

C# : brouillon page perso "apprendre c#"

- Objet : révision c#
- Niveau requis :
[débutant](#), [avisé](#)
- Commentaires : *Contexte d'utilisation du sujet du tuto.*
- Débutant, à savoir : [Utiliser GNU/Linux en ligne de commande](#), tout commence là ! 😊

Introduction : glossaire

Structure d'un programme

```
namespace <nom_du_projet>
{
    class <nom_d'un_programme_principal>
    {
        static Main(string[] args)
        {
            <corps du programme principale>
        }
    }
}
```

MES FONCTIONS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ExosAfpA
{
    class Fonctions
    {
        public const char STOP = '.';

        /*****
        *****/
        // Fonction qui remplit un tableau de caractères

        public static char[] chainetab(int MAX)
        {
            char[] phrase;
            phrase = new char[MAX];
        }
    }
}
```

```
    string Saisie;

    int i;
    i = 0;

    //Console.WriteLine(invite);
    Console.WriteLine();
    Saisie = Lire.Chaine("Entrez votre phrase");

    while ((i < phrase.Length - 1) && (i < Saisie.Length) &&
(Saisie[i] != STOP))
    {
        phrase[i] = Saisie[i];
        i++;
    }

    // Mettre le point à la fin
    phrase[i] = STOP;

    return phrase;
}

/*****
*****/
// Fonction qui permet de créer un tableau d'entiers
// Elle déclare et affecte un tableau référencié par le nom "suite"

public static int[] tabEntier(int MAX)
{
    int[] suite;
    suite = new int[MAX];

    int i;
    i = 0;

    //Console.WriteLine(invite);
    //Console.WriteLine();
    int nbrEntiers;

    do
    {
        nbrEntiers = Lire.Entier("Combien d'entiers voulez-vous ");
    } while (nbrEntiers < 0 || nbrEntiers > MAX);

    while (i < nbrEntiers)
    {
        suite[i] = Lire.Entier("Donnez votre entier n° " + (i + 1));
    }
}
```

```
        i++;

    }
    return suite;
}
/*****
*****/
// Autre fonction qui permet de créer un tableau d'entiers
//Elle prend en entrée la référence d'un tableau du programme
principal (MAIN),
// le nombre maximum de case du tableau référencié par le programme
principal (MAIN),
// le remplit (change les valeurs),
// et renvoie (return) le tab remplit au programme principal

public static int[] tabEntier(int[] tab, int MAX)
{
    //int[] suite;
    //suite = new int[MAX];

    int i;
    i = 0;

    //Console.WriteLine(invite);
    //Console.WriteLine();

    while (i < MAX)
    {
        tab[i] = Lire.Entier("Donnez votre entier n° " + (i + 1));
        i++;
    }
    return tab;
}
/*****
*****/
// Autre fonction qui permet de créer un tableau d'entiers
public static int[] tabEntierES(int[] tableauEntrée, int MAX, out
int[] tableauSortie)
{
    int j;
    j = 0;

    while (j < MAX)
    {
        tableauEntrée[j] = Lire.Entier("Donnez votre entier n° " +
(j + 1));
        j++;
    }
}
}
```

```

/*****
*****/
    // Fonction qui permet d'afficher un tableau de caractères

public static void afficher(char[] tableauEntier)
{
    int i = 0;

    while (i < tableauEntier.Length && tableauEntier[i] != STOP)
    {
        Console.Write(tableauEntier[i]);
        i++;
    }
    Console.WriteLine(STOP);
}

/*****
*****/
    // Fonction qui permet d'afficher un tableau d'entier

public static void afficherEntier(int[] tabEntier)
{
    int i = 0;

    while (i < tabEntier.Length && tabEntier[i] != STOP)
    {
        Console.Write(tabEntier[i] + " | ");
        i++;
    }
    Console.WriteLine("");
}

