


DESCRIPTION D'UNE MISSION BTS SIO			
Prénom – Nom	Quentin Coqueran	N° mission	3
Option	SLAM		
Situation		Formation	
Lieu de réalisation	Campus Montsouris 2 Rue Lacaze – 75014 PARIS		
Période de réalisation	Novembre 2019	Décembre 2019	
Modalité de réalisation	VÉCUE		

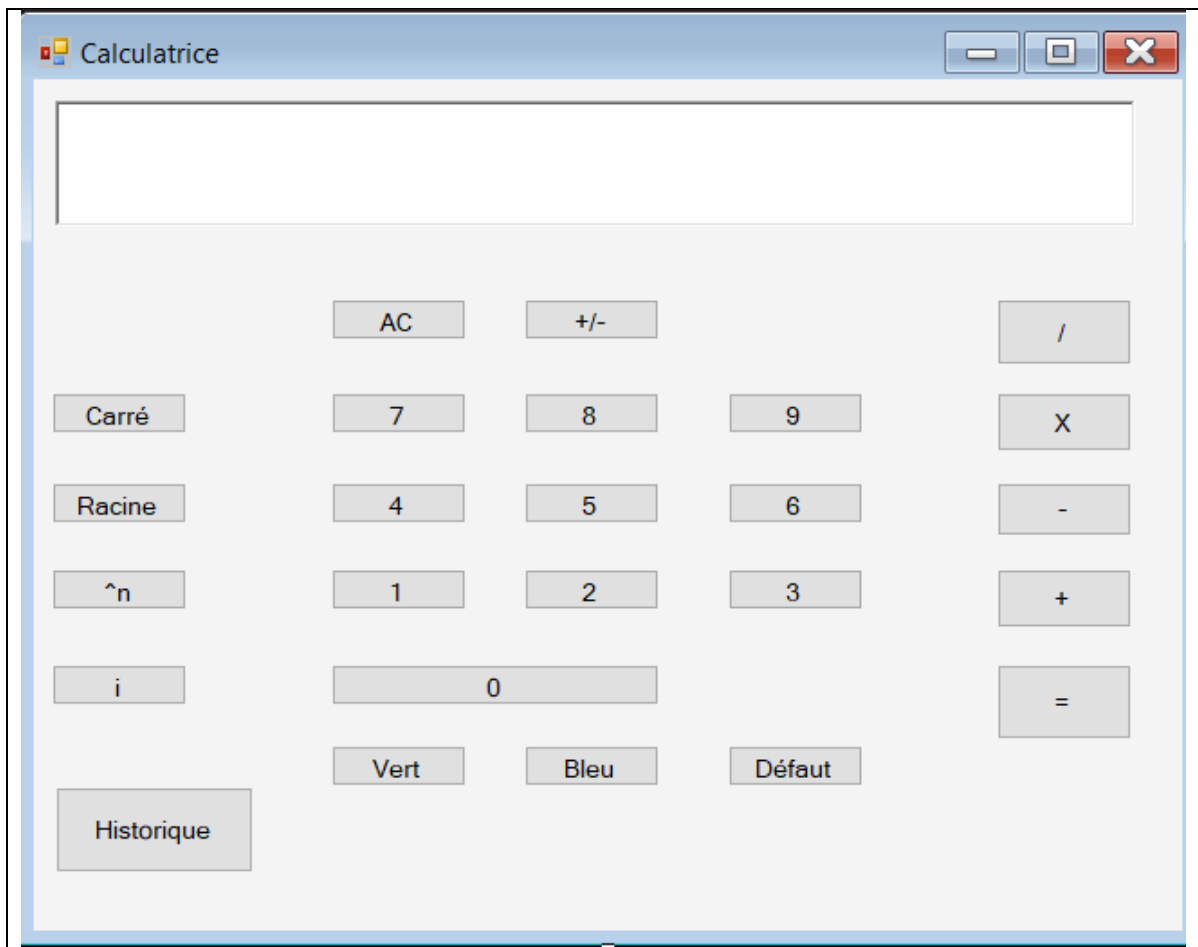
Intitulé de la mission	Titre de la mission
	Création d'une calculatrice Windows en C#
Description du contexte de la mission	Description en 2 à 3 lignes maxi
	Créer une calculatrice WindowsForm en C#

Ressources et Outils utilisés	Liste des ressources disponibles et outils utilisés (Documentations, Matériels et Logiciels)
	<ul style="list-style-type: none"> - Visual Studio 2019 - Cahier des charges : <p>Vous trouverez ici la référence de quelques fonctions scientifiques</p> <p>https://docs.microsoft.com/en-us/dotnet/api/system.math.abs?view=netcore-3.1</p> <p>Créer une application C# graphique reproduisant une calculatrice basique (quatre opérations) et des opérations suivantes :</p> <p>Carré, Racine Carrée, Puissance</p> <p>Vous agrémenterez votre calculatrice de trois boutons (cachés par défaut et qui n'apparaissent et disparaissent qu'après un clic sur le bouton 'i')</p> <ul style="list-style-type: none"> • Le clic sur le bouton "Vert" change la couleur du formulaire vers le vert, désactive le bouton "Vert" et active le bouton "Bleu" et le bouton "Couleur par défaut". • Le clic sur le bouton "Bleu" change la couleur du formulaire vers le bleu, active le bouton "Vert" et le bouton "Couleur par défaut" et désactive le bouton "Bleu". • Par défaut, le bouton "Couleur par défaut" doit être désactivé. <p>Vous trouverez ici un petit visuel du fonctionnement des contrôles :</p> <p>https://www.screencast.com/t/BU3b7imkGTF</p>
Résultat attendu	Résultat attendu avec la réalisation de cette mission
	L'utilisateur doit pouvoir faire différents calculs à l'aide de sa souris

Contraintes	Contraintes : techniques budgétaires temps O.S. ou outils imposés...
	<ul style="list-style-type: none">- L'utilisateur doit utiliser la souris- Gérer les exceptions ainsi que les erreurs

Compétences associées <i>(voir tableau) à faire</i>	Liste des intitulés du tableau de compétences (avec les références)
	<p>A1.1.3 : Étude des exigences liées à la qualité attendue d'un service</p> <p>A1.3.3 : Accompagnement de la mise en place d'un nouveau service</p> <p>A2.1.1 : Accompagnement des utilisateurs dans la prise en main d'un service</p> <p>A2.1.2 : Évaluation et maintien de la qualité d'un service</p> <p>A2.1.2 : Évaluation et maintien de la qualité d'un service</p> <p>A5.1.1 : Mise en place d'une gestion de configuration</p> <p>Réalisation des tests nécessaires à la validation d'éléments adaptés ou développés</p>

