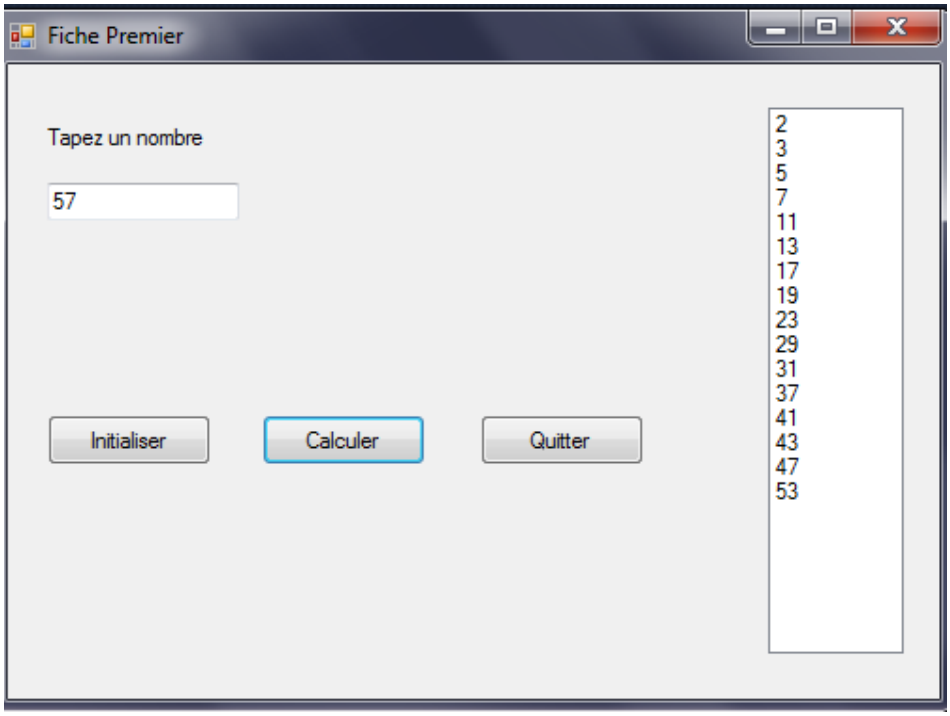
Travaux pratiques

Donnez la liste de tous les nombres premiers compris entre 1 et n, n étant lu au clavier