/*****
*****/
    // Fonction qui permet de recommencer un programme

public static bool veutContinuer(String invite)
{
    char cara;
    do
    {
        cara = Char.ToLower(Lire.Caractere(invite));
        //if (!( (cara == 'n') || (cara == 'o') )) // selon De

        if (((cara != 'n') && (cara != 'o'))

```

Morgan

```
        {
            Console.WriteLine("Répondez oui ou non (o/n)");
        }

        } while (!((cara == 'o') || (cara == 'n')));
        return ((cara == 'o'));
    }
}
/*****
*****/
// Fonction qui rend le nombre d'occurrence(s) d'une lettre pour une
chaîne de caractères.

public static void Compt1Lettre(int MAX, char[] tabPhrase, char
occCherchee, out int nombreOcc)
{
    int i = 0;
    nombreOcc = 0;

    while ((tabPhrase[i] != STOP) && (i < MAX - 1))
    {
        if (tabPhrase[i] == occCherchee)
        {
            nombreOcc++;
        };
        i++;
    };

    Console.WriteLine("Le nombre d'occurrence de " + tabPhrase[i] +
" est " + occCherchee);
}

/*****
*****/
// Fonction qui rend la taille d'une chaîne de caractères.

public static void comptTailleChaine(char[] tableau, char Fin, out
int tailleChaine)
{
    int indice = 0;

    while (tableau[indice] != Fin)
    {
        indice++;
    };

    tailleChaine = indice;
}
/*****
*****/
```

```

// Fonction qui, pour une chaîne de caractères, rend le nombre
d'occurrence de deux lettres choisies par l'utilisateur, et accolées.

public static void compt2lett(char[] texte, char term, char lettre1,
char lettre2, out int nbreCouple)
{
    int ind = 0;           //indice du tableau de caractères et de la
string Saisie
    nbreCouple = 0;
    while (texte[ind] != term)
    {
        if (texte[ind] == lettre1)
        {
            if ((texte[ind + 1] == lettre2) && (texte[ind + 1] !=
term))
            {
                nbreCouple++;
                ind++;
            }
        }
        ind++;
    }
    Console.WriteLine("Le couple \"" + lettre1 + "\" ET \"" +
lettre2 + "\" apparaît dans la phrase \"" + nbreCouple + "\" fois.");
}

/*****
*****
// Cette fonction permet de détecter si une chaîne terminée par un
point est un palindrome.*****/

public static void fauxPalindrome(int MAX, char[] phrase, out bool
result)
{
    //char term = '.';
    int j = 0;

    result = false;

    while ((phrase[j] != Palindrome.term) && (j < MAX - 1))
    {
        j = j + 1;
    };

    j--;

    int i = 0;

    while ((i < j) && (phrase[i] == phrase[j]))
    {
        i++;
    }
}

```

```

        j--;
    };

    if (i >= j)
    {
        result = true;
        //Console.WriteLine("C'est un palindrome."); // A ne pas
mettre dans la fonction : elle fait rien d'autre que ce qu'elle fait !!!!
    }
    else
    {
        result = false;
        //Console.WriteLine("Ce n'est pas un palindrome.");
    }
}

/*****
*****/
// Cette fonction permet de trier un tableau par la méthode du tri à
bulles.
// On entre un nombre d'entiers qui seront triés du plus petit au
plus grand.

public static void TriBulles(ref int[] table, int Igutile)
{
    int i;
    bool inversion;
    int tampon;

    do
    {
        inversion = false;
        i = 0;
        while( i < Igutile - 1)
        {
            if (table[i] > table[i + 1])
            {
                tampon = table[i];
                table[i] = table[i + 1];
                table[i + 1] = tampon;
                inversion = true;
            };
            i = i + 1;
        };
    }
    while(!(inversion));
}

/*****
*****/
// Cette fonction permet de trier un tableau d'entier par
dichotomie.

```