Description simplifiée des différentes étapes de réalisation de la mission en mettant en évidence la démarche suivie, les méthodes et les techniques utilisées
<p><u>Chapitre 1 : Création de l'interface graphique</u></p> <p><u>Etape :</u></p> <p>1.1/ Tout d'abord nous créons un projet WindowsForm.</p> <p>1.2/ Ensuite on crée l'interface graphique qui sera afficher en tant que calculatrice.</p>



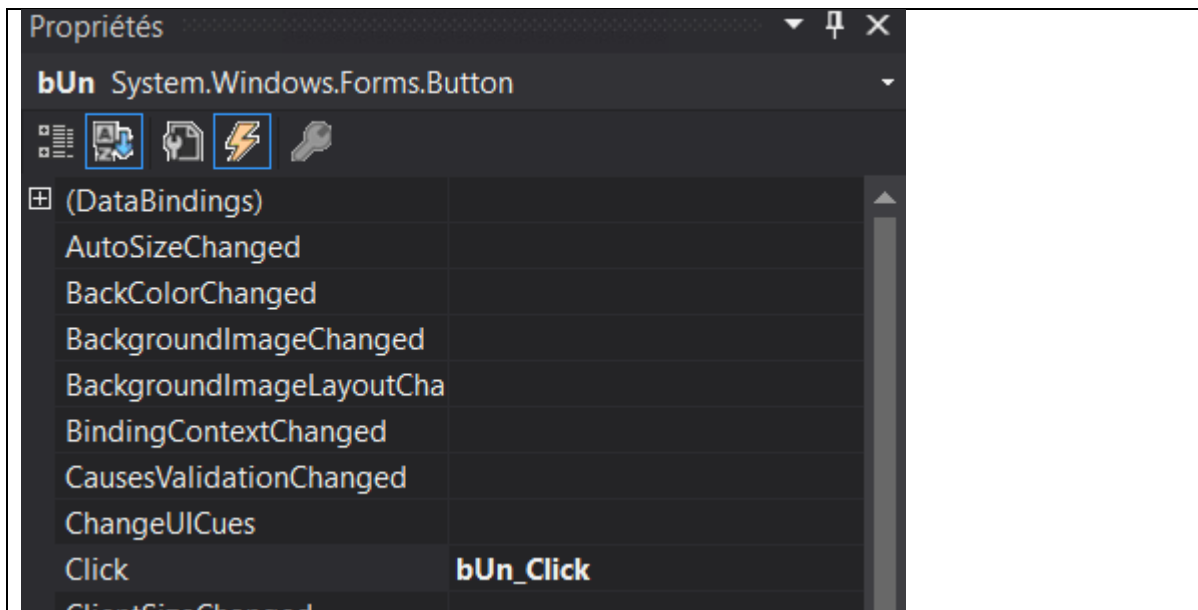
Résultat :

L'interface graphique de la calculatrice est prête.

Chapitre 2 : Affecter le clic de l'utilisateur à un évènement

Etape :

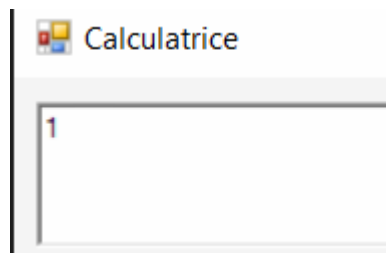
2.1/ Nous allons affecter le clic de l'utilisateur sur l'un des boutons grâce à l'évènement "Click".



2.2/ Lorsque le joueur appuie sur le bouton '1', la méthode intitulé "bUn_Click" est exécuter.

```
private void bUn_Click(object sender, EventArgs e)
{
    rtb1.Text += 1;
}
```

Cette méthode affiche le chiffre 1 dans la barre de calcul



Résultat :

Chaque bouton effectue une action

Chapitre 3 : Effectuer le calcul

Etape :

3.1/ Lorsque l'on appuie sur un opérateur de calcul la calculatrice va stocker la première valeur, puis effacer le contenu pour laisser l'utilisateur écrire son 2e chiffre.

```
private void bAddition_Click(object sender, EventArgs e)
{
    try
    {
        premierNumero = double.Parse(rtb1.Text);
        rtb1.Text = "";
        operateur = "addition";
    }
    catch
    {
        rtb1.Text = "Veuillez insérer des chiffres avant";
    }
}
```

Calculatrice

	AC	+/-		/
Carré	7	8	9	x
Racine	4	5	6	-
^n	1	2	3	+

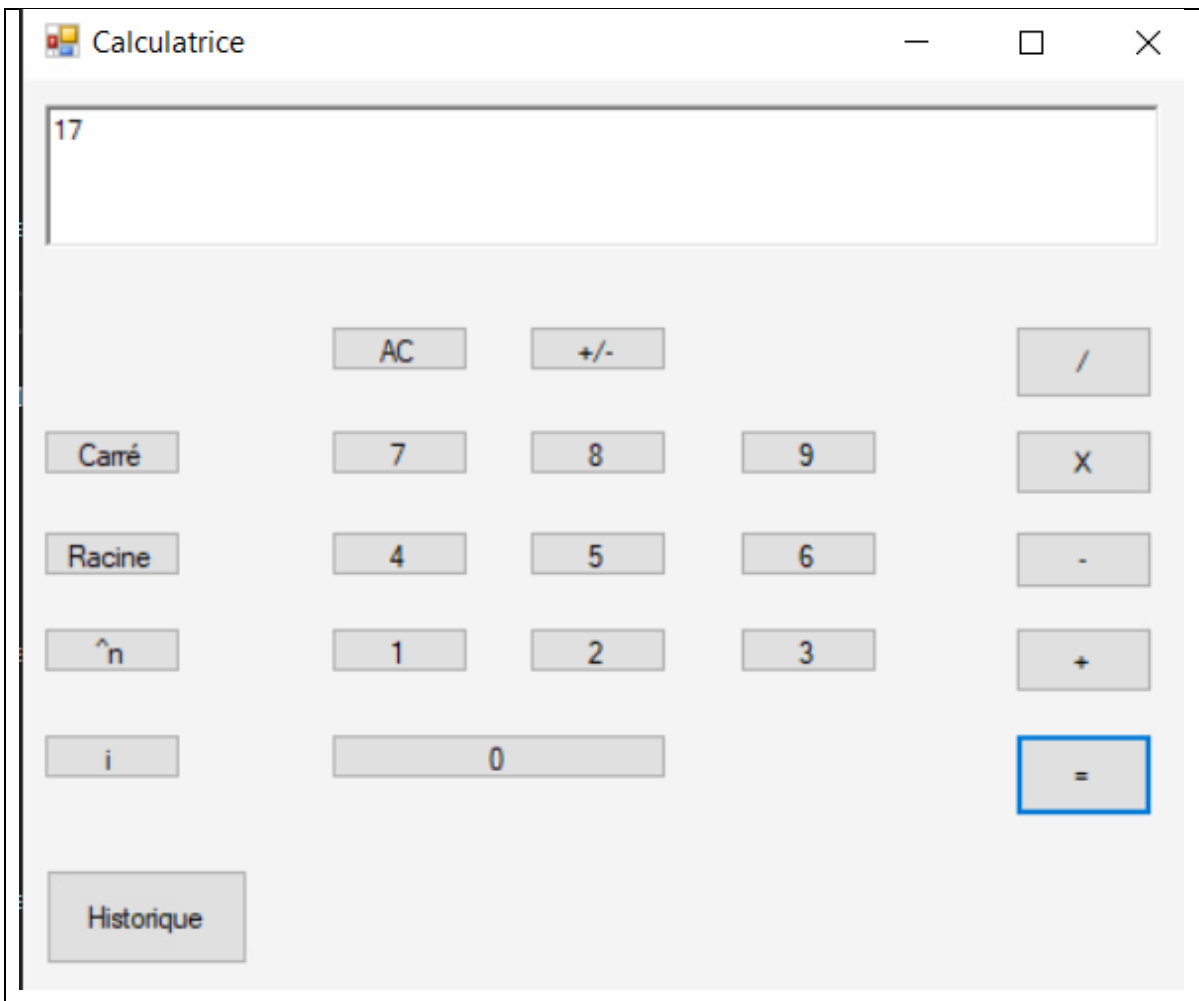
Résultat :

