Découverte de l'IDE et premier exemple

Découverte du code et de l'environnement de développement :

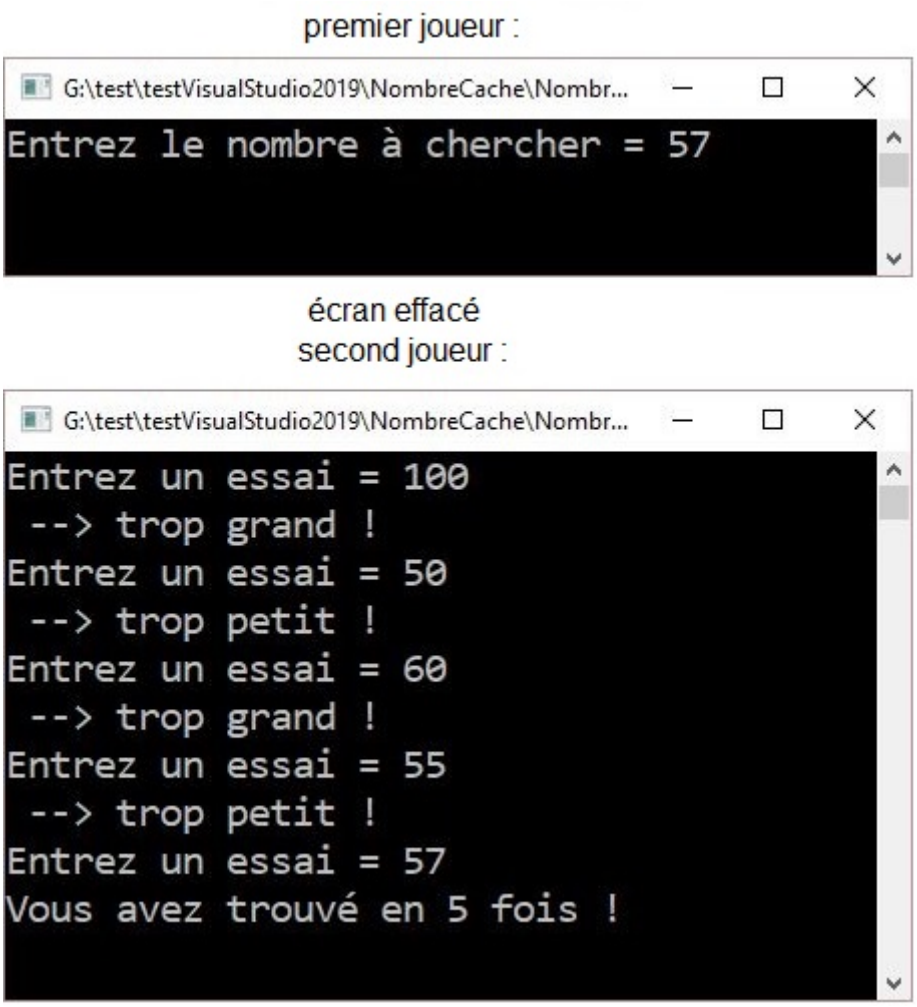
Vous allez coder une petite application qui va vous permettre de découvrir un peu mieux l'environnement de développement et les instructions de base de la programmation en langage C# (C Sharp).

But du premier exemple :

Le jeu du nombre caché se joue à deux joueurs.

- Le premier joueur saisit un nombre puis sa saisie est effacée.
- Le second joueur va ensuite tenter de trouver le nombre en saisissant des essais.
- A chaque essai le message **trop grand** ou **trop petit** est affiché.
- Dès que la valeur est trouvée, le nombre d'essais est affiché.

Les captures ci-contre montrent un exemple de fonctionnement.



Etape 1

Saisie de la valeur à chercher

La première étape de votre programme consiste à demander au premier joueur de saisir une valeur et de mémoriser cette valeur.

