S4 – Développement logiciel

Premiers pas avec C# et Visual Studio

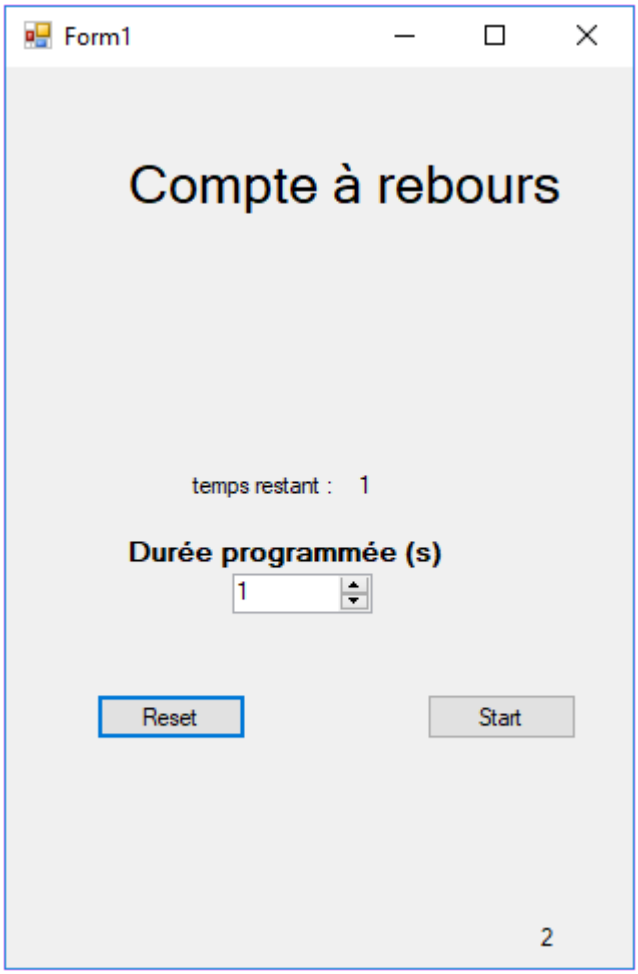
Gestion des événements et du temps

Objectifs : être capable de coder une application simple sous Windows en utilisant C# et Visual Studio à partir d'un cahier des charges, d'un diagramme d'état et d'un « squelette » de code donnés.

Présentation de l'application à coder

Le but de ce TP est de réaliser une application compte à rebours (un minuteur) en utilisant Visual Studio et C#.

L'allure de l'interface de l'application est la suivante.



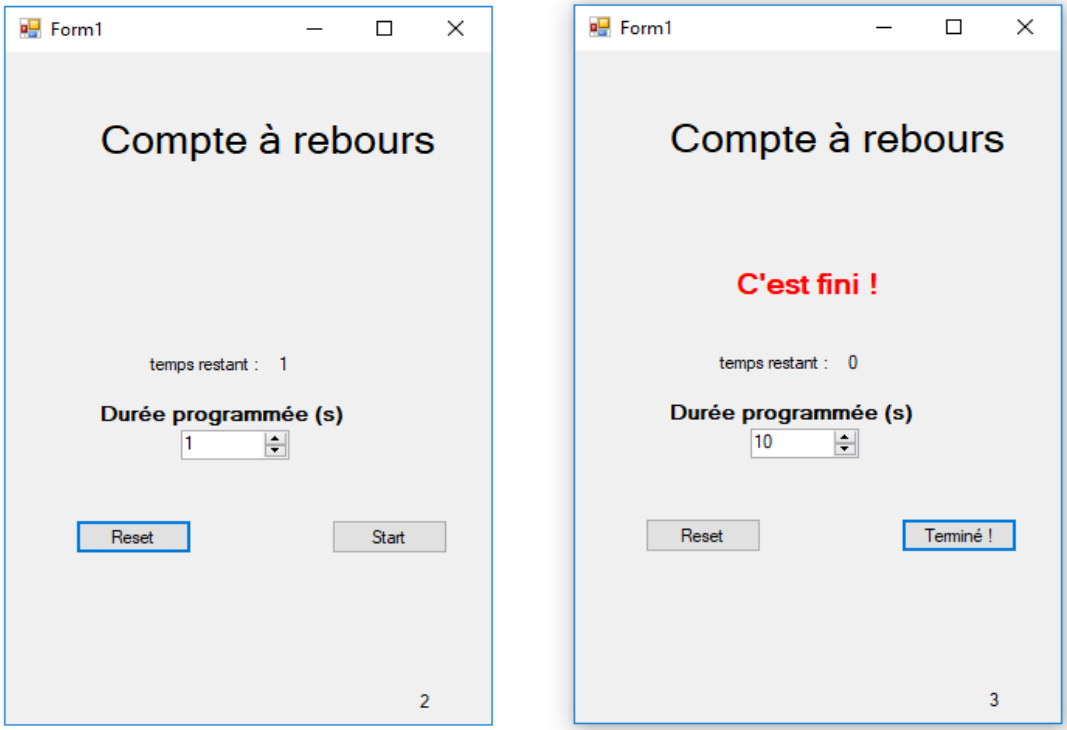
Au démarrage, en jouant sur les flèches, l'utilisateur peut régler la durée du décompte en seconde (par exemple 180 s pour la cuisson d'un œuf à la coque en 3 minutes). Dans cette phase, la valeur affichée « temps restant » suit la valeur de « durée programmée ».

