TPC#1 Variables, Calcul et branchement

1 Préliminaires

J'étais alors en proie à la mathématique.
Temps sombre ! enfant ému du frisson poétique,
Pauvre oiseau qui heurtais du crâne mes barreaux,
On me livrait tout vif aux chiffres, noirs bourreaux ;
On me faisait de force ingurgiter l'algèbre :
On me liait au fond d'un Boisbertrand funèbre ;
On me tordait, depuis les ailes jusqu'au bec,
Sur l'affreux chevalet des X et des Y ;
Hélas ! on me fourrait sous les os maxillaires
Le théorème orné de tous ses corollaires ;
Et je me débattais, lugubre patient
Du diviseur prêtant main-forte au quotient.
De là mes cris.

Victor Hugo, `Les contemplations`

1.1 Objectifs

Le but de ce TP est de vous familiariser avec le `C#` tout en introduisant le concept de variables, le mécanisme d'assignation, et les branchements conditionnels. Pour cela vous allez donc réaliser une calculatrice scientifique en utilisant vos connaissances du TP précédent et les nouveaux principes abordés.

1.2 Consignes

N'oubliez pas de lire le sujet en entier avant de commencer le tp. Des pénalités pourront être appliquées en cas de non respect des règles suivantes :

- Un fichier AUTHORS doit être présent à la racine du rendu. Il doit contenir une étoile *, un espace, puis votre login (ex : login_x) suivi d'un retour à la ligne.
- La solution doit compiler.
- Votre code doit être commenté –au moins un commentaire par fonction.
- Le code doit être propre, correctement indenté. N'oubliez pas que `Visual Studio` dispose d'outils de mise en forme automatique, ce qui justifie les lourdes pénalités appliquées en cas d'abus.
- Vous devez respecter l'arborescence de rendu donné par vos `ACDC` respectifs.
- Le nom de votre Solution est Calculatrice.
- La triche ou l'échange de code sont sévèrement punis.

2 Cours et exemples

Avant de vous lancer dans les exercices, un peu de lecture !

2.1 Définition d'une variable

"En informatique, les variables associent un nom (le symbole) à une valeur ou un objet, elles font partie des identificateurs de l'algorithmique, et doivent donc par conséquent avoir des noms différents des mots-réservés propre à chaque langage de programmation. Il est notable que beaucoup de ces langages (les langages impératifs) autorisent les variables à changer de valeur au cours du temps."

Wikipédia, `Variables`

2.2 Présentation et utilisation des variables

En C# vous disposez de différents types de variables pour contenir différents types de données , voici une présentation des principaux types que vous serez amenés à utiliser :

Principaux types du `C#`	Description
byte	Entier de 0 à 255
short	Entier de -32768 à 32767
int	Entier de -2147483648 à 2147483647
long	Entier de -9223372036854775808 à 9223372036854775807
float	Nombre simple précision de -3,402823e38 à 3,402823e38
double	Nombre double précision de -1,79769313486232e308 à 1,79769313486232e308
char	Représente un caractère
string	Une chaine de caractère
bool	Le type booleen avec 2 valeurs True ou False

Dans notre convention les noms de variables doivent commencer par une lettre minuscule et peuvent contenir ensuite des lettres minuscules et majuscules, des chiffres et le caractère '_'.

La déclaration et l'affectation de variables se font ainsi :

```
int x = 1;  // x holds the value 1, This is a comment
x = 2;      // now x holds the value 2
```

2.3 Alternatives 2 : le retour

Bien entendu les variables ne sont pas les seuls outils à votre disposition, les alternatives sont également présentes. Leur utilisation vous sera relativement familière, vous disposez des deux fameux mots clés **if** et **else**. Leur utilisation est la suivante :

```
bool flagCheck = true;

if (flagCheck == true)
{
  Console.WriteLine}(‘‘The flag is set to true.’’);
}
else
{
  Console.WriteLine}(‘‘The flag is set to false.’’);
}
```

Rien de bien compliqué, le mot clé **else if** permet d'enchaîner les alternatives :

```
int eger = 3;

if (eger == 3)
{
  Console.WriteLine(‘‘Eger is equal to 3’’);
}
else if (eger == 5)
{
  Console.WriteLine}(‘‘Eger is equal to 5’’);
}
else
{
  Console.WriteLine(‘‘Eger is not interesting’’);
}
```

Ce code assigne donc la valeur 3 à un entier Eger, et teste si cet entier vaut 3, sinon il teste si il vaut 5, et si aucun des deux cas n'est satisfait le code dans le else est lancé.

2.4 Les tests

Il existe différents tests pour les alternatives, utilisables aussi bien dans les if que dans les else if. Bien entendu le else n'a pas besoin de test.

Test	Description
a == b	Si a est égal à b
a <= b	Si a est inférieur ou égal à b
a >= b	Si a est supérieur ou égal à b
a < b	Si a est inférieur à b
a > b	Si a est supérieur à b
a != b	Si a est différent de b

Hint

| Gardez bien à l'esprit la différence entre l'affectation '=' et le test d'égalité '==' !

2.5 Récursion

La récursion est également très utile en C#, elle est possible avec n'importe quelle fonction sans qu'aucun mot clé ne soit nécessaire.

