

# Compte rendu d'activité

TP : Revue de Code

10/2020

## Contexte

Un client réalise des applications d'apprentissage ainsi que de tests de connaissances dans divers environnements. Actuellement, une application est réalisée dans un environnement non graphique, afin de l'intégrer dans un projet global de tests de connaissances en mathématiques, à différents niveaux de difficultés.

## Application console C#

### Code existant

Ce fichier **Program.cs** rassemble l'ensemble de l'algorithme de l'application. Son code source initial est accessible [ici](#).

### Les fonctionnalités existantes

Actuellement, à l'ouverture de l'application, un menu s'affiche permettant au choix :

- D'accéder à la fonctionnalité de permutation
  - tous les mélanges possibles dans un ensemble d'objets
- D'accéder à la fonctionnalité d'arrangement
  - tirage d'une quantité d'objets dans l'ordre
- D'accéder à la fonctionnalité de combinaison
  - tirage d'une quantité d'objets sans ordre

```
Permutation ..... 1
Arrangement ..... 2
Combinaison ..... 3
Quitter ..... 0
Choix : 1
nombre total d'éléments à gérer = 4
4! = 24
Permutation ..... 1
Arrangement ..... 2
Combinaison ..... 3
Quitter ..... 0
Choix : 2
nombre total d'éléments à gérer = 4
nombre d'éléments dans le sous ensemble =
```

Figure 1: Affichage de l'application et de son menu

## Expressions des besoins

Le chef de projet vous demande de réaliser la revue de code de cette application puis de remplir le rapport d'analyse de la revue de code qui contient la liste des points à contrôler. Un fichier contenant un ensemble de règles de codage est fournis et un second sera à remplir avoir les erreurs de codages détectés et leurs modifications apportées. Enfin, l'envoi du code corrigé sera envoyé sur Github par l'intermédiaire d'une branche dédiée à cette correction dans l'attente de sa validation par pull request.

## Informations techniques et outils utilisés

### Mise en place de l'environnement de développement

#### IDE Visual Studio

Visual Studio est un IDE permettant le développement d'application C#. L'IDE est téléchargeable à [ce lien](#), cependant la version pourrait changer selon les évolutions de l'IDE. Après l'installation de cet IDE avec les options par défaut, il convient de le démarrer en mode administrateur afin de simplifier les différents éléments de démarrage. De plus, ici le logiciel a été configuré pour s'ouvrir automatiquement dans ce mode. Pour cela, il convient d'effectuer un clic droit sur l'icône de lancement de Visual Studio, d'aller dans « Propriétés », puis à l'ouverture de la fenêtre des propriétés, de cliquer sur « Avancé » puis de cocher « exécuter en tant qu'administrateur » comme le montre la [figure](#).