Vous savez déjà afficher un message à l'écran et attendre une saisie. Commencez à écrire dans le 'main' :

```
Console.Write("Entrez le nombre à chercher = ");
```

```
Console.ReadLine();
```

test

Faites un test d'exécution. Remarquez que l'affichage se fait mais le curseur se positionne non pas à la suite mais sous la ligne affichée.

Pour changer cela, il suffit d'utiliser **Write** à la place de **WriteLine**.

Faites la modification.

test Cette fois le curseur se positionne bien à la suite du message.

Pour le moment, la saisie est possible mais n'est pas mémorisée. Il faut enregistrer la saisie dans une **variable**. Vous êtes libre du nom donné à la variable en respectant certaines règles (cela sera préciser plus tard).

```
Console.Write("Entrez le nombre à chercher = ");
valeur = Console.ReadLine();
```

Remarquez que **valeur** est souligné en rouge pour indiquer une erreur.

Placez le curseur sur **valeur** sans cliquer : vous allez voir le même message qui apparaît en bas **Le nom 'valeur' n'existe pas dans le contexte actuel**.

Avant d'être utilisée, une variable doit être déclarée pour que l'ordinateur réserve une place mémoire pour elle. La déclaration consiste à préciser son nom précédé de son type (une variable peut être de type entier, réel, chaîne, ...).

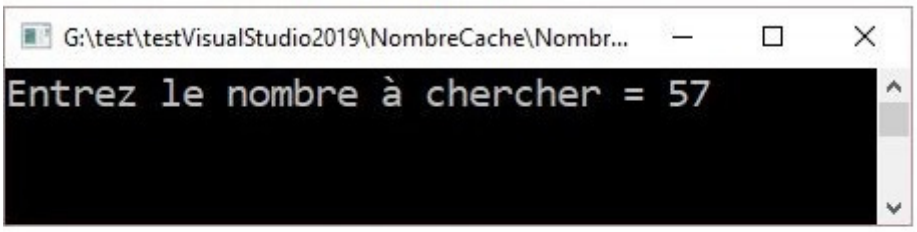
```
// déclaration
int valeur;

// saisie du nombre à chercher
Console.Write("Entrez le nombre à chercher = ");
valeur = Console.ReadLine();
```

Remarquez l'ajout de **commentaires** : ce sont les lignes qui commencent par ***%%%%. A la suite, vous pouvez écrire ce que vous voulez. Cela permet d'expliquer le code. Il est alors plus facile à relire et comprendre. L'ordinateur ne tient pas compte des commentaires. Suite à la déclaration de valeur, cette fois une autre erreur est apparue : Impossible de convertir implicitement le type 'string' en 'int'. En effet, une saisie à la console retourne toujours un type string (une chaîne de caractères) quelle que soit l'information saisie. valeur doit être de type int (entier) car on va faire des comparaisons de nombres. Il est possible de changer le type de la saisie de la façon suivante : <code c#> valeur = int.Parse(Console.ReadLine()); </code> Cela signifie qu'on demande de parser (changer le type) de l'information entre parenthèses (donc ici, de ce qui est saisi) en type int. Cette fois il n'y a plus d'erreur. Après la récupération de la valeur à chercher, il faut effacer l'écran avant que le second joueur ne commence. Cela se fait avec l'instruction suivante : <code c#> Console.Clear(); </code> Enfin, comme on l'a vu précédemment, prenez l'habitude de laisser en fin de programme la ligne suivante, qui permet d'éviter la fermeture directe de la fenêtre de commande, en fin d'exécution du programme : <code c#> Console.ReadLine(); </code> Au final, vous devriez obtenir ceci : <code c#> déclaration int valeur; saisie du nombre à chercher Console.Write("Entrez le nombre à chercher = "); valeur = int.Parse(Console.ReadLine()); Console.Clear(); Console.ReadLine(); </code>***