Une fois la durée programmée, dès que l'utilisateur appuie sur le bouton Start (bouton de droite), la valeur affichée « temps restant » décrémente chaque seconde. L'affichage du bouton de droite « Start » devient « Stop ».

Pendant cette phase, à tout moment l'utilisateur peut mettre en pause le décomptage en appuyant sur Stop. Après l'appui, dans la phase de pause, l'affichage du bouton de droite « Stop » devient « Start ».

En appuyant sur Start, le décompte reprends à la valeur où il s'était arrêté, le bouton de droite affiche maintenant « Stop ».

A la fin du décompte, le message « C'est fini ! » s'affiche en rouge au milieu de la fenêtre, le compte à rebours « temps restant » affiche 0, le bouton de droite affiche « Terminé ! » et l'appui sur ce bouton n'a plus aucun effet. Voir figure ci dessous.

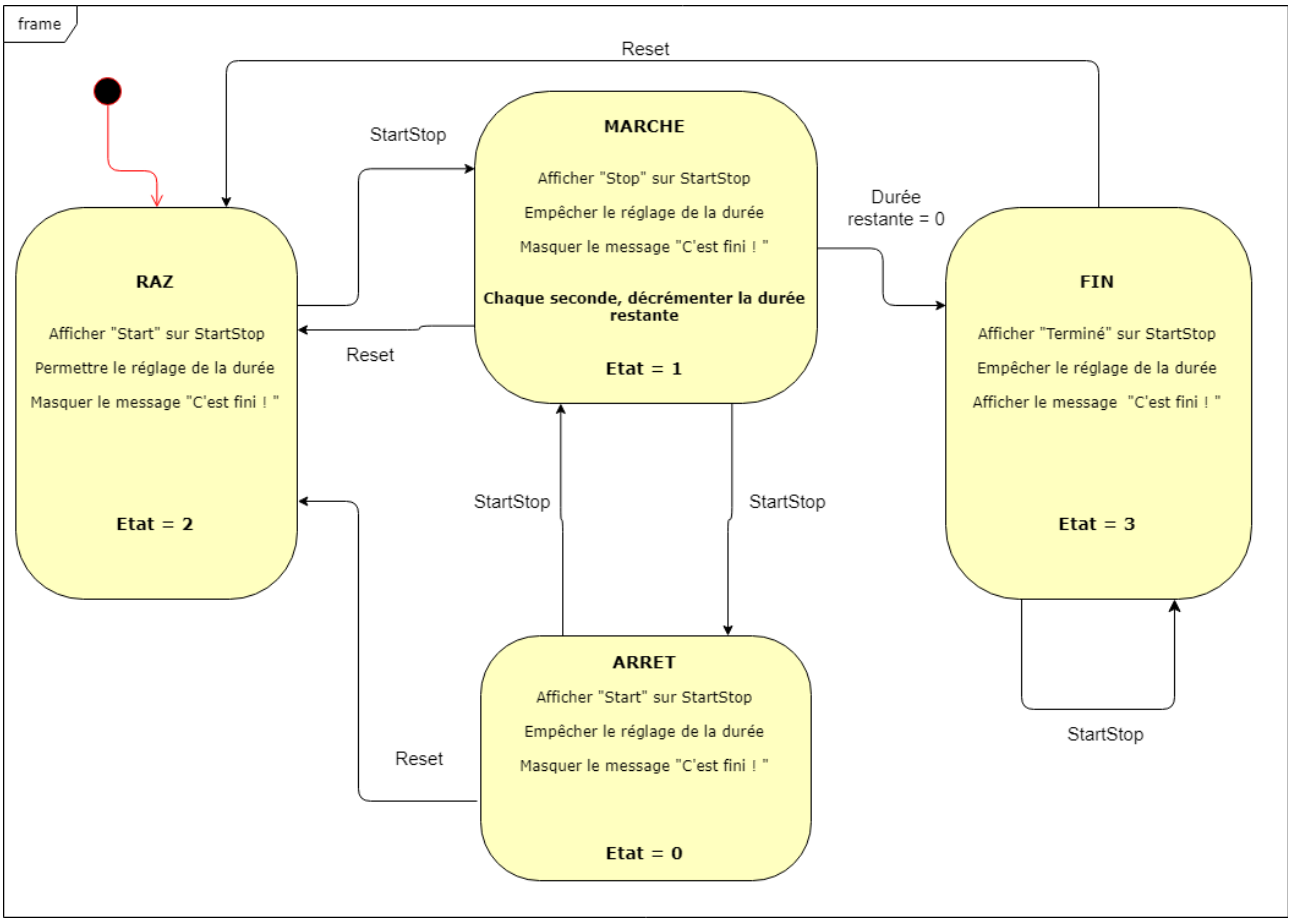


L'utilisateur n'a pas d'autre choix que d'appuyer sur le bouton Reset pour recommencer un cycle.

Remarque : dans n'importe quelle phase, à tout moment, un appui sur Reset arrête le décompte et réinitialise le valeur temps restant à la durée programmée.

Représentation machine à états

On a représenté le fonctionnement de l'application par le diagramme d'états suivant :



Remarque : ce diagramme a été dessiné avec l'outil en ligne https://draw.io

