

EXERCICES RESOLUS EN C#

Aimé DIUMI DIKOLO

www.wissen-corp.com

Dédicace à

Julie APAMI DIKOLO

Pauline TSHULU DIKOLO

Thérèse OTAKOTSHA DIKOLO

Albertine MBOHELAKA DIKOLO

QUESTION 1

Ecrire un programme C# qui reconnait si un nombre est de Kaprekar. Un nombre n est un nombre de Kaprekar en base 10, si la représentation décimale de n^2 peut être séparée en une partie gauche u et une partie droite v tel que $u + v = n$

Exemple : $45^2 = 2025$, comme $20 + 25 = 45$, 45 est un nombre de Kaprekar.

$4879^2 = 23804641$, comme $238 + 04641 = 4879$ (le 0 de 04641 est inutile, je l'ai juste placé pour éviter toute confusion), alors 4879 est encore un nombre de Kaprekar.

Ce programme comprendra :

- ✓ 1. Une fonction public static int sommeParties(int n, int p) qui découpe n en deux nombres dont le deuxième comporte p chiffres, et qui retourne leur somme. Par exemple, sommeParties(12540,2)=125+40=165
- ✓ 2. Une fonction public static bool estKaprekar(int n)

(Interro 2019)

Résolution

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

namespace QUESTION1
{
    //Cette classe permet de définir des variables globales
    static class Classe
    {
        public static int u=0;
        public static int v=0;
        public static int s=0;
    }

    class Program
    {
        static void Main(string[] args)
        {
            int n = 0, carreN = 0, i=0,p=0, t=0;
            bool test=false;
            String ncarre;
            Console.WriteLine("Entrez la valeur de n");
            n = Int32.Parse(Console.ReadLine());
            carreN = n * n;
            //On convertit le nombre en chaine pour connaitre aisement le nombre de
            chiffres
        }
    }
}
```

```

ncarre = carreN.ToString();
t = ncarre.Length;
// Appel des fonctions
for (i=1; i<t;i++)
{
    p = i;
    Classe.s=sommeParties(carreN, p);
    test = estKaprekar(n);
    if (test==true)
    {
        Console.WriteLine("Le nombre " + n + " est un nombre de Kaprekar car
");
        Console.WriteLine(" " + n + "*" + n + "=" + carreN + " et " + Classe.u +
"+" + Classe.v + "=" + n);
        break;
    }
}
if(test==false)
{
    Console.WriteLine("Ce nombre n'est pas un nombre de Kaprekar ");
}

Console.ReadLine();
}
//La fonction qui va découper
public static int sommeParties(int n, int p)
{
    int somme = 0, t = 0, i = 0, j = 1;
    String ncarre, vInter="", uInter="";
    ncarre = n.ToString();
    t = ncarre.Length;
    int[] chiffres = new int[t];
    for (i = 0; i < t; i++)
    {
        chiffres[i] = Int32.Parse(ncarre.Substring(i, 1));
    }
    //On forme la partie droite
    for(i=t-1; j<= p; i-- )
    {
        vInter =chiffres[i].ToString()+ vInter;
        j++;
    }
    //on forme la partie gauche
    int k;
    j = 1;
    k = t - p;
    for (i = 0; j <= k; i++)
    {
        uInter = uInter + chiffres[i].ToString();
        j++;
    }
    Classe.u = Int32.Parse(uInter);
    Classe.v = Int32.Parse(vInter);