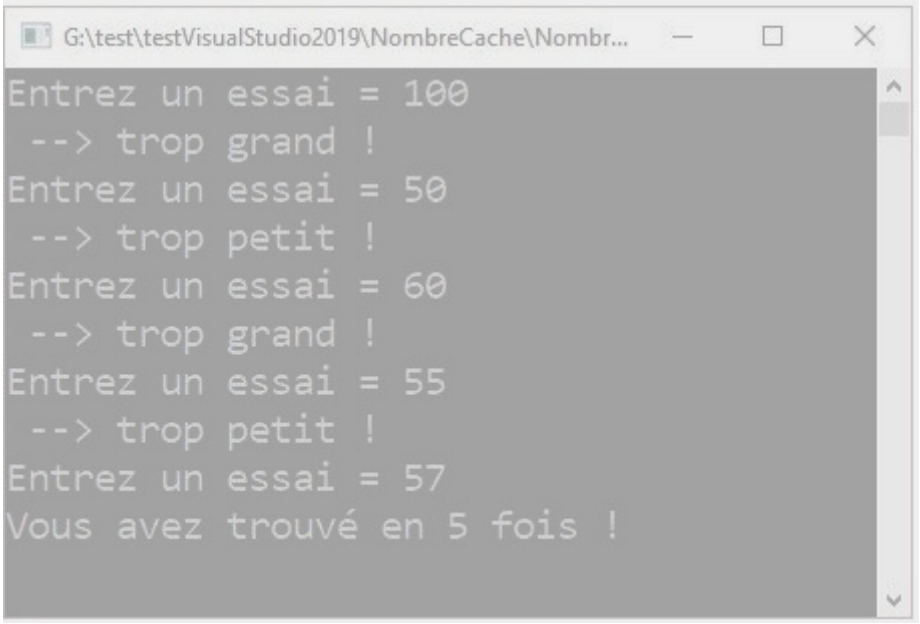
test Vérifiez si vous arrivez à saisir puis si l'écran s'efface.

premier joueur :



écran effacé

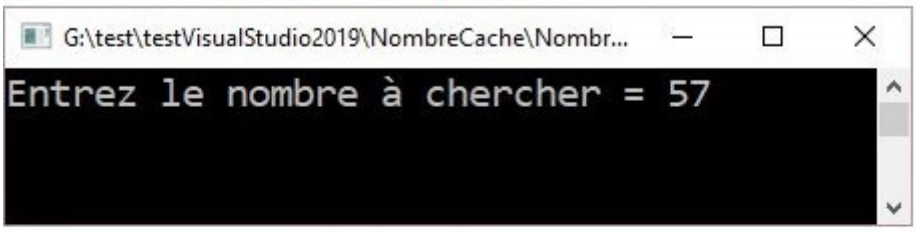
second joueur :



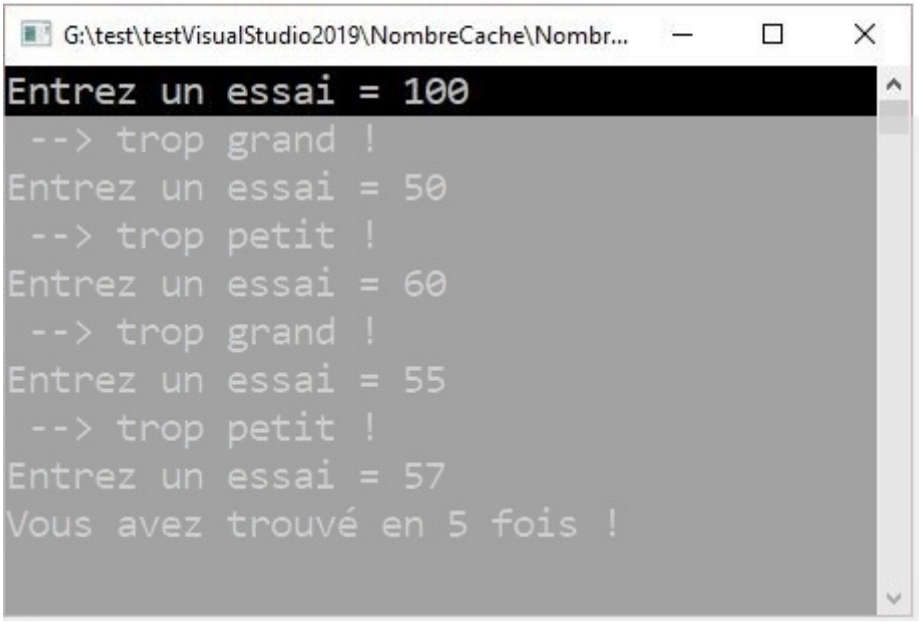
===== Etape 2 ===== ==== Saisie d'un premier essai ==== Maintenant que l'écran est effacé, le second joueur va pouvoir commencer. Le principe va être le même que précédemment : * il faut afficher un message * et mémoriser la saisie dans une nouvelle variable. Il faut utiliser une autre variable pour ne pas perdre la valeur d'origine qui est à trouver. A la suite du code actuel (mais toujours avant le Console.ReadLine final), ajoutez : <code c#> *saisie du premier essai Console.Write("Entrez un essai = "); essai = int.Parse(Console.ReadLine()); </code> Remarquez que **essai** est soulignée en rouge. C'est normal et vous devriez savoir pourquoi.: la variable n'est pas déclarée. Ajoutez sa déclaration, au même niveau que 'valeur' :* <code c#> déclaration int valeur, essai; </code> Lorsque plusieurs variables sont de même type, il est possible de les déclarer sur la même ligne. Vous pouvez aussi faire une ligne par déclaration. Maintenant, il n'y a plus d'erreur.