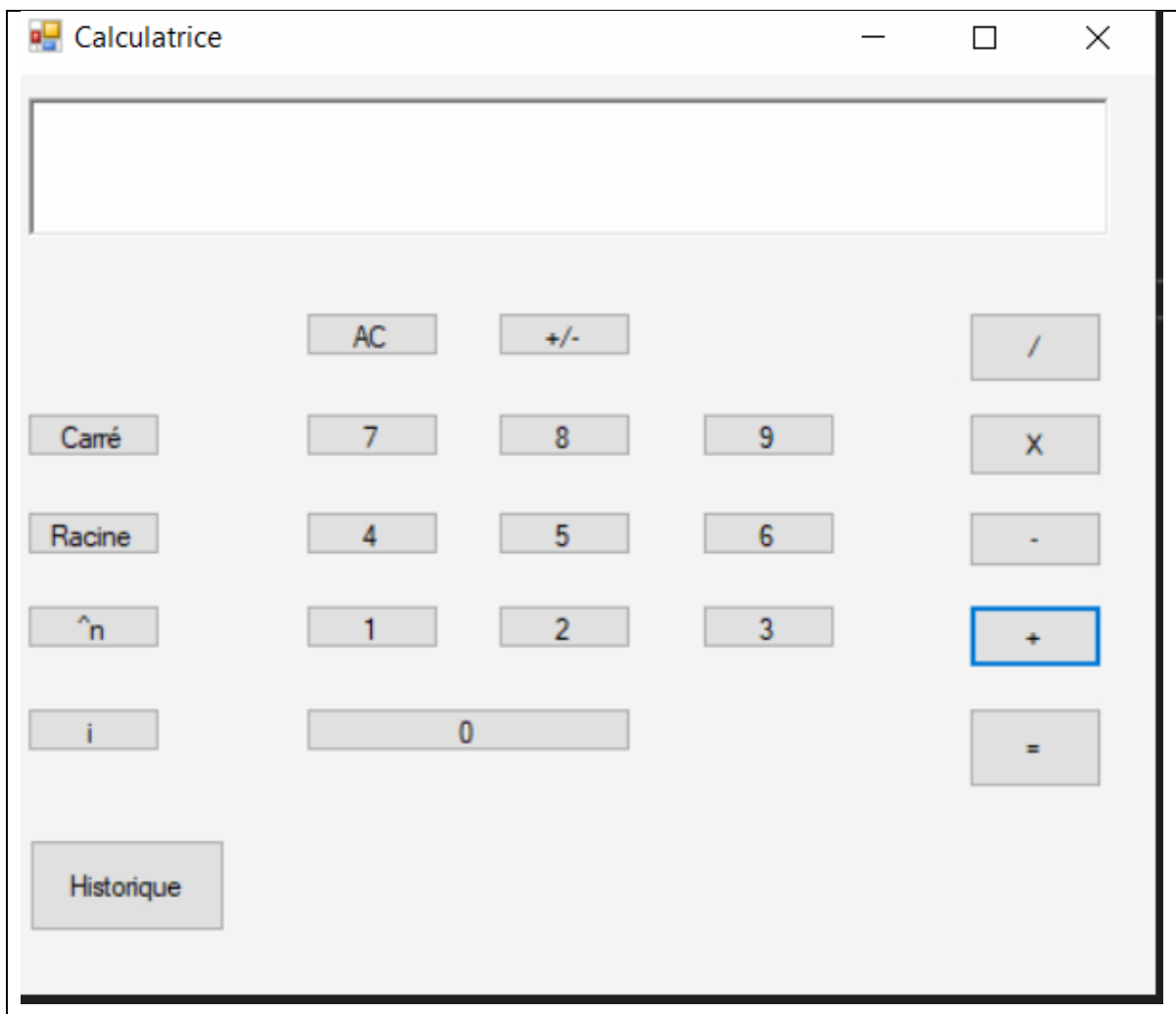
L'utilisateur peut choisir un 2e chiffre pour son calcul