    somme = Classe.u + Classe.v;
    return somme;
}

```

```

//La fonction booléenne
public static bool estKaprekar(int n)
{
    bool test;
    if (Classe.s==n)
    {
        test = true;
    }
    else
    {
        test = false;
    }
    return test;
}
}

```

Illustrations de l'exécution

file:///E:/A.DIUMI/EXERCICES/BATTS RESOLUS C#/QUESTION1/QUESTION1/bin/Debug/QUESTION1.EXE

Entrez la valeur de n
45
Le nombre 45 est un nombre de Kaprekar car
45*45=2025 et 20+25=45
-

file:///E:/A.DIUMI/EXERCICES/BATTS RESOLUS C#/QUESTION1/QUESTION1/bin/Debug/QUESTION1.EXE

Entrez la valeur de n
4879
Le nombre 4879 est un nombre de Kaprekar car
4879*4879=23804641 et 238+4641=4879

file:///E:/A.DIUMI/EXERCICES/BATTS RESOLUS C#/QUESTION1/QUESTION1/bin/Debug/QUESTION1.EXE

Entrez la valeur de n
234
Ce nombre n'est pas un nombre de Kaprekar

QUESTION 2

Soient :

- Une classe exécutable Article composée des attributs suivants :
 - Code_article
 - Désignation_article
 - Prix_article
 - Catégorie_article
- Une collection ARTICLES permettant le stockage des différents articles

Questions :

- a) Ecrire la classe **Article** (les attributs ne peuvent être visibles que dans les classes filles de la classe article)
- b) Ecrire un constructeur d'initialisation
- c) Ecrire la méthode **toString ()** qui renvoie toutes les propriétés séparées par un point-virgule
- d) Ecrire la classe **ArticleSpécial** et redéfinir la méthode toString()
- e) Donner la requête LINQ affichant les propriétés de tous les articles qui coutent au moins 1000 FC.
- f) Donner la requête LINQ affichant les articles par catégorie

(Examen S1 2017-2018)

Résolution

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

namespace EXERCICE_2
{
    class Article
    {
        protected int Code_article;
        protected string Désignation_article;
        protected int Prix_article;
        protected string Catégorie_article;
    }
}
```

b) //Constructeur sans paramètres

```
public Article()
{
    this.Code_article = 0;
    this.Designation_article = "Article non défini";
    this.Prix_article = 0;
    this.Categorie_article = "Catégorie non définie";
}
```

//Constructeur avec des paramètres

```
public Article(int code, string designation, int prix, string categorie)
{
    this.Code_article = code;
    this.Designation_article = designation;
    this.Prix_article = prix;
    this.Categorie_article = categorie;
}
```

c) public virtual string toString()

```
{
    return "Code article : " + this.Categorie_article + "; Désignation article: "
+ this.Designation_article + "; Prix article : " + this.Prix_article + "; Catégorie
article: " + this.Categorie_article;
}
```

} Suivant ce qui a été demandé en a), b) et c) le code complet est :

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

```
namespace EXERCICE_2
```

```
{
```

```
    class Article
```

```
    {
```

```
        protected int Code_article;
        protected string Designation_article;
        protected int Prix_article;
        protected string Categorie_article;
```

//Constructeur sans paramètres

```
public Article()
{
    this.Code_article = 0;
    this.Designation_article = "Article non défini";
    this.Prix_article = 0;
    this.Categorie_article = "Catégorie non définie";
}
```

//Constructeur avec des paramètres

```
public Article(int code, string designation, int prix, string categorie)
{
    this.Code_article = code;
    this.Designation_article = designation;
    this.Prix_article = prix;
    this.Categorie_article = categorie;
}
```

```

    }

    public virtual string toString()
    {
        return "Code article : " + this.Categorie_article + "; Désignation article: "
+ this.Designation_article + "; Prix article : " + this.Prix_article + "; Catégorie
article: " + this.Categorie_article;
    }
}

```

- d) Pour la redéfinition de la méthode toString, j'ai juste préféré que la méthode renvoie la désignation de l'article ainsi que la description de l'article (attribut que j'ai ajouté à la classe ArticleSpecial)

Comme la valeur de retour de la méthode n'a pas été précisée, vous êtes libre de la redéfinir comme bon vous semble

```

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

namespace EXERCICE_2
{
    class ArticleSpecial: Article
    {
        private string Description_article;

        //Constructeur sans paramètres
        public ArticleSpecial() : base()
        {
            this.Description_article = "Je suis un article spécial";
        }

        //Constructeur avec des paramètres
        public ArticleSpecial (int code, string designation,int prix, string categorie,
string description) : base( code, designation, prix, categorie)
        {
            this.Code_article = code;
            this.Designation_article = designation;
            this.Prix_article = prix;
            this.Categorie_article =categorie;
            this.Description_article = description;
        }

        //Redéfinition de la méthode toString
        public override string toString()
        {
            return "Désignation Article: "+this.Designation_article +"; Description:
"+this.Description_article;
        }
    }
}