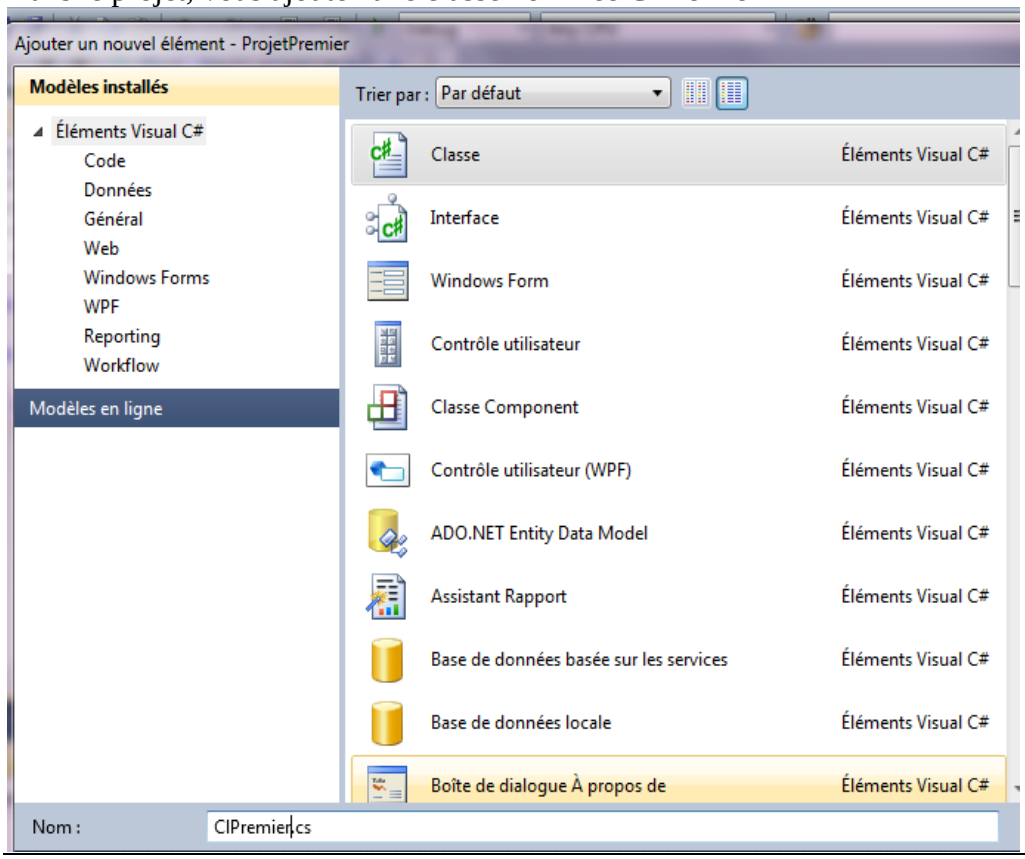


Démarche de construction :

Sur la fenêtre principale, vous ajoutez :

- 3 boutons
- une liste de type ListBox

Dans le projet, vous ajoutez une classe nommée ClPremier



Cette classe contiendra une fonction publique qui dira si le nombre passé en paramètre est premier ou non

```
using System.Linq;
using System.Text;

namespace ProjetPremier
{
    class ClPremier
    {
        public Boolean EstPremier(int n)
        {
           // code à écrire
        }
    }


}
```

Fenêtre principale

Dans le constructeur de la fenêtre, vous devez mettre :

```
// on déclare et on instancie un objet
        ClPremier unPremier = new ClPremier();
```

Sur le bouton Quitter, vous ajoutez le code suivant :

```
private void bt_quitter_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
```

Sur le bouton Initialiser, vous remettez à « zéro » les objets

```
private void bt_initialiser_Click(object sender, EventArgs e)
        {
             lb_premier.Items.Clear();
             tb_nombre.Text = "";
        }
```

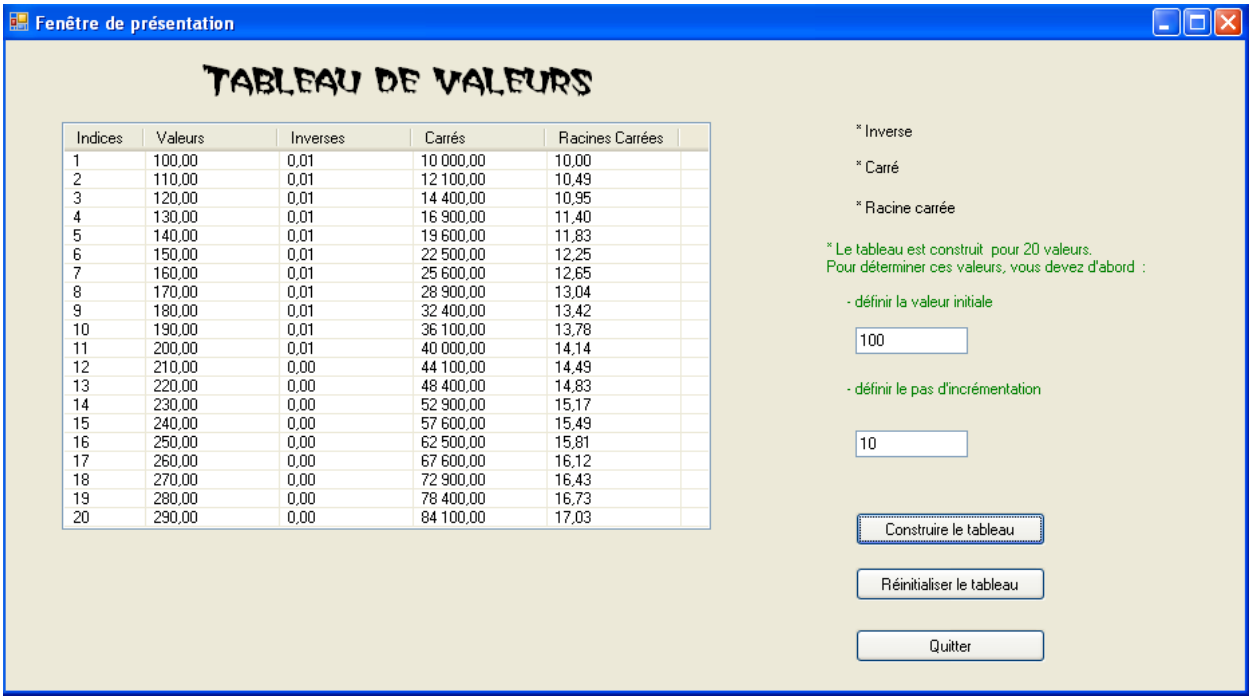
Sur le bouton Calculer, vous devez :