Voici un court exemple :

```
int rec(int b)
{
  if (b > 10)
  {
    b = 0;
  }
  if (b == 10)
  return (b);
  return (rec(b + 1));
}
```

Comme notre fonction retourne un entier, nous avons besoin du mot clé **return** qui indique la valeur que notre fonction renvoie a chaque fois qu'elle termine.

3 Un peu de mise en jambes

Maintenant entrons dans le vif du sujet avec vos premières fonctions. Pour l'ensemble des TP C# nous définirons les fonctions à réaliser par ce qui s'appelle un prototype, la ligne qui caractérise votre fonction. Chaque prototype comprend donc le type de retour de la fonction, son nom, et le nom des ses paramètres précédés de leur type, le tout entre parenthèses. Pour votre plus grand bonheur nous allons faire des maths, l'utilisation de math.h est prohibée.

3.1 Exercice 1 Approx

Ecrivez une fonction qui approxime un flottant en lui laissant b décimales après la virgule. Si le nombre a moins de décimales que b, rien n'est fait. Si b vaut 0, approx retourne la partie entière.

Son prototype est le suivant :

```
float approx(float a, int b)
```

3.2 Exercice 2 Sqrt

Un grand classique, une fonction racine carrée par la méthode de Héron. Vous allez utiliser votre amie la récursion en appliquant la formule suivante :

$$x_{n+1} = x_n - \frac{x_n^2 - A}{2x_n} = \frac{x_n^2 + A}{2x_n} = \frac{x_n + \frac{A}{x_n}}{2}$$

Référez vous a Wikipedia pour de plus amples informations sur la méthode.

Le prototype attendu est :

```
float my_sqrt(float a)
```

3.3 Exercice 3 Pow

Un autre incontournable, la fonction puissance en récursif. A vous de trouver votre algorithme.

```
float my_pow(float a, int b)
```

3.4 Exercice 4 SexyPrime

Les nombres premiers sexys (sic !) sont des nombres premiers dont la différence vaut 6. Ecrivez une fonction qui détermine si a et b sont des nombres premiers sexys. Si a ou b sont plus grand que 1000 renvoyez false.

```
bool is_sexy(int a, int b)
```

3.5 Exercice 5 Factorielle

Cette fonction vous servira toute votre vie, voire même tout EPITA. Ecrivez donc la classique fonction factorielle en récursif qui renvoie la factorielle de a.

```
int facto(int a)
```

3.6 Exercice 6 Degrés et Radians

Une fonction d'une simplicité déconcertante, convertir des degrés en radians et sa réciproque. Les prototypes sont les suivants :

```
float deg_to_rad(float a)

float rad_to_deg(float a)
```

3.7 Exercice 7 Trigo

Un ensemble de fonction trigonométriques, référez vous aux séries de Taylor, la méthode à appliquer est toujours récursive. Voici par exemple pour le cosinus :

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}$$

Contentez vous de six récursions.

Voici les prototypes des fonctions à réaliser, choisissez entre degrés et radians, sachant qu'une solution sera plus simple que l'autre.

```
float my_cos(float a)

float my_sin(float a)

float my_tan(float a)
```

4 Here comes the funny part

Il est maintenant temps de mettre vos fonctions en application, le but est d'obtenir une calculatrice scientifique fonctionnelle. Utilisez vos connaissances acquises en Windows Form pour nous éblouir avec des boutons et des fonctionnalités inédites. Globalement néanmoins vous aurez besoin d'une RichTextBox pour l'affichage des nombres, du résultat et des boutons correspondant à vos fonctions. N'hésitez pas cependant à révolutionner la calculatrice !

Pour stocker votre résultat dans tout votre programme nous vous autorisons l'usage d'une variable globale, qui définie hors de toute fonction sera accessible dans l'ensemble de votre programme. Gardez à l'esprit que ceci demeure néanmoins **strictement interdit** et n'est autorisé ici qu'en attente d'outils bien plus élégants pour faire la même chose.

4.1 Exercice 1

Pour cette partie vous devez implémenter 5 fonctions qui prennent en paramètre un flottant et effectuent l'opération correspondante sur la variable globale contenant le résultat. Voici les prototypes :

```
void add(float a)

void sub(float a)

void mul(float a)

void div(float a)

void mod(float a)
```

4.2 Exercice 2

Votre objectif ici est de lier vos Windows Form et vos opérations afin d'obtenir une calculatrice fonctionnelle sur les 5 opérations de l'exercice 1.

4.3 Exercice 3

Maintenant que vous avez développé la partie basique de la calculatrice, attaquons nous a la partie scientifique. La consigne est tout aussi simple, cette fois réutilisez vos fonctions de la partie 1 dans votre calculatrice scientifique. N'oubliez pas de permettre à l'utilisateur de choisir entre degrés et radians.

5 Bonus

Vous croyez en avoir fini ? Voici des idées de fonctions à ajouter :

```
void exp(float a)

void ln(float a)

void log(float a)

void my_super_pow(float a, float b)
```

Mais de nombreux bonus sont aussi possibles dans votre interface de Calculatrice, voici quelques idées :
- Ajouter une barre de menu
- Permettre de changer de modes de calcul
- Gérer différentes bases
- Implémenter un mode programmeur

Vous êtes libres donc étonnez nous !

6 Conclusion

N'oubliez pas de bien commenter votre code, Bonne chance ;)