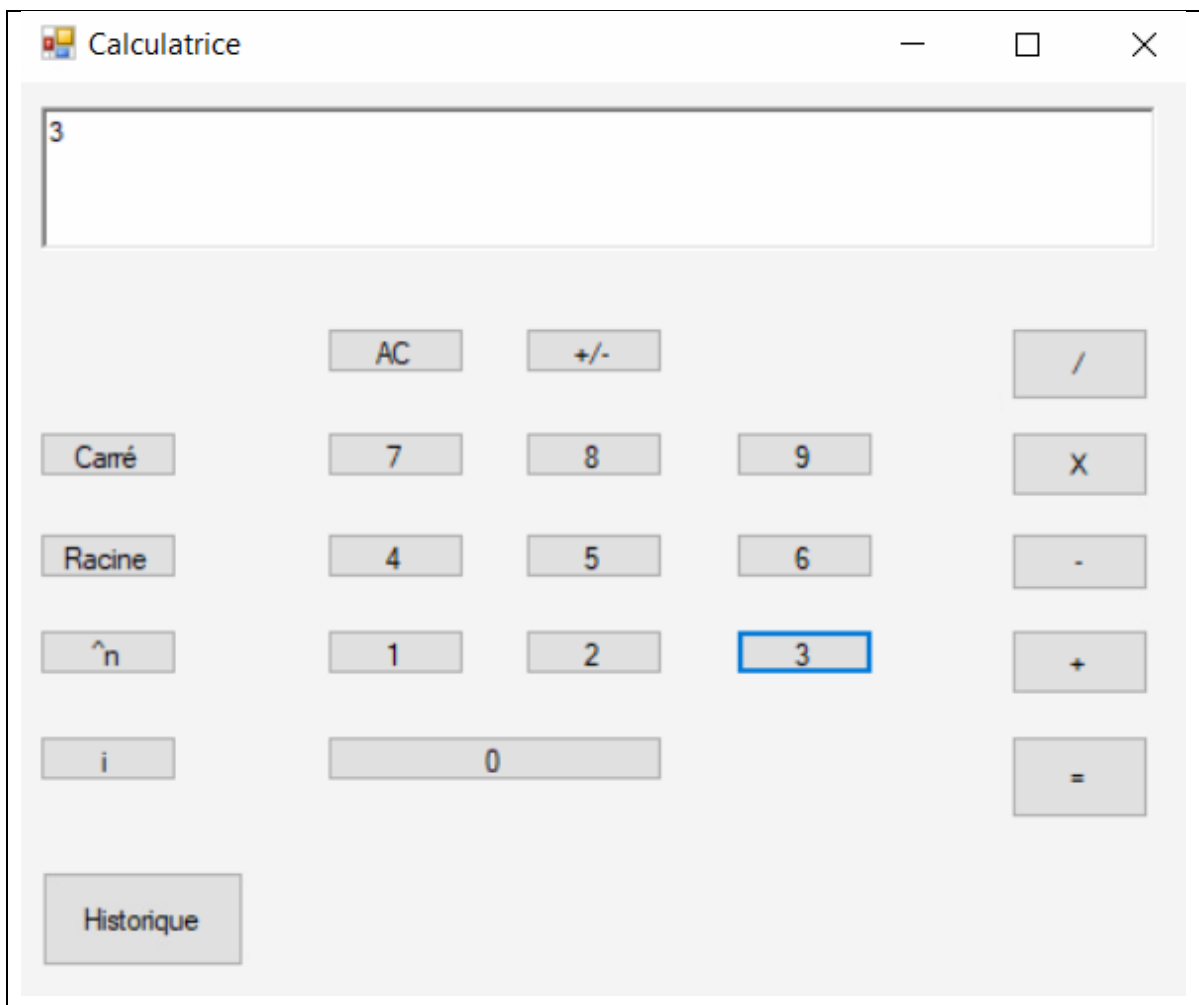
Chapitre 4 : Calculer

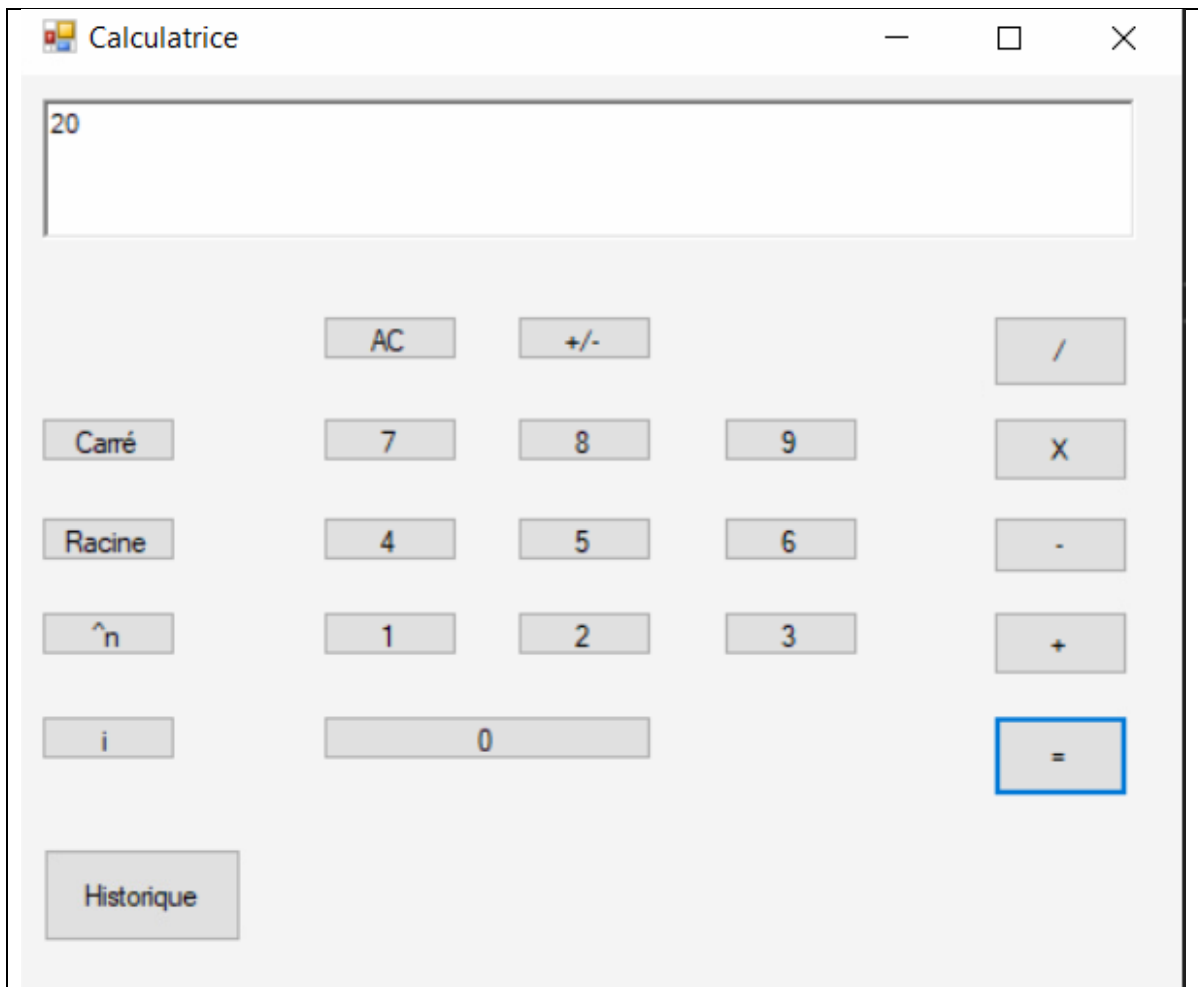
Etape :