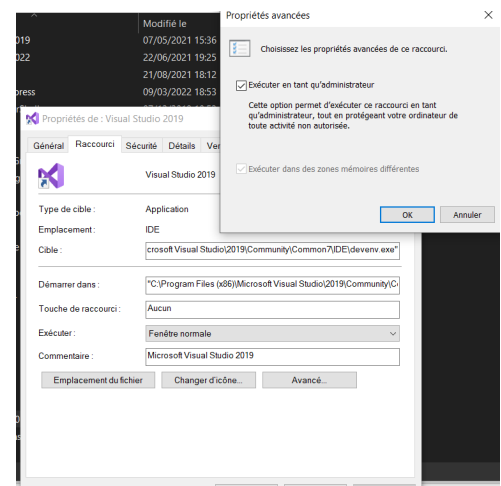


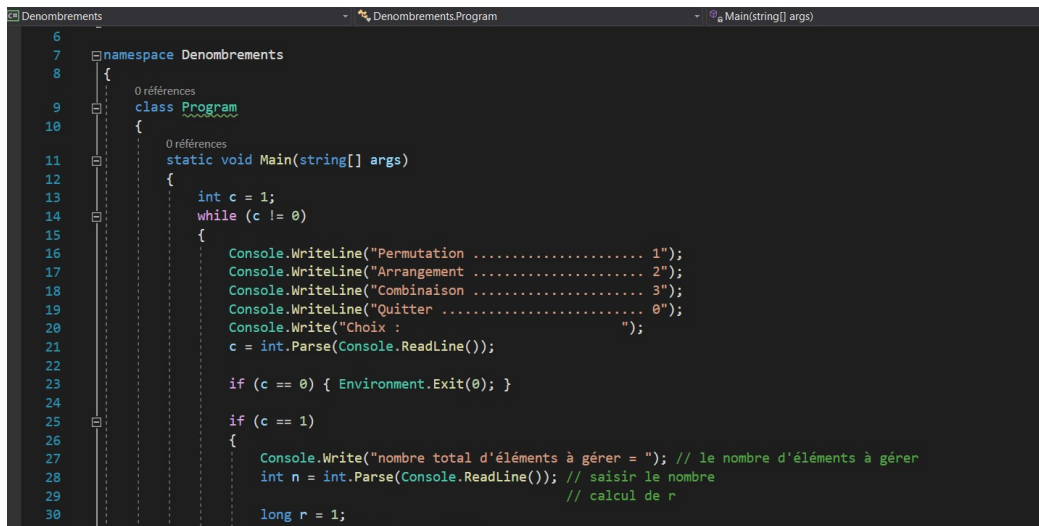
Figure 2: Fenêtre des propriétés de l'icône de lancement de l'IDE

#### Récupération du code source

La récupération du code source se fait sur la plateforme CNED sous forme de dossier zippé. Après téléchargement, avoir dézipper le dossier et avoir placé le projet dans le dossier de traitement, il est possible d'ouvrir directement le projet dans l'IDE en cliquant sur le fichier finissant par **.sln**, ici **Denombrement.sln**. Enfin, il suffit de cliquer sur l'icône de lecture verte à côté de démarrer afin de lancer l'application.

## Lier le dépôt Github avec Visual Studio

Afin de pouvoir directement gérer le dépôt GitHub depuis Visual Studio, il convient d'aller dans l'onglet « **Modification Git** » puis de saisir un message de commit dans la fenêtre qui apparaît. A cet instant, les modifications non intégrés au dépôt s'affiche dans l'onglet **Modification**. Dans le cas présent, le dépôt étant préalablement créé et le compte de dépôt étant déjà lié à Visual Studio, il suffit de commiter puis de cliquer sur « **valider tout et pousser** ». A la réussite de l'envoi, une pop-up s'affiche.



```
6
7 namespace Denombrements
8 {
9     class Program
10    {
11        static void Main(string[] args)
12        {
13            int c = 1;
14            while (c != 0)
15            {
16                Console.WriteLine("Permutation ..... 1");
17                Console.WriteLine("Arrangement ..... 2");
18                Console.WriteLine("Combinaison ..... 3");
19                Console.WriteLine("Quitter ..... 0");
20                Console.Write("Choix : ");
21                c = int.Parse(Console.ReadLine());
22
23                if (c == 0) { Environment.Exit(0); }
24
25                if (c == 1)
26                {
27                    Console.Write("nombre total d'éléments à gérer = "); // le nombre d'éléments à gérer
28                    int n = int.Parse(Console.ReadLine()); // saisir le nombre
29                    // calcul de r
30                    long r = 1;
```

Figure 3: Affichage du fichier Program.cs dans Visual Studio

L'évolution de l'application récupérée a été effectué avec la version 16.11.10 de l'IDE Microsoft Visual Studio 2019.

## Spécificités documentations fournies

La revue du code produite a été effectuée par l'intermédiaire du fichier de rapport de revue de code fourni par le chef de projet qui est récupérable [ici](#) à l'aide du fichier présentant un ensemble de [règles de codage](#) sous Visual Studio

# Réalisation

## Revue de code

### Erreur et dysfonctionnement

Il a été noté qu'il existe un problème d'exécution dû à une mauvaise configuration de condition if, comme le montre la figure suivante seul les choix 0, 1, 2 et 3 sont pris en charge.

```
if (c == 0) { Environment.Exit(0); }

if (c == 1)
{
    Console.Write("nombre total d'éléments à gérer = "); // le nombre d'éléments
    int n = int.Parse(Console.ReadLine()); // saisir le nombre
                                           // calcul de r

    long r = 1;
    for (int k = 1; k <= n; k++)
        r *= k;
    Console.WriteLine(n + "! = " + r);
}
else
{
    if (c == 2)
    {
```