```



```

e) var requete = from Article in COLLECTIONS
                  where Article.Prix_article >= 1000
                  select new
                  {
                      Article.Code_article,
                      Article.Designation_article,
                      Article.Prix_article,
                      Article.Categorie_article
                  };
Console.WriteLine("Voici les propriétés des articles qui coutent au moins
1000FC");

foreach (var art in requete)
{
    Console.WriteLine(" Code: {0}, Désignation: {1}, Prix: {2}, Catégorie:
{3}", art.Code_article, art.Designation_article, art.Prix_article, art.Categorie_article);
}

```

```

f) var requete = from art in COLLECTIONS
                  group art by art.Categorie_article;

Console.WriteLine("Voici les désignations des articles groupés par
catégorie");

foreach (var element in requete)
{
    Console.WriteLine("La catégorie: {0} a {1} produits(s) qui (est) sont:",
element.Key, element.Count());

    foreach (Article art in element)
    {
        Console.WriteLine("{0} ", art.Designation_article);
    }
}

```

QUESTION 3

Proposez une liste de codes C# permettant à l'utilisateur de jouer au jeu illustré à l'interface ci-dessous :

Wissen corporation

QCM

1. La RDC a eu son indépendance en (sur 4)

☐ 1961 ☐ 1930 ☐ 1960 ☐ 1990

2. Le parlement de la RDC a combien de chambres (sur 3)

☐ 2 ☐ 3 ☐ 4 ☐ 1

3. Le chef de l'Etat congolais est élu pour (sur 3)

☐ 5 ☐ 2 ☐ 4 ☐ 10

Résultat

Valider

(Support)

Résolution

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EXERCICE_3
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void checkBox1_1961_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1_1961.Checked==true)
            {

```

```

        checkBox1_1930.Enabled = false;
        checkBox1_1960.Enabled = false;
        checkBox1_1990.Enabled = false;
    }
}

private void checkBox1_1930_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1_1930.Checked == true)
    {
        checkBox1_1961.Enabled = false;
        checkBox1_1960.Enabled = false;
        checkBox1_1990.Enabled = false;
    }
}

private void checkBox1_1960_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1_1960.Checked == true)
    {
        checkBox1_1930.Enabled = false;
        checkBox1_1961.Enabled = false;
        checkBox1_1990.Enabled = false;
    }
}

private void checkBox1_1990_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox1_1990.Checked == true)
    {
        checkBox1_1930.Enabled = false;
        checkBox1_1960.Enabled = false;
        checkBox1_1961.Enabled = false;
    }
}

private void checkBox2_2_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox2_2.Checked==true)
    {
        checkBox2_1.Enabled = false;
        checkBox2_3.Enabled = false;
        checkBox2_4.Enabled = false;
    }
}

private void checkBox2_3_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox2_3.Checked == true)
    {
        checkBox2_1.Enabled = false;
        checkBox2_2.Enabled = false;
        checkBox2_4.Enabled = false;
    }
}

```

```

private void checkBox2_4_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox2_4.Checked == true)
    {
        checkBox2_1.Enabled = false;
        checkBox2_3.Enabled = false;
        checkBox2_2.Enabled = false;
    }
}

private void checkBox2_1_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox2_1.Checked == true)
    {
        checkBox2_2.Enabled = false;
        checkBox2_3.Enabled = false;
        checkBox2_4.Enabled = false;
    }
}

private void checkBox3_5_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox3_5.Checked == true)
    {
        checkBox3_10.Enabled = false;
        checkBox3_2.Enabled = false;
        checkBox3_4.Enabled = false;
    }
}

private void checkBox3_2_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox3_2.Checked == true)
    {
        checkBox3_10.Enabled = false;
        checkBox3_5.Enabled = false;
        checkBox3_4.Enabled = false;
    }
}

private void checkBox3_4_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox3_4.Checked == true)
    {
        checkBox3_10.Enabled = false;
        checkBox3_2.Enabled = false;
        checkBox3_5.Enabled = false;
    }
}

private void checkBox3_10_CheckedChanged(object sender, EventArgs e)
{
    if (checkBox3_10.Checked == true)
    {
        checkBox3_5.Enabled = false;
        checkBox3_2.Enabled = false;
        checkBox3_4.Enabled = false;
    }
}

```