On a défini ici une variable **Etat** (variable d'état du système) qui peut prendre 4 valeurs (entre 0 et 3). La valeur de cette variable d'état est affichée pour débogage en bas à gauche de l'écran principal de l'application.

Questions préliminaires

Q.1) Quel est l'état du système au démarrage ? Justifiez.

Q.2) Comparez les différences entre l'appui sur Reset et l'appui sur StartStop.

Q.3) Que se passe-t-il si durée restante =0 alors que le système se trouve dans l'état 2 ?

Q.4) Compléter le tableau suivant.

On utilisera les méthodes nommées Raz(), Marche(), Arret(), Fin() qui permettent de passer dans chacun des états :

Exemple : la méthode Raz() permet au système d'évoluer dans l'état noté RAZ (etat =2) sur le diagramme d'état.

Les actions effectuées dans cette méthode Raz() sont celles définies dans le diagramme d'état.

événement	**Condition**	**Méthode ou Action**
Démarrage système	X	Raz()
Appui sur Reset	X	Raz()
Appui sur StartStop	Etat = 0	
	Etat = 1	
	Etat = 2	
	Etat = 3	
Chaque seconde	Etat = 1	
Changement saisie durée	X	Modifier tRestant

Codage de l’application

Travaux pratiques

Créer l’interface utilisateur et configurer les contrôles

Ouvrir Visual Studio, créer un nouveau projet. Se placer en mode