Ainsi, afin de résoudre ce problème le if a été modifié de la manière suivante :

```
// choix correct excluant le choix de quitter
if (choix == "1" || choix == "2" || choix == "3")
{
    try
```

### Gestion des commentaires

Il a été remarqué que les commentaires normalisés nécessaires à la génération de la documentation technique n'étaient pas mis en place. Ainsi, ceux-ci ont été rajouté devant chaque méthode, **main** et **ProduitEntier**. Aussi, plusieurs commentaires informatifs non pertinents ont été supprimés du code afin de le rendre plus lisible, dont certains étaient des appels consoles n'ayant pas d'intérêts pour l'application.

```
int n = int.Parse(Console.ReadLine()); // saisir le nombre
// calcul de r1
long r1 = 1;
for (int k = (t - n + 1); k <= t; k++)
    r1 *= k;
// calcul de r2
long r2 = 1;
for (int k = 1; k <= n; k++)
    r2 *= k;
// calcul de r3
//Console.WriteLine("résultat = " + (r1 / r2));
Console.WriteLine("C(" + t + "/" + n + ") = " + (r1 / r2));
```

Figure 4: Exemple de code mort et de commentaires non pertinents supprimés.

### Éviter les répétitions

Selon le chef de projet, une fonction doit être mise en place afin de simplifier le programme qui possède de nombreuses répétitions. Elle doit permettre de multiplier une suite d'entiers, d'une valeur à une autre (par exemple la multiplication de 4 à 6, donc  $4*5*6$ ).

```
/// <summary>
/// Calcul du produit des entiers successifs, depuis valeurDepart jusqu'à valeurArrivee
/// </summary>
/// <param name="valeurDepart">valeur de départ</param>
/// <param name="valeurArrivee">valeur d'arrivée</param>
/// <returns>résultat du produit entre les valeurs de départ et d'arrivée
/// ou 0 si dépassement de capacité</returns>
3 références
static long ProduitEntiers(int valeurDepart, int valeurArrivee)
{
    long produit = 1;
    for (int k = valeurDepart; k <= valeurArrivee; k++)
    {
        produit *= k;
    }
    return produit;
}
```

Figure 5: Méthode ProduitsEntiers demandés par le chef de projet

### Présentation globale

Aussi sur certaines boucles les accolades sont manquantes. Bien que cela ne nuit pas à l'exécution du programme, cela le rend moins lisible, ainsi lorsque cela a été nécessaire, les accolades ont été rajoutés et correctement indentés.

```
long r = 1;
for (int k = 1; k <= n; k++)
    r *= k;
Console.WriteLine(n + "! = " + r);
```

Enfin, comme le montre la figure ci-dessus, les noms de variables utilisées ne permettent pas de comprendre le code du programme, ainsi il a été nécessaire de les renommer en conséquence afin de rendre le code compréhensible. Si nous avons ce code suivant :

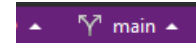
```
Console.Write("nombre total d'éléments à gérer = "); // le nombre d'éléments à gérer
int t = int.Parse(Console.ReadLine()); // saisir le nombre
Console.Write("nombre d'éléments dans le sous ensemble = "); // le sous ensemble
int n = int.Parse(Console.ReadLine()); // saisir le nombre
// calcul de r1
long r1 = 1;
for (int k = (t - n + 1); k <= t; k++)
    r1 *= k;
// calcul de r2
long r2 = 1;
for (int k = 1; k <= n; k++)
    r2 *= k;
// calcul de r3
// Console.WriteLine("résultat = " + (r1 / r2));
Console.WriteLine("C(" + t + "/" + n + ") = " + (r1 / r2));
```

Le code a été remplacé par celui de la figure qui suit :