```

    }

    private void Form1_Load(object sender, EventArgs e)
    {
        textBoxResultat.ReadOnly = true;
    }

    private void buttonValider_Click(object sender, EventArgs e)
    {
        int cote = 0;
        if (checkBox1_1960.Checked == true)
        {
            cote += 4;
        }
        if (checkBox2_2.Checked == true)
        {
            cote += 3;
        }
        if (checkBox3_5.Checked == true)
        {
            cote += 3;
        }
        textBoxResultat.Text = cote.ToString();
    }
}

```

QUESTION 4

Ecrire un programme C# qui lit un mot, si le mot possède 7 caractères, le programme trouve le mot miroir de ce mot, dans le cas contraire le programme renvoie le nombre d'occurrence de chaque caractère. Pour cela créer une classe Mot qui possède deux méthodes :

- a) La méthode TrouveMotMiroir
- b) NombreOccurence

(Interro 2019)

Résolution

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

namespace EXERCICE_4
{
    class Mot
    {
        static void Main(string[] args)
        {
            string mot = "";
            Console.WriteLine("Entrez le mot");
            mot = Console.ReadLine();
            if (mot.Length==7)
            {
                TrouveMotMiroir(mot);
            }
            else
            {
                NombreOccurence(mot);
            }
            Console.ReadLine();
        }

        //Méthode NombreOccurence
        public static void NombreOccurence(string m)
        {
            //Tableau qui stocke les caractères de la chaîne
            string[] lettres = new string[26];
            int i = 0, j=1, k=0;
            string l;
            //On recherche les caractères composant la chaîne
            lettres[0] = m.Substring(0, 1);
            for(i=1; i<m.Length; i++)
            {
                l = m.Substring(i, 1);
                //on vérifie si le caractère n'est pas déjà dans le tableau
                if (Array.IndexOf(lettres, l) == -1)
```

```

        {
            lettres[j] = 1;
            j++;
        }
    }
    //On compte le nombre d'occurrence
    int[] occ = new int[j];
    for(i=0;i< j;i++)
    {
        occ[i] = 0;
    }

    for(i=0;i< j;i++)
    {
        for(k=0;k<m.Length;k++)
        {
            if (lettres[i].CompareTo(m.Substring(k,1))==0)
            {
                occ[i]++;
            }
        }
    }
    //Affichage du resultat
    Console.WriteLine("Voici les occurrences de chaque lettre");
    for(i=0;i< j;i++)
    {
        Console.WriteLine("La lettre " + lettres[i] + " est reprise " + occ[i] + "
fois");
    }
}

//méthode TrouveMotMiroir
public static void TrouveMotMiroir(string m)
{
    int i=0;
    char c;
    //Le tableau qui va contenir les lettres du mot
    char[] A = m.ToCharArray();

    for(i=0;i<3;i++)
    {
        c = A[i];
        A[i] = A[7 - i - 1];
        A[7 - i - 1] = c;
    }
    // Reconstitution de la chaîne à partir du tableau des caractères
    m = new string(A);
    Console.WriteLine("Le mot miroir est " + m);
}
}
}

```

EXERCICE 5

Evaluer les expressions suivantes en supposant

$$A = 40 \quad B = 10 \quad C = 20 \quad D = 4 \quad X = 24 \quad Y = 30$$

Noter chaque fois la valeur rendue comme résultat de l'expression et les valeurs des variables dont le contenu a changé