```

    public static void TriDichotomie(int[] tabprenom, int lgutile, int
search, out int indice)
    {
        int idebut;
        int ifin;

        idebut = 0;
        ifin = lgutile - 1;
        indice = (idebut + ifin) / 2;
        while ((idebut < ifin) && (tabprenom[indice] != search))
        {
            if (tabprenom[indice] > search)
            {
                ifin = indice - 1;
            }
            else
            {
                idebut = indice + 1;
            }
            indice = (idebut + ifin) / 2;
        }
        if ((idebut > ifin) || (tabprenom[indice] != search))
        {
            indice = 0; // indice = indice - 1
        }
    }
}
/*****
*****
    CHERCHER MOT
    * Cette fonction permet de chercher un mot dans une chaîne de
caractères.
    Si le mot est se trouve dans la chaîne, alors le résultat est vrai.
*/
    public static void chercherMot(char[] texte, string motCherche, int
longueurMotCherche, out bool trouvé)
    {
        int i = 0;
        int taille_mot;
        trouvé = false;

        do
        {
            Fonctions.prendreMot(texte, ref i, out taille_mot);
            if(taille_mot == longueurMotCherche)
            {
                Fonctions.comparerMot(texte, i, longueurMotCherche,
motCherche, out trouvé);
                Console.WriteLine("fonction chercher Mot");
            }
        } while (!(trouvé) && (taille_mot != 0)); // => En algo:

```



```

jusqu'à trouvé OU (taille_mot = 0) car :
//
//   repeter
//
prendreMot(texte, i, long, motCherché, trouvé)
//   si
taille_mot = long Alors
//
comparerMot(texte, i, long, motCherché, trouvé)
//   finisi
//   jusqu'à trouvé OU
(taille_mot = 0)
}
/*****
*****
PRENDRE MOT
* Cette procédure positionne un indice sur le caractère suivant le
mot repéré et donne sa longueur
ou positionne l'indice sur le caractère terminateur et donne la
longueur zéro*/
public static void prendreMot(char[] texte, ref int ind, out int lg)
{
    const char carterm = '.';
    const char espace = ' ';

    while(texte[ind] == espace)
    {
        ind++;
    }
    lg = 0;

    while((texte[ind] != espace) && (texte[ind] != carterm))
    {
        ind++;
        lg++;
    }
}
/*****
*****
COMPARER MOT
* Cette procédure positionne compare un mot repéré dans la chaîne
par sa position "après" d'un mot donné.
Les deux mots ont la même longueur.*/
public static void comparerMot(char[] texte, int ind, int longmot,
string mot, out bool égal)
{
    ind = ind - 1;

    while ( (longmot != 0) && (texte[ind] == mot[longmot-1]) ) //
-1 car indice à zéro en c#
    {

```

```

        ind = ind - 1;
        longmot = longmot - 1;
    }
    égal = (longmot == 0);
}

/*****
*****
*
*****
*****

    RECOPIER PHRASE EN INVERSANT CHAQUE MOT / Procédure
phraseInversChaqMot

    public static void inverserPhrase(char[] texte, out char []
phraseInversee)
    {
        const char ESPACE = ' ';    // !!! Voir constantes déclarées
avant le main du prog principal,
                                //appelée ici avec
nomProgrammePricipal.nomConstante (voir ci-dessous pour ESPACE et carterm)
        const char carterm = '.';
        const int taille = 80;

        int iChaine = 0;            // Indice de parcours du texte
donnée début zéro en c#
        int iChaineInvers = 0;      // Indice de parcours de la chaine
inversée début zéro en c#
        int taille_mot;            // longueur d'un mot repéré

        phraseInversee = new char[taille]; // initialisation de la
phrase inversee (.taille)

        Fonctions.prendreMot(texte, ref iChaine, out taille_mot);

        while( taille_mot != 0 )
        {
            Fonctions.inverserMot(texte, iChaine, taille_mot, ref
iChaineInvers, ref phraseInversee);
            Fonctions.prendreMot(texte, ref iChaine, out taille_mot);
            if(taille_mot != 0)
            {
                phraseInversee[iChaineInvers] = ESPACE;
                iChaineInvers++;
            }
        }
        phraseInversee[iChaineInvers] = carterm;
    }
/*****
*****/

```