4.1/ Grâce au bouton '=' on pourra calculer ce que l'utilisateur aura saisi









```
private void bEgale_Click(object sender, EventArgs e)
{
    try
    {
        dernierNumero = double.Parse(rtb1.Text);
        switch (opérateur)
        {
            case "addition":
                resultat = premierNumero + dernierNumero;
                rtb1.Text = resultat.ToString();
                rtbHistorique.Text += premierNumero.ToString() + "+" + dernierNumero.ToString() + "=" + resultat.ToString() + '\n';
                break;
            case "soustraction":
                resultat = premierNumero - dernierNumero;
                rtb1.Text = resultat.ToString();
                rtbHistorique.Text += premierNumero.ToString() + "-" + dernierNumero.ToString() + "=" + resultat.ToString() + '\n';
                break;
            case "multiplication":
                resultat = premierNumero * dernierNumero;
                rtb1.Text = resultat.ToString();
                rtbHistorique.Text += premierNumero.ToString() + "*" + dernierNumero.ToString() + "=" + resultat.ToString() + '\n';
                break;
        }
    }
}
```

Résultat : L'utilisateur peut faire un calcul

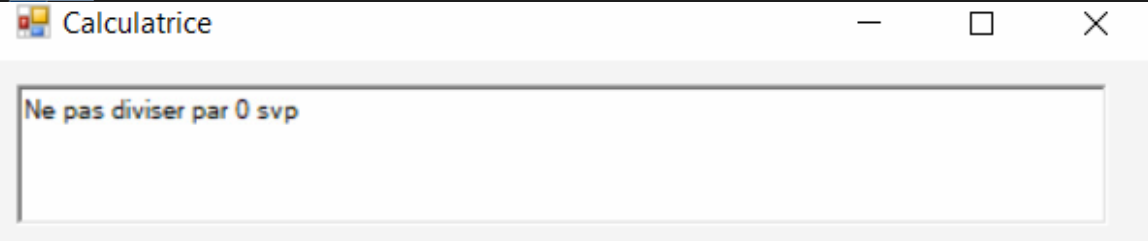
Chapitre 5 : Gérer les exceptions

Une division par 0 est impossible alors il faut gérer le cas où l'utilisateur voudrait faire une division par 0. Ou encore si l'utilisateur appuie sur '+' mais n'a pas saisi de chiffres avant

```

case "division":
    if (dernierNumero == 0)
        rtb1.Text = "Ne pas diviser par 0 svp";
    else
    {
        resultat = premierNumero / dernierNumero;
        rtb1.Text = resultat.ToString();
        rtbHistorique.Text += premierNumero.ToString() + "/" + dernierNumero.ToString() + " = " + resultat.ToString() + '\n';
    }
    break;

```



```

catch
{
    rtb1.Text = "Veuillez insérer des chiffres avant";
}

```

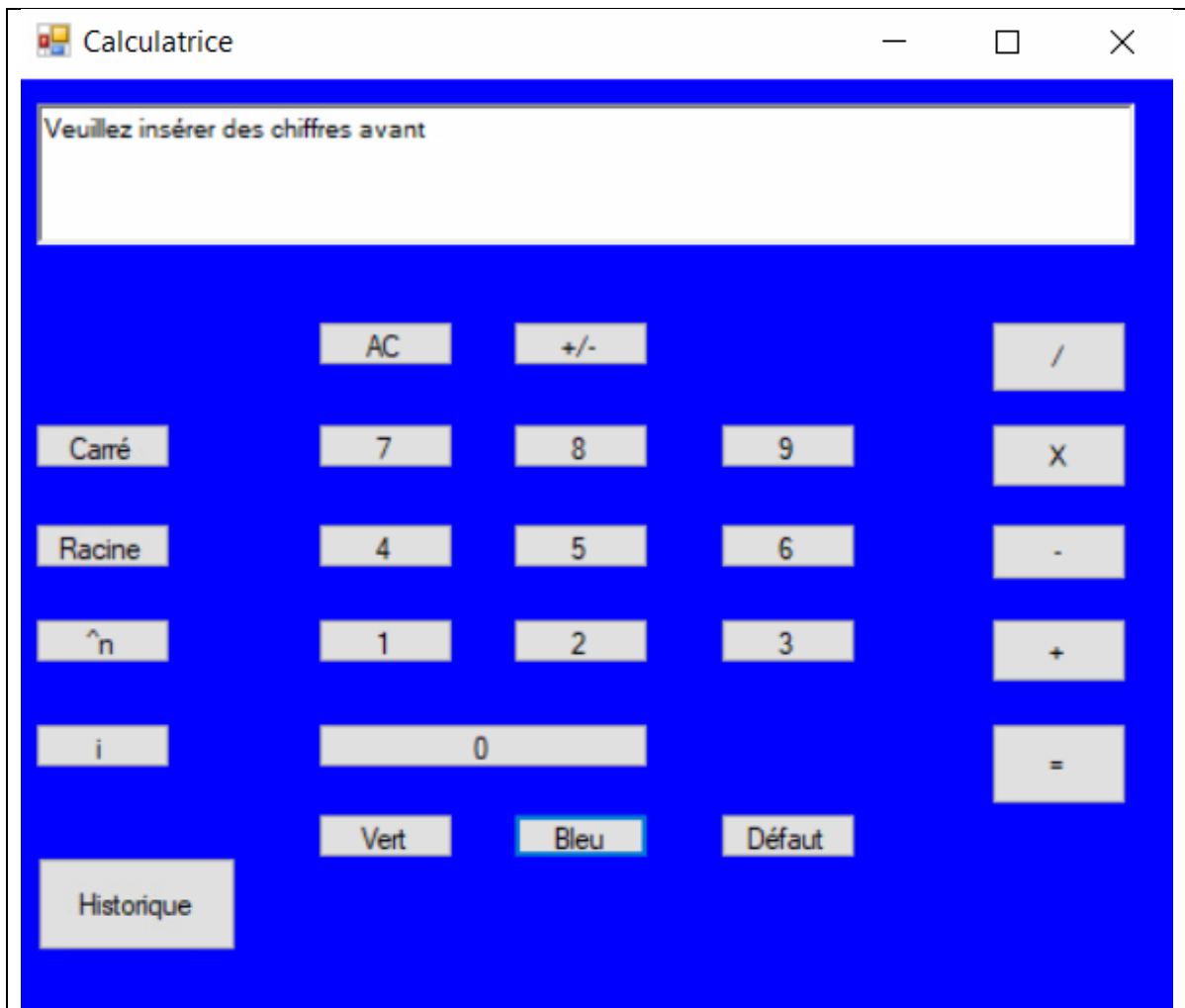


Résultat : L'utilisateur ne peut pas faire dysfonctionner la calculatrice

Chapitre 6 : Modifier l'apparence de la calculatrice

Grace au bouton i des boutons pour changer la couleur de la calculatrice apparaissent en 3 couleurs différentes (Vert, Bleu et par défaut)