- récupérer le nombre saisi
- créer un objet instance de la classe ClPremierr
- lancer une boucle qui appelle la méthode publique estPremier (n) de la classe pour savoir si le nombre passé en paramètre est premier.

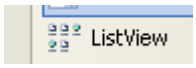
```
private void bt_calculer_Click(object sender, EventArgs e)
        {
         int  nbre=0;
         String  temp ;
        int j;
        lb_premier.Items.Clear();
        temp = tb_nombre.Text;
        if (temp == "")
            MessageBox.Show("Le nombre n'est pas saisi", "Erreur",
MessageBoxButtons.OK);
        else
            nbre = Convert.ToInt16(temp);
            for (j = 2; j< nbre;j++)
              {
                // on appelle la méthode pour tester si le nombre est premier
                if (unPremier.EstPremier(j))
                    lb_premier.Items.Add(j.ToString());
              }
        }
            }
```

Travaux pratiques

Afficher la table des inverses, des carrés et des racines carrées des 20 premiers entiers naturels



Indices	Valeurs	Inverses	Carrés	Racines Carrées
1	100,00	0,01	10 000,00	10,00
2	110,00	0,01	12 100,00	10,49
3	120,00	0,01	14 400,00	10,95
4	130,00	0,01	16 900,00	11,40
5	140,00	0,01	19 600,00	11,83
6	150,00	0,01	22 500,00	12,25
7	160,00	0,01	25 600,00	12,65
8	170,00	0,01	28 900,00	13,04
9	180,00	0,01	32 400,00	13,42
10	190,00	0,01	36 100,00	13,78
11	200,00	0,01	40 000,00	14,14
12	210,00	0,00	44 100,00	14,49
13	220,00	0,00	48 400,00	14,83
14	230,00	0,00	52 900,00	15,17
15	240,00	0,00	57 600,00	15,49
16	250,00	0,00	62 500,00	15,81
17	260,00	0,00	67 600,00	16,12
18	270,00	0,00	72 900,00	16,43
19	280,00	0,00	78 400,00	16,73
20	290,00	0,00	84 100,00	17,03

Le tableau est de type : **ListView** 

Ajout de lignes :
On passe par un objet de type : **ListViewItem**
On ajoute les valeurs dans les cellules puis on transfère la ligne au tableau.

```
ListViewItem lvi = new ListViewItem();
```

On ajoute les données de type texte dans les cellules :

```
// 1ère cellule
lvi.Text = valeur ;
// Cellule suivante Valeur
lvi.SubItems.Add(valeur1);
// Cellule suivante Valeur
lvi.SubItems.Add(valeur2);
// on tansfère la ligne au tableau
// lv_valeurs désigne le ListView
lv_valeurs.Items.Add(lvi);
```

écriture formatée

```
lvi.SubItems.Add(valeurinitiale.ToString("#,##0.00"));
```

La chaîne de formatage' #,##0.00 ' veut dire obligatoirement 2 chiffres après le séparateur décimal et un avant:
Racine carrée : utilisation de Math.Sqrt(val)

Initialisation du tableau :

```
// on efface les valeurs
lv_valeurs.Items.Clear();
llv_valeurs.Columns.Add("Indices", 60, HorizontalAlignment.Left);
```

```
lv_valeurs.Columns.Add("Valeurs", 100, HorizontalAlignment.Left);
lv_valeurs.Columns.Add("Inverses", 100, HorizontalAlignment.Left);
lv_valeurs.Columns.Add("Carrés", 100, HorizontalAlignment.Left);
lv_valeurs.Columns.Add("Racines Carrées", 100, HorizontalAlignment.Left);
lv_valeurs.ViewDetail ;
```