```

*****
    * INVERSER MOT : Cette fonction permet d'inverser l'ordre des
lettres d'un mot
    public static void inverserMot(char[] texte, int ind, int longmot,
ref int indinv, ref char[] chaineInversee )
    {
        ind = ind - 1;

        while(longmot != 0)
        {
            chaineInversee[indinv] = texte[ind];
            indinv = indinv + 1;
            ind = ind - 1;
            longmot = longmot - 1;
        }
    }
}
/*****
*****
    *
*****
*****

    Justifier une phrase / Procédure justifier

public static void justifier(char[] texte, out char[] texte_jus)
{
    const char ESPACE = ' ';
    const int taille = 15;
    const char carterm = '.';

    int i_t; // pour comparer les mots
    int i_j; // pour comparer les caractères à copier
    int taille_mot; // longueur d'un mot repéré
    int nb_mots; // nombre de mots dans le texte
    int nb_lettres; // nombre de caractères utiles du texte
    int intervalle; // nombre d'espaces à mettre entre les mots
    int reste; // espaces à répartir entre les mots
    int nb_espaces; // nombre d'espaces à mettre après le mot copié

    i_t = 1;
    nb_mots = 0;

    Fonctions.prendreMot(texte, ref i_t, out taille_mot);

    while(taille_mot != 0)
    {
        nb_mots = nb_mots + 1;
        nb_lettres = (nb_lettres + taille_mot);
        Fonctions.prendreMot(texte, ref i_t, out taille_mot);
    }
}

```

```

        if(nb_mots > 1)
        {
            intervalle = ((taille - nb_mots - 1) / (nb_mots - 1));
            reste = (taille - nb_lettres - 1) % (nb_mots - 1);
        }

        i_t = 1;
        i_j = 1;

        while(nb_mots != 0)
        {
            Fonctions.prendreMot(texte, ref i_t, out taille_mot);
            Fonctions.copierMot(texte, i_t, taille_mot, i_j,
texte_jus);

            nb_mots = nb_mots - 1;

            if(nb_mots != 0)
            {
                nb_espaces = intervalle;
                if(reste != 0)
                {
                    reste = reste - 1;
                    nb_espaces = nb_espaces + 1;
                }

                while(nb_espaces == 0)
                {
                    texte_jus[i_j] = ESPACE;
                    i_j = i_j + 1;
                    nb_espaces = nb_espaces - 1;
                }
            }
        }

        texte_jus[i_j] = carterm;

    }
}*/

/*****
*****
Fonction qui permet de "LIRE" les types basiques

*****/

    /// <summary>
    /// La méthode Entier permet de lire un entier au clavier.
    /// Tant que la saisie est incorrecte, une saisie sera demandée à
l'utilisateur.
    /// </summary>

```

```
    /// <param name="invite">C'est l'invite affichée à l'écran pour
solliciter la saisie</param>
    /// <returns>La valeur de l'entier saisi.</returns>
    public static int Entier(String invite)
    {
        int num = 0;
        string userEntry = "";
        bool isOk = false;
        do
        {
            Console.Write(invite + " : ");
            userEntry = Console.ReadLine();
            isOk = int.TryParse(userEntry, out num);
            if (!isOk) Console.WriteLine("Erreur de saisie. Une valeur
entière est attendue.");
        } while (!isOk);

        return num;
    }

    /// <summary>
    /// La méthode Double permet de lire un décimal long au clavier.
    /// Tant que la saisie est incorrecte, une saisie sera demandée à
l'utilisateur.
    /// </summary>
    /// <param name="invite">C'est l'invite affichée à l'écran pour
solliciter la saisie</param>
    /// <returns>La valeur du double saisi.</returns>
    public static double Double(String invite)
    {
        double num = 0;
        string userEntry = "";
        bool isOk = false;
        do
        {
            Console.Write(invite + " : ");
            userEntry = Console.ReadLine();
            isOk = double.TryParse(userEntry, out num);
            if (!isOk) Console.WriteLine("Erreur de saisie. Une valeur
décimale étendue est attendue.");
        } while (!isOk);

        return num;
    }

    /// <summary>
    /// La méthode Float permet de lire un décimal simple précision au
clavier.
    /// Tant que la saisie est incorrecte, une saisie sera demandée à
l'utilisateur.
    /// </summary>
```