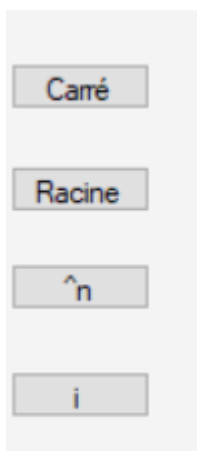




Résultat : L'utilisateur peut modifier la couleur de la calculatrice

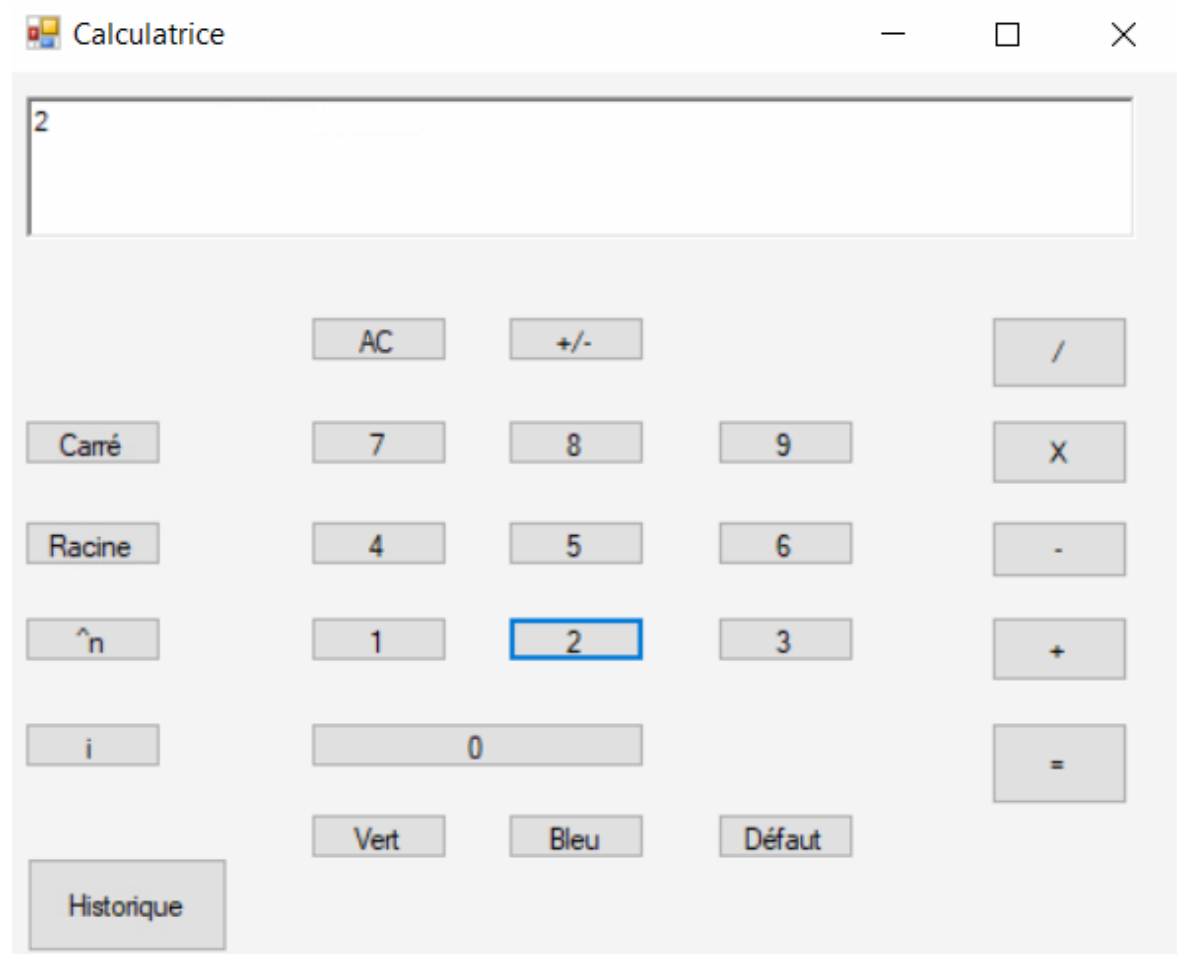
Chapitre 7 : Effectuer d'autres calculs plus complexes

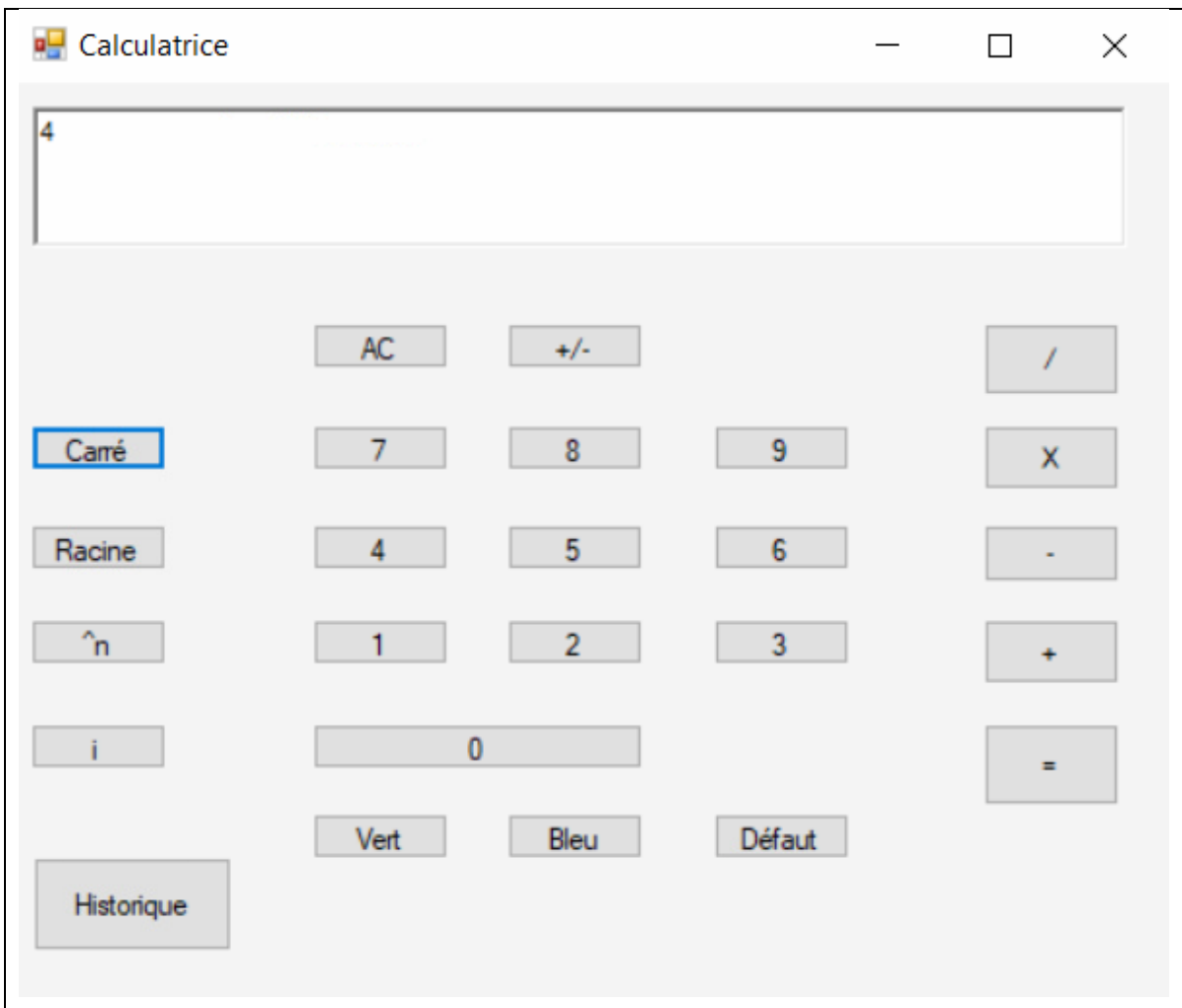
L'utilisateur peut, en plus des opérations classiques, effectuer un calcul au carré, à la racine et la puissance