Test Vous pouvez saisir la valeur à chercher, puis l'écran s'efface et vous pouvez saisir un premier essai.

premier joueur :



écran effacé
second joueur :



===== Etape 3 ===== ==== Comparaison de l'essai avec la valeur à trouver ==== Une fois l'essai saisi, il faut le comparer avec la valeur d'origine pour afficher le message **trop grand** si l'essai est supérieur à la valeur d'origine, ou **trop petit** si l'essai est inférieur à la valeur d'origine. Il est donc nécessaire de comparer **essai** avec **valeur**. Vous allez ainsi découvrir un nouveau principe de codage : **l'alternative (la condition)**. A la suite du code actuel, ajoutez : <code c#> *test de l'essai par rapport à la valeur à chercher if (essai > valeur) { Console.WriteLine(“ –> trop grand !”); } else { Console.WriteLine(“ –> trop petit !”); } </code> L'alternative ***if*** *contient une condition (entre parenthèses) et un ou 2 blocs : * le premier bloc s'exécute uniquement si la condition est vraie, * le second bloc (dans le* ***else****) s'exécute uniquement si la condition est fausse. La seconde partie (le* ***else****) est optionnelle. Dans le cas présent, les 2 parties sont importantes pour afficher le message correspondant. L'exécution se poursuit alors normalement après l'alternative.*

test

Vérifiez que vous obtenez bien le message (**trop grand** ou **trop petit**) correspondant à ce qui est attendu.

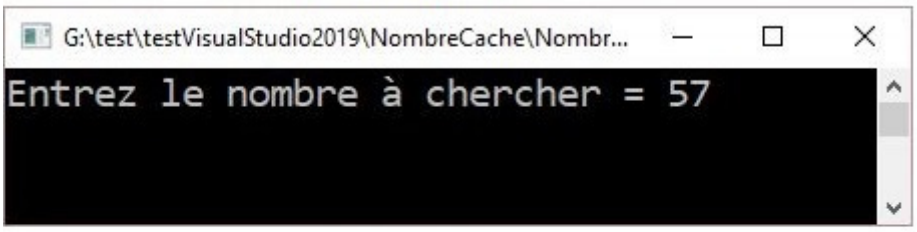
Par exemple, si la valeur est 57 et l'essai est 100, vous devriez obtenir **trop grand**). Même image que l'écran précédent avec une étape supplémentaire (par rapport au code ajouté) : affichage du premier résultat (“–

> trop grand !")