```
    /// <param name="invite">C'est l'invite affichée à l'écran pour
solliciter la saisie</param>
    /// <returns>La valeur du float saisi.</returns>
    public static float Float(String invite)
    {
        float num = 0;
        string userEntry = "";
        bool isOk = false;
        do
        {
            Console.Write(invite + " : ");
            userEntry = Console.ReadLine();
            isOk = float.TryParse(userEntry, out num);
            if (!isOk) Console.WriteLine("Erreur de saisie. Une valeur
décimale est attendue.");
        } while (!isOk);

        return num;
    }

    /// <summary>
    /// La méthode Caractere permet de lire un caractère au clavier.
    /// Tant que la saisie est incorrecte ( touche Entrée sans avoir
frappé un caractère ) une saisie sera demandée.
    /// </summary>
    /// <param name="invite">C'est l'invite affichée à l'écran pour
solliciter la saisie</param>
    /// <returns>la valeur du caractère saisi.</returns>
    public static char Caractere(String invite)
    {
        char car = ' ';
        string userEntry = "";
        bool isOk = false;
        do
        {
            Console.Write(invite + " : ");
            userEntry = Console.ReadLine();
            isOk = (userEntry!="");
            if (!isOk) Console.WriteLine("Erreur de saisie. Un caractère
est attendu.");
            else car = userEntry[0];
        } while (!isOk);

        return car;
    }

    /// <summary>
    /// La méthode Chaine permet de lire une chaîne de caractères au
clavier.
    /// la saisie est forcément correcte, la chaîne pouvant être vide.
    /// </summary>
```

```
    /// <param name="invite">C'est l'invite affichée à l'écran pour
solliciter la saisie</param>
    /// <returns>la chaîne de caractères saisie.</returns>
    public static String Chaine(String invite)
    {
        Console.Write(invite + " : ");
        return Console.ReadLine();
    }
    /// <summary>
    /// La méthode PetitEntier permet de lire un short au clavier.
    /// Tant que la saisie est incorrecte, une saisie sera demandée à
l'utilisateur.
    /// </summary>
    /// <param name="invite">C'est l'invite affichée à l'écran pour
solliciter la saisie</param>
    /// <returns>La valeur du short saisi.</returns>
    public static short PetitEntier(string invite)
    {
        short num = 0;
        string userEntry = "";
        bool isOk = false;
        do
        {
            Console.Write(invite + " : ");
            userEntry = Console.ReadLine();
            isOk = short.TryParse(userEntry, out num);
            if (!isOk) Console.WriteLine("Erreur de saisie. Une valeur
entière est attendue.");
        } while (!isOk);

        return num;
    }
    public static bool Question(string invite)
    {
        string userEntry = "";
        bool isOk = false;
        do
        {
            Console.Write(invite + " : ");
            userEntry = Console.ReadLine().ToLower();
            isOk = (userEntry.Length != 0) && (userEntry[0] == 'o' ||
userEntry[0] == 'n');
            if (!isOk)
            {
                Console.WriteLine("Erreur de saisie. un caractère o, O,
n ou N est attendu.");
            }
        } while (!isOk);

        return (userEntry.ToLower()[0] == 'o');
```

```
    }  
  
    internal static char Caracter(string p)  
    {  
        throw new NotImplementedException();  
    }  
  
    internal static char Caractère(string p)  
    {  
        throw new NotImplementedException();  
    }  
  
    } // celui de la class Fonctions  
} // celui de namespace Tableau
```

Programme d'appel de mes fonctions

From:

<http://debian-facile.org/> - **Documentation - Wiki**

Permanent link:

<http://debian-facile.org/utilisateurs:hypathie:tutos:c>

Last update: **20/12/2014 15:59**