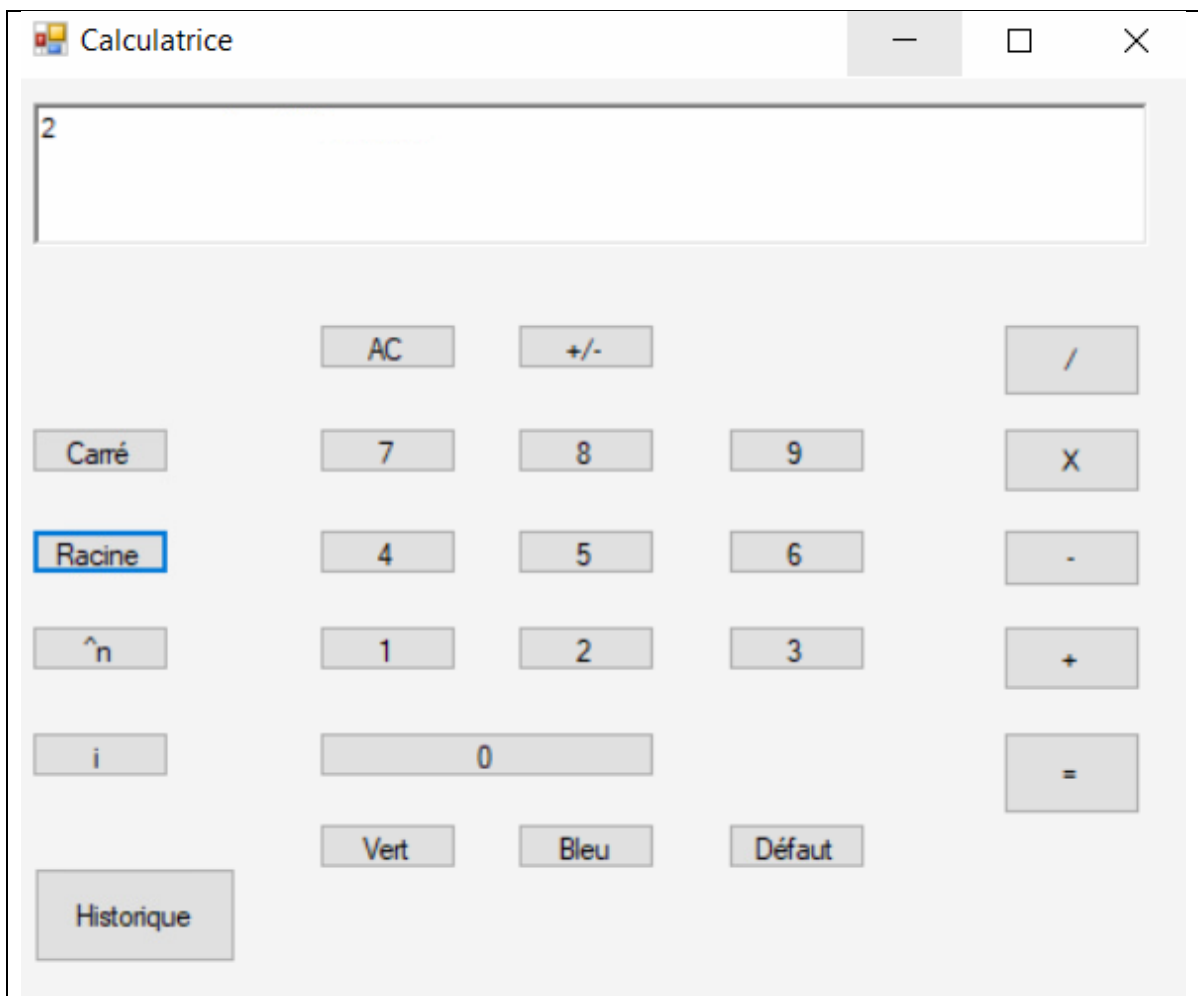


Le fonctionnement du bouton “^n” est le même qu'un opérateur classique

“Carré” et “Racine” ont le même fonctionnement, lorsque l'utilisateur appuie sur le bouton le chiffre est automatiquement mis au carré ou à la racine.