(1) $(5 * X) + 2 * ((3 * B) + 4)$

(2) $(5 * (X + 2) * 3) * (B + 4)$

(3) $A == (B += 5)$

(4) $A += (-X)$

(5) $A ! = (C * = (D))$

(6) $A * = C + (X.D)$

(7) $A \% = D + +$

(8) $A \% = + + D$

(9) $(X + +) * (A + (-C))$

(10) $A = X * (B < C) + Y * !(B < C)$

(Interro 2019)

Résolution

(1) $(5 * X) + 2 * ((3 * B) + 4)$

$$(5 * 24) + 2 * ((3 * 10) + 4)$$

$$120 + 2 * (30 + 4)$$

$$120 + 2 * 34$$

$$120 + 68$$

$$188$$

Le résultat de l'expression est 188.

(2) $(5 * (X + 2) * 3) * (B + 4)$

$$(5 * (24 + 2) * 3) * (10 + 4)$$

$$(5 * 26 * 3) * 14$$

$390 * 14$

5460

Le résultat de l'expression est 5460

(3) $A == (B += 5)$

$40 == (B = 10 + 5)$

$40 == B$

$40 == 15$ *B vaut maintenant 15*

False

Le résultat de l'expression est **False** et **B=15**

(4) $A += (-X)$

$A = 40 + (-24)$

$A = 40 - 24$

$A = 16$

EXERCICE 6

- a) Donnez le parallélisme entre la spécialisation et la généralisation
- b) Quelle est la classe qu'on peut définir qui prouve qu'un attribut est de la classe ?
- c) Comment appelle-t-on la méthode qui n'a pas de type, qui a le même nom que la classe ? Et c'est quoi son rôle ?
- d) Donner un exemple clair d'une classe et d'un objet, donner la description d'une classe en C# et montrer comment on peut l'instancier.

(Interro 2019)

Résolution

- a) Le parallélisme entre la généralisation et spécialisation est que les deux s'appliquent à un même type de relation entre classes (Héritage) ; on peut employer l'un ou l'autre selon le sens de lecture.
- b) Le constructeur, il crée et initialise une instance de la classe

EXERCICE 7

Sans définir, donnez la différence entre :

- a) Une collection et un tableau
- b) Un attribut de classe et d'instance
- c) Le LINQ et le SQL
- d) Un langage orienté objet et un langage procédural

(Examen S1 2017-2018)

Résolution

- a) La différence réside dans le fait que le tableau est de taille fixe alors que la collection est de taille variable.

EXERCICE 8

- a) On enregistre dans une liste Gly, le taux de glycémie des différents patients dans un centre hospitalier. La glycémie est anormale, si elle est supérieure à 120. Pour des personnes dont la glycémie est anormale, on leur administre le glucophage, par contre les personnes dont la glycémie est normale, on leur recommande les exercices physiques. Ecrire un programme en Csharp qui calcule la moyenne des glycémies anormales. L'effectif des patients enregistrés est saisi par l'utilisateur à partir du clavier. Le programme calculera aussi la proportion des personnes dont la glycémie est anormale.
- b) Définir les concepts suivants :
Classe, Objet, Accesseur
- c) Etablir la différence entre la programmation procédurale et la programmation orientée objet

(Examen)

Résolution

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

namespace QUESTION_8
{
    class Program
    {
        static void Main(string[] args)
        {
            //Création de la liste
            List<int> Gly = new List<int>();
            int n = 0, i=0, g=0;
label1:      Console.WriteLine("Entrez le nombre de patients");
            n = Int32.Parse(Console.ReadLine());
            if(n<=0 )
            {
                Console.WriteLine("Le nombre de patients incorrect, recommencez");
                goto label1;
            }

            //Lecture des glycémies des patients
            for(i=1; i<= n;i++)
            {
                label2: Console.WriteLine("Entrez la glycémie du patient no " + i);
                g = Int32.Parse(Console.ReadLine());
                if (g<0)
                {
                    Console.WriteLine("La glycémie incorrecte, recommencez");
                    goto label2;
                }