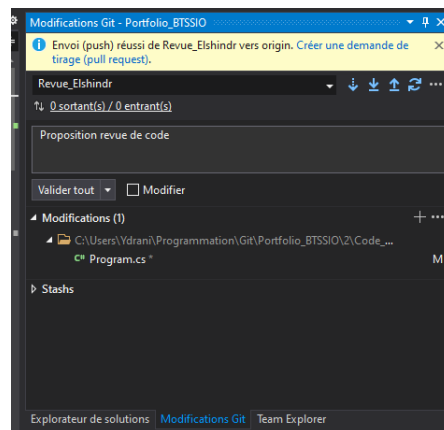
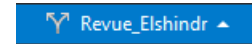
```
Console.Write("nombre d'éléments dans le sous ensemble = ");
int nbSousEnsemble = int.Parse(Console.ReadLine());
// calcul de l'arrangement qui sert aussi au calcul de la combinaison
long arrangement = ProduitEntiers(nbTotal - nbSousEnsemble + 1, nbTotal);
// choix : arrangement
if (choix == "2")
{
    Console.WriteLine("A(" + nbTotal + "/" + nbSousEnsemble + ") = " + arrangement);
}
// choix : combinaison
else
{
    long combinaison = arrangement / ProduitEntiers(1, nbSousEnsemble);
    Console.WriteLine("C(" + nbTotal + "/" + nbSousEnsemble + ") = " + combinaison);
}
```

## Envoi sur github depuis une branche annexe

Avant de commiter, il convient de se rendre en bas à droite de l'écran de l'IDE afin de cliquer sur le nom de branche : main.

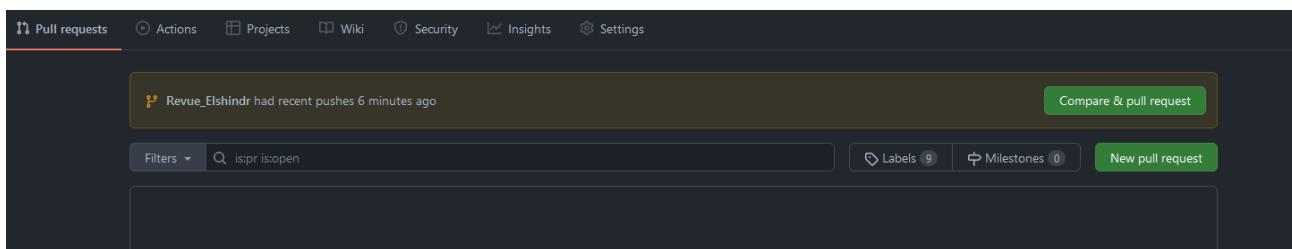


En cliquant dessus une fenêtre s'ouvre proposant de récupérer une branche ou bien d'en créer une nouvelle. Dans notre cas, la branche **Revue\_Elshindr** a été créée.



Afin de commiter les propositions de modifications, il faut se rendre dans l'onglet **Modification Git** puis entrer le message de **commit** puis cliquer sur **valider et de push**. On obtient donc la figure suivante :

En se rendant dans l'onglet **Pull Requests** sur le repository Github, il est possible de voir le message suivant :




En cliquant sur **Compare and pull request**, une seconde fenêtre s'ouvre montrant la figure suivante qui permet d'ouvrir un pull request ainsi que de voir les différences entre les branches et les différents fichiers modifiés puis de valider ou non les modifications proposées par la branche.



## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: main ← compare: Revue\_Elshindr ✓ **Able to merge.** These branches can be automatically merged.

 Proposition revue de code

Write

Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Reviewers

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Development


Use [Closing keywords](#) in the description to

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

## Proposition revue de code #1

Open Elshindr wants to merge 1 commit into `main` from `Revue_Elshindr`

Conversation 0 Commits 1 Checks 0 Files changed 41

 Elshindr commented now

No description provided.

Proposition revue de code 34bee54

Add more commits by pushing to the `Revue_Elshindr` branch on `Elshindr/Portfolio_BTSSIO`.

This branch has not been deployed


No deployments

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also open this in [GitHub Desktop](#) or view [command line instructions](#).

 Write

Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Close pull request

Comment

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

# Bilan

Après modification du code dans **Program.cs**, l'affichage et l'algorithme mis initialement en place ne sont pas affectés par les modifications. Ainsi, le code final du programme est plus lisible, clair et respecte les usages d'indentations de code et de commentaires normalisés.

## Liste des compétences couvertes

### B1-2 Répondre aux incidents et aux demandes d'assistance et d'évolution

- Traiter des demandes concernant les applications