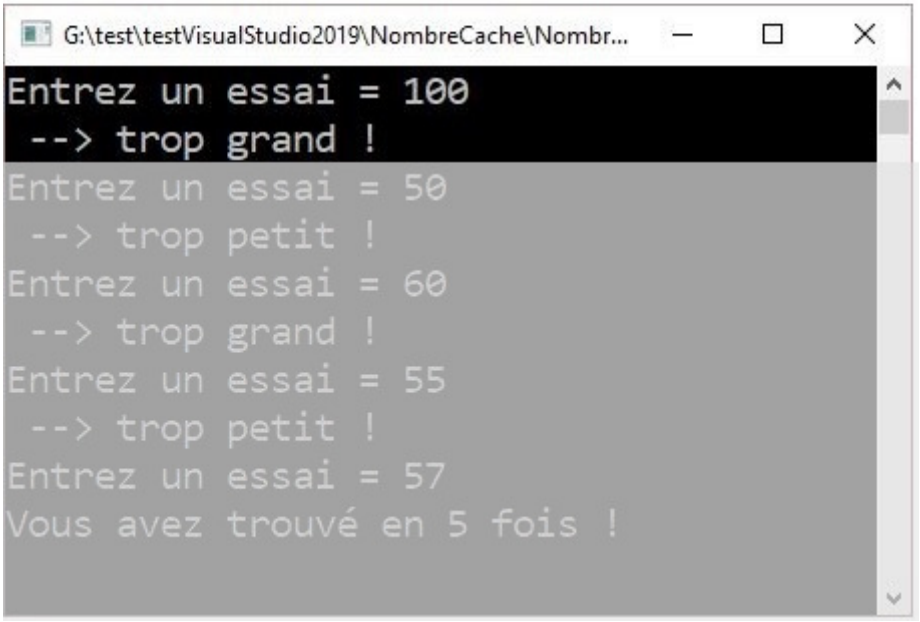
<

/WRAP>

premier joueur :



écran effacé
second joueur :



Etape 4

Saisie d'un nouvel essai ?

Il sera nécessaire de saisir plusieurs essais avant de trouver la bonne valeur, sans savoir à l'avance le nombre d'essais à saisir. Il n'est donc pas question de répéter plusieurs fois les lignes de codes précédentes : il faut faire en sorte que le programme se charge de cette répétition.

Vous allez ainsi découvrir un nouveau principe de codage : **l'itération (la boucle)** qui permet de répéter plusieurs fois une ou plusieurs lignes de code, tant qu'une **condition** est vérifiée.

Sur quoi doit-on boucler ? Sur la saisie de l'essai et la comparaison avec la valeur d'origine. Dans notre cas on veut boucler tant que la valeur n'a pas été trouvée, donc le test doit ressembler à ceci :

```
while (essai != valeur)
{
    // instructions qui vont se répéter
}
```

Le signe **!=** correspondant à **différent** en mathématique.

Le test se faisant dès le début, il faut qu'un premier essai soit saisi. Puis, dans la boucle, il faut intégrer le test déjà écrit. Mais ensuite, toujours dans la boucle, il faut saisir un nouvel essai :

```
// saisie du premier essai
Console.Write("Entrez un essai = ");
essai = int.Parse(Console.ReadLine());
while (essai != valeur)
{
    // test de l'essai par rapport à la valeur à chercher
    if (essai > valeur)
    {
        Console.WriteLine(" --> trop grand !");
    }
```

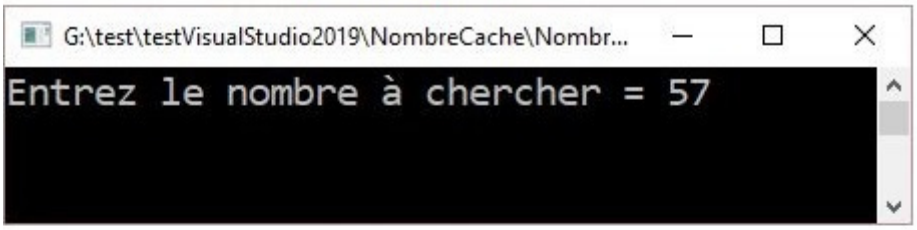
```
    else
    {
        Console.WriteLine(" --> trop petit !");
    }
    // saisie d'un nouvel essai
    Console.Write("Entrez un essai = ");
    essai = int.Parse(Console.ReadLine());
}
```