                Gly.Add(g);
            }
            //Calcul de la moyenne et de la proportion des glycémies anormales
            int t = 0, somme=0;
            float moy = 0, p = 0;
            foreach(int val in Gly)
            {
                if(val>120)
                {
                    somme += val;
                    t++;
                }
            }

            if(t==0)
            {
                moy = 0;
            }
            else
            {
                moy = somme / t;
            }
        }
    }
}
```

```

        p = (t * 100) / n;
        //Affichage de résultat
        Console.WriteLine("La moyenne des glycémies anormales est de " + moy);
        Console.WriteLine("La proportion des personnes dont la glycémie est anormale
est de " + p + "%");
        Console.ReadLine();
    }
}
}

```

EXERCICE 9

On dénomme nombre d'Armstrong un entier naturel qui est égal à la somme des cubes des chiffres qui le composent, faire un programme C# décrivant ce problème.

(Support page 65)

Résolution

```

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

namespace QUESTION_9
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = 0, i=0;
            string chn = "";
            int sommeCube = 0;
            label1: Console.WriteLine("Entrez un entier naturel");
            n = Int32.Parse(Console.ReadLine());
            if (n<=0)
            {
                Console.WriteLine("Nombre invalide, recommencez");
                goto label1;
            }

            //On convertit le nombre en chaine de caractères pour la manipulation facile
            de chaque chiffre
            chn = n.ToString();
            //On fait la somme des cubes des chiffres qui composent le nombre
            for(i=0;i<chn.Length;i++)
            {
                sommeCube += (int)Math.Pow(Int32.Parse(chn.Substring(i, 1)), 3);
            }
            if(sommeCube==n)
            {
                Console.WriteLine(" " + n + " est un nombre d'Armstrong");
            }
        }
    }
}

```

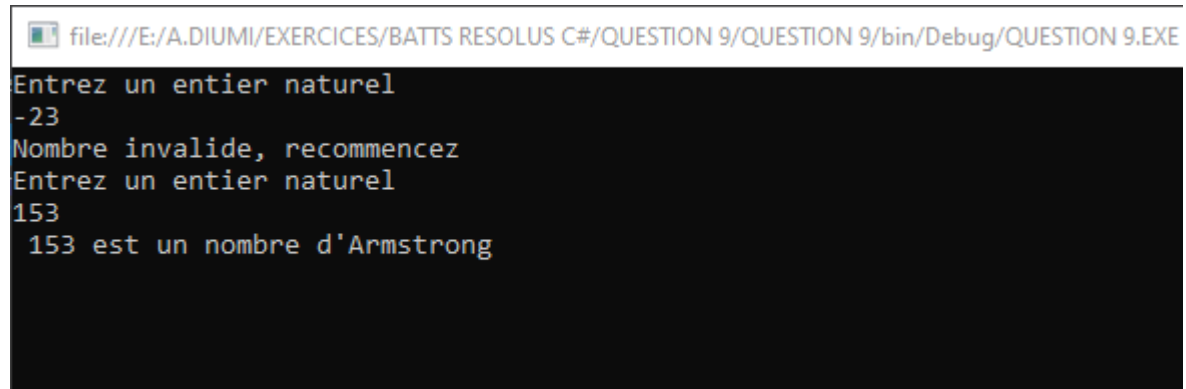
```

        else
        {
            Console.WriteLine(" " + n + " n'est pas un nombre d'Armstrong");
        }

        Console.ReadLine();
    }
}

```

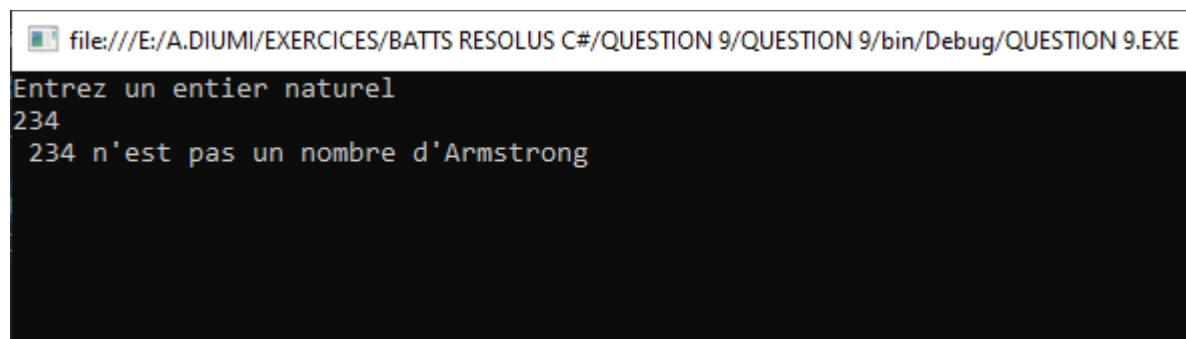
Illustrations de l'exécution