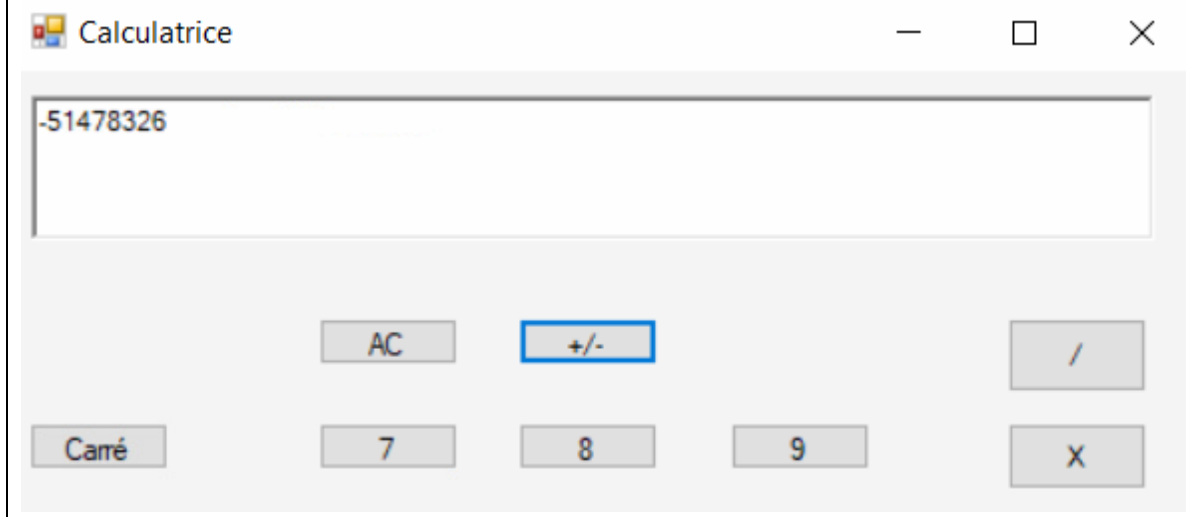


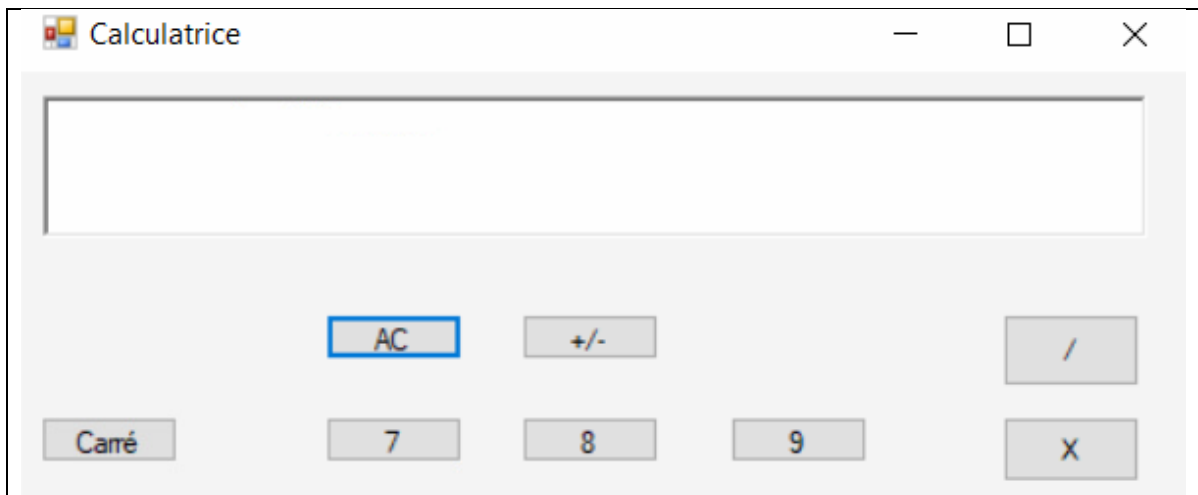
Résultat : L'utilisateur peut faire des calculs plus complexes

Chapitre 8 : Insérer d'autres fonctionnalités pour aider l'utilisateur

D'autres boutons sont utilisables comme mettre un nombre au négatif ou au positif.

Le bouton AC permet d'effacer le contenu présent dans la barre de calculs.

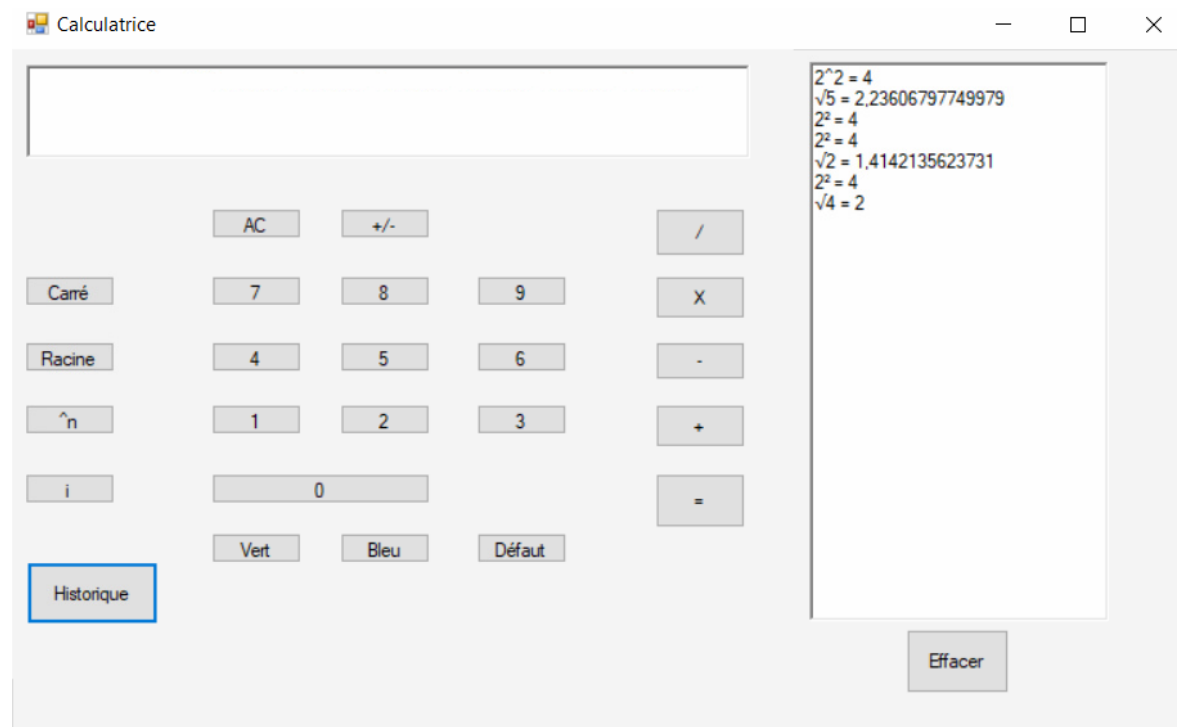




Résultat : L'interface est plus intuitive et ergonomique

Chapitre 9 : Insérer un historique de calculs

Un historique de calcul est disponible grâce au bouton "Historique"



Un bouton "Effacer" est également disponible pour effacer l'historique

Résultat : L'utilisateur possède un historique des différents calculs qu'il a effectué

--

Conclusion	Que pouvez-vous dire de cette mission : apport personnel, expérience, etc
	Meilleure compréhension des applications Windows Meilleure compréhension de l'avancement d'un projet Amélioration dans le langage C#.

Productions associées	Liste des documents produits et description