LE PENDU

Le travail peut être réalisé en binôme, il est déposé dans le répertoire suivant

Spécifications du jeu

- Un mot est tiré aléatoirement depuis un fichier texte. Il est affiché sous forme de tirets à l'écran.
- Un tableau de lettres permet de sélectionner une lettre. Cette lettre est affichée parmi les lettres déjà tirées et éventuellement à la place de tiret(s) dans le mot. Si la lettre est déjà tirée, un message d'alerte prévient le joueur.
- Si la lettre tirée n'est pas bonne, le graphisme augmente.
- Le jeu s'arrête sur un abandon de l'utilisateur ou sur une victoire ou un échec.

Objets à utiliser :

- tableau de lettres : `System.Windows.Forms.DataGridView;`
- graphisme : libre à vous de chercher
- mot à chercher : label
- lettres tirées : label
- choix du fichier : objet CommonDialog
- son : objet MediaPlayer de Microsoft

Choix d'un fichier pour le tirage des mots qui seront stockés dans un dictionnaire :

```
// déclaration et instanciation d'une collection pour le dictionnaire
private List<String> dictionnaire = new List<String>();

private void selectionnerFichier()
 {
   OpenFileDialog ofd = new OpenFileDialog();
  // on initalise la boîte de dialogue
   ofd.Filter = "txt files (*.txt)|*.txt";
   ofd.FileName = "";
   if (ofd.ShowDialog() == DialogResult.OK)
            {   // on ouvre le fichier
                FileStream fs = new FileStream(ofd.FileName, FileMode.Open,
FileAccess.Read);
                StreamReader sr = new StreamReader(fs);
                String ligne;
                while ((ligne = sr.ReadLine()) != null)
                    // Insertion de chaque ligne dans le dictionnaire
                    dictionnaire.Add(ligne);
                // Renseignement de la propriété Mot_secret d'un mot tiré au
                   hasard dans le dictionnaire
                fs.Close();
                sr.Close();
                Mot_secret = dictionnaire[new Random().Next(dictionnaire.Count)];
                tsplb_information.Text = "Mot secret chargé, choisissez votre première
lettre.";
                rafraichirSecret();
```

```
                bt_verifier.Enabled = true;
            }
        }
```

Abstraction de l'application

Tout le contrôle du jeu sera réalisé dans votre module :

