exercice 10. Programmer le jeu du plus ou moins dont les règles sont les suivantes : l'utilisateur doit deviner un nombre entier que l'ordinateur a tiré au hasard entre 0 et 99. A chaque coup l'utilisateur peut faire une proposition et l'ordinateur indique si le nombre saisi est plus grand ou plus petit que le nombre à trouver. Une fois trouvé, le nombre de coups utilisés est affiché.

Pour obtenir un nombre entier au hasard compris entre 0 et 99 utiliser l'instruction :

```
int nombreSecret= new Random().Next(0,100);
```

2 TP 2

2.1 Tableaux

Exercice 11. Ecrire une méthode qui met en oeuvre l'algorithme du tri à bulles décrit ci-dessous :
local : $i, j, n, temp \in$ Entiers naturels
Entrée-Sortie : $Tab \in$ tableau d'entiers naturels de n éléments (indices entre 1 et n)
début
pour i de n jusqu'à 1 faire
pour j de 2 jusqu'à i faire
si $Tab[j-1] > Tab[j]$ alors
$temp \leftarrow Tab[j-1]$
$Tab[j-1] \leftarrow Tab[j]$
$Tab[j] \leftarrow temp$
Fsi
Fpour
Fpour

Tester la méthode en triant un tableau construit à partir de nombres entiers aléatoires. Pour obtenir plusieurs entiers aléatoires, fabriquer une variable de type `Random` et invoquer la méthode `Next`. Par exemple, le code suivant affiche 7 entiers aléatoires entre 0 et 9 :

```
Random gen=new Random();
for (int i=0; i<7; i++) Console.WriteLine(gen.Next(0,10));
```

Exercice 12. Créer un programme qui détermine les N premiers nombres premiers où N est un entier plus grand que 1 donné par l'utilisateur. On pourra utiliser que si $(p_k)_{k\geq 1}$ est la suite strictement croissante de tous les nombres premiers on a $p_1 = 2$ et que pour tout $k \geq 2$: p_k est le plus petit entier strictement plus grand que p_{k-1} qui n'est pas divisible par p_ℓ pour tout $\ell \in \{1, \ldots, k-1\}$.

Exercice 13. Ecrire une méthode qui à partir d'un tableau $Tab1$ (de `double`) et d'un tableau d'entier $Tab2$ crée un tableau Tab tel que $Tab[i] = Tab1[Tab2[i]]$. Ecrire un programme pour la tester.

2.2 Permutations

Soit N un entier ≥ 1. Une permutation de l'ensemble $E_N = \{0, 1, \ldots, N-1\}$ est une bijection σ de E_N dans E_N qui peut-être représentée dans un programme par un tableau dont les éléments sont $\sigma(0), \sigma(1), \ldots, \sigma(N-1)$.

Exercice 14. Ecrire une méthode qui fabrique une permutation aléatoire de l'ensemble $\{0, 1, \ldots, N-1\}$ où N est un entier ≥ 1 donné en argument.

L'ordre d'un permutation σ est le plus petit entier $k \geq 1$ tel que σ^k est égal à l'identité (ici $\sigma^k = \underbrace{\sigma \circ \sigma \circ \cdots \circ \sigma}_{k \text{ fois}}$).

Exercice 15. Ecrire une méthode qui détermine l'ordre d'un permutation σ donnée en argument.

Pour $i \in \{0, \ldots, N-1\}$, l'orbite de i sous l'action de σ est l'ensemble $\Omega_{i,\sigma} = \left\{\sigma^k(i); k \geq 1\right\}$. Le nombre d'orbites de σ est le nombre d'orbites différents parmi $\Omega_{0,\sigma}, \Omega_{1,\sigma}, \ldots, \Omega_{N-1,\sigma}$.

Exercice 16. Ecrire une méthode qui détermine le nombre d'orbites d'un permutation σ donnée en argument.

Exercice 17. Vérifier numériquement (par la méthode de Monte-Carlo) que l'espérance du nombre d'orbites d'une permutation aléatoire est égal à $\sum_{i=1}^{N} \frac{1}{i}$.

2.3 Chaînes de caractères - Conversion - Tableaux déchiquetés

Exercice 18. Considérons un nombre entier positif, par exemple 377. Multiplions ses chiffres : $3 \times 7 \times 7 = 147$. Opérons de même avec le résultat 147 : $1 \times 4 \times 7 = 28$. Recommençons : $2 \times 8 = 16$. Encore : $1 \times 6 = 6$. Arrivé à un nombre d'un seul chiffre, on ne peut plus rien faire : $377 \to 147 \to 28 \to 16 \to 6$.

Cette suite est la « suite multiplicative » de 377 et la « persistance multiplicative » p de 377 est le nombre de fois qu'il a fallu multiplier les chiffres avant d'arriver à un nombre à un seul chiffre ; ici, $p = 4$.

1. Ecrire un programme qui détermine la suite multiplicative d'un nombre entier positif donné par l'utilisateur.
2. Ecrire un programme qui construit un tableau contenant les suites multiplicatives des nombres entiers compris entre deux constantes N_0 et N_1 (par exemple $N_0 = 277777788888890$ et $N_1 = 277777788888905$). On affichera le contenu du tableau sur la console.

Astuce : Une chaîne de caractères `S` en C# peut être considéré comme un tableau : `S[i]` est le $(i+1)$ème caractère de `S` (de type `char`) et `S.Length` est le nombre de caractères de `S`.