```

file:///E:/A.DIUMI/EXERCICES/BATTS RESOLUS C#/QUESTION 9/QUESTION 9/bin/Debug/QUESTION 9.EXE
Entrez un entier naturel
-23
Nombre invalide, recommencez
Entrez un entier naturel
153
153 est un nombre d'Armstrong

```



```

file:///E:/A.DIUMI/EXERCICES/BATTS RESOLUS C#/QUESTION 9/QUESTION 9/bin/Debug/QUESTION 9.EXE
Entrez un entier naturel
234
234 n'est pas un nombre d'Armstrong

```

EXERCICE 10

On souhaite écrire un programme C# de calcul de n premiers nombre parfaits. Un nombre est dit parfait s'il est égal à la somme de ses diviseurs, 1 compris.

(Support page 65)

Résolution

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

namespace EXERCICE_10
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = 0, p = 2, t = 0, sommeDiviseurs=0 ,i=0;
label1:      Console.WriteLine("Entrez le nombre de valeurs à afficher");
            n = Int32.Parse(Console.ReadLine());
            if(n<=0)
            {
                Console.WriteLine("Nombre incorrect, recommencez");
                goto label1;
            }

            //Recherche et affichage des nombres parfaits
            Console.WriteLine("Voici les " + n + " premiers nombres parfaits");
            while(t!=n)
            {
                sommeDiviseurs = 0;
                for(i=1;i< p;i++)
                {
                    if(p%i==0)
                    {
                        sommeDiviseurs += i;
                    }
                }
                if(sommeDiviseurs==p)
                {
                    Console.WriteLine(p.ToString());
                    t++;
                }
                p++;
            }
            Console.ReadLine();
        }
    }
}
```

Illustrations de l'exécution

```
file:///E:/A.DIUMI/EXERCICES/BATTS RESOLUS C#/EXERCICE 10/EXERCICE 10/bin/Debug/EXERCICE 10.EXE
Entrez le nombre de valeurs à afficher
4
Voici les 4 premiers nombres parfaits
6
28
496
8128
```

```
file:///E:/A.DIUMI/EXERCICES/BATTS RESOLUS C#/EXERCICE 10/EXERCICE 10/bin/Debug/EXERCICE 10.EXE
Entrez le nombre de valeurs à afficher
2
Voici les 2 premiers nombres parfaits
6
28
```

EXERCICE 11

On souhaite écrire un programme de calcul du pgcd de deux entiers non nuls, en C# à partir de l'algorithme de la méthode d'Euclide. Voici une spécification de l'algorithme de calcul du PGCD de deux nombres (entiers strictement positifs) a et b

(Support 65)

Résolution

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

namespace EXERCICE_11
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 0, b = 0, m = 0, n = 0, r = 10, q = 0;
            label1: Console.WriteLine("Entrez le premier nombre");
            a = Int32.Parse(Console.ReadLine());
            if(a <= 0)
            {
                Console.WriteLine("Le nombre doit etre positif, recommencez");
                goto label1;
            }