Juste après la saisie d'un nouvel essai, l'exécution revient sur la ligne du **while** est compare **essai* avec valeur. Si les 2 sont égaux, alors l'exécution sort de la boucle, sinon, le contenu de la boucle est exécuté à nouveau.**

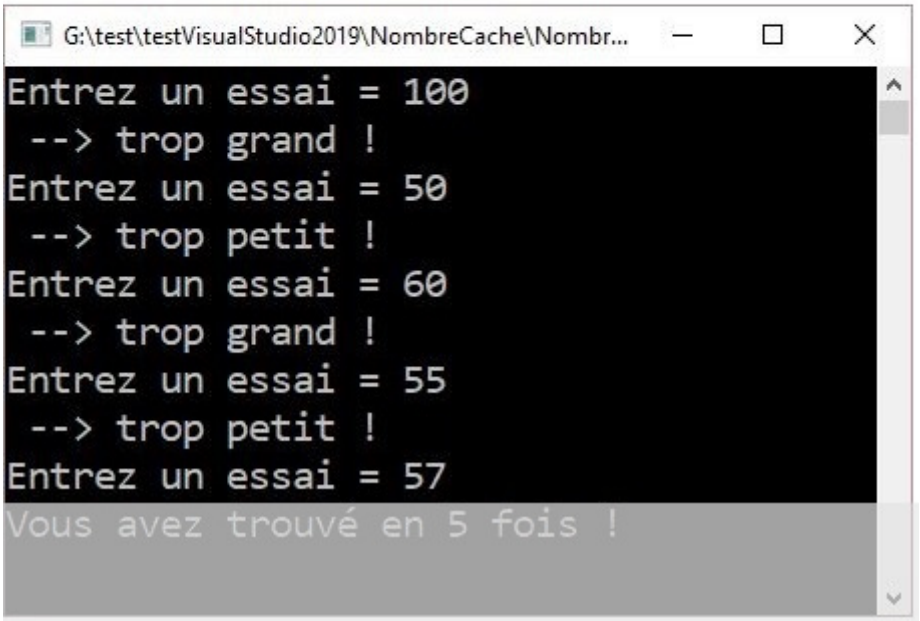
test

Vérifiez qu'à chaque essai saisi, le bon message s'affiche. Une fois la valeur trouvée, il ne se passe pour le moment rien, mais surtout, il n'est pas demandé de saisir un nouvel essai.

premier joueur :



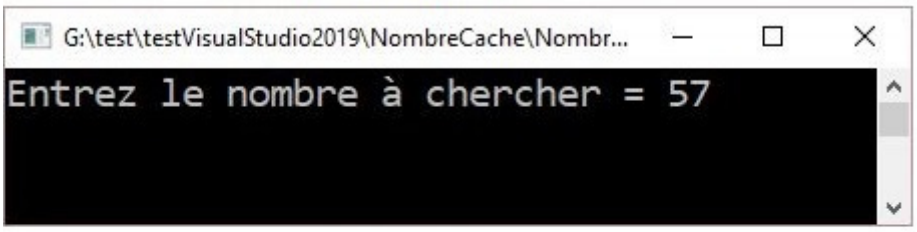
écran effacé
second joueur :



===== Etape 5 ===== ==== Valeur trouvée ==== Après la boucle, et avant que la fenêtre ne se ferme, ce serait bien d'afficher un message pour dire que la valeur a été trouvée. Ajoutez la ligne de code correspondante : <code c#> *valeur trouvée Console.WriteLine(“Vous avez trouvé”); </code>*

test Cette fois, après avoir saisi plusieurs essais, puis la bonne valeur, vous obtenez bien le message “Vous avez trouvé”.

premier joueur :



écran effacé
second joueur :