- présence de la lettre,
- lettre déjà tirée
- …

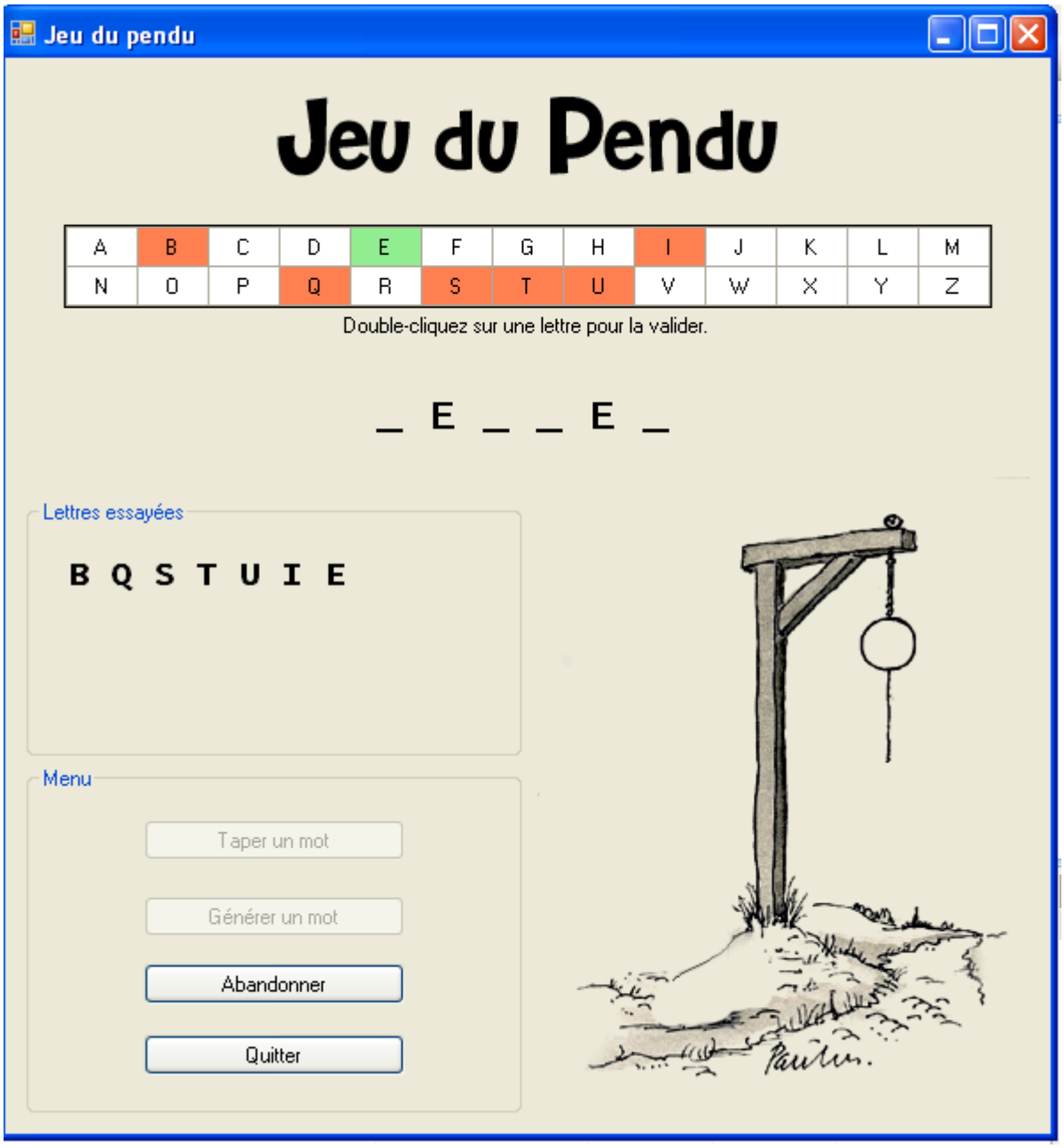
Initialisation du tableau MSF_TABLE (tableau des lettres)

Je vous donne le code à insérer dans votre fenêtre principale.

```
// Initialise les composants graphiques du formulaire
private void initComposants()
 {
   // Charge les cellules du DataGridView
   dgv_alphabet.ColumnCount = 13;
   dgv_alphabet.RowCount = 2;
   dgv_alphabet.Rows[0].Height = 30;
   dgv_alphabet.Rows[1].Height = 30;
   for (int i = 0; i < 13; i++)
    {
      dgv_alphabet.Columns[i].Width = 30;
      dgv_alphabet.Rows[0].Cells[i].Style.Alignment =
DataGridViewContentAlignment.MiddleCenter;
      // Type casting du code ascii (entier) vers sa représentation en caractère
(wrapper Char)
     // Les lettres de l'alphabet se suivent de 65 à 91 dans la table, ce qui rend
facile la manipulation
     dgv_alphabet.Rows[0].Cells[i].Value = ((Char)(i + 65)).ToString();
      // Traitement de la deuxieme partie de l'alphabet pour la seconde ligne
     dgv_alphabet.Rows[1].Cells[i].Style.Alignment =
DataGridViewContentAlignment.MiddleCenter;
    dgv_alphabet.Rows[1].Cells[i].Value = ((Char)(i + (65 + 13))).ToString();
   }
 tsplb_information.Text = "Veuillez charger un fichier de dictionnaire de mots.";
 }
```

Interface

Je vous présente une esquisse d'interface pour la réalisation de cette application.


Jeu du pendu
Jeu du Pendu
A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
Double-cliquez sur une lettre pour la valider.
_ E _ _ E _
Lettres essayées
B Q S T U I E
Menu
Taper un mot
Générer un mot
Abandonner
Quitter