            label2: Console.WriteLine("Entrez le deuxieme nombre");
            b = Int32.Parse(Console.ReadLine());
            if (b <= 0)
            {
                Console.WriteLine("Le nombre doit etre positif, recommencez");
                goto label2;
            }
            // Recherche du PGCD
            if(a >= b)
            {
                m = a;
                n = b;
            }
            else
            {
                m = b;
                n = a;
            }
            q = m / n;
            r = m - n * q;
            while (r != 0)
            {
                m = n;
                n = r;
            }
        }
    }
}
```

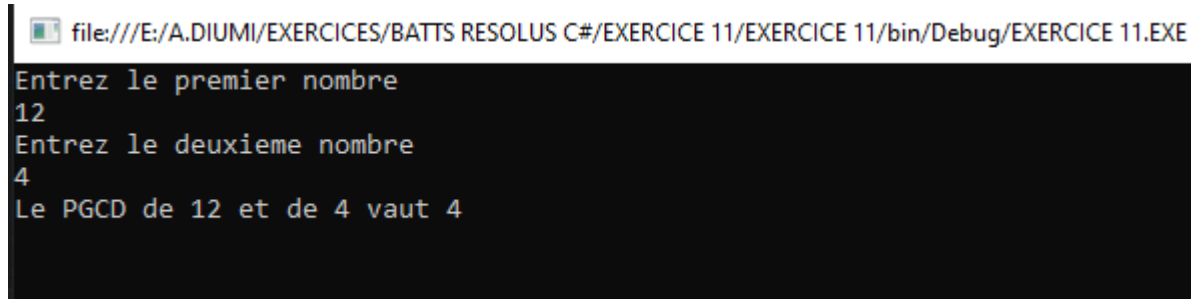


```

        q = m / n;
        r = m - n * q;
    }
    Console.WriteLine("Le PGCD de " + a + " et de " + b + " vaut " + n);
    Console.ReadLine();
}
}
}

```

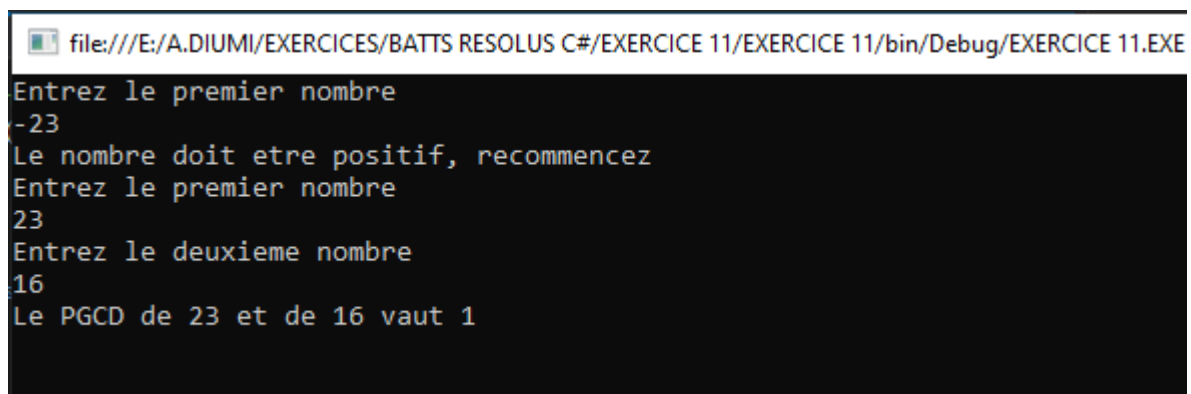
Illustrations de l'exécution



```

file:///E:/A.DIUMI/EXERCICES/BATTS RESOLUS C#/EXERCICE 11/EXERCICE 11/bin/Debug/EXERCICE 11.EXE
Entrez le premier nombre
12
Entrez le deuxieme nombre
4
Le PGCD de 12 et de 4 vaut 4

```



```

file:///E:/A.DIUMI/EXERCICES/BATTS RESOLUS C#/EXERCICE 11/EXERCICE 11/bin/Debug/EXERCICE 11.EXE
Entrez le premier nombre
-23
Le nombre doit etre positif, recommencez
Entrez le premier nombre
23
Entrez le deuxieme nombre
16
Le PGCD de 23 et de 16 vaut 1

```

*Ce recueil est en cours de rédaction,
vos suggestions et remarques nous
seront très utiles pour améliorer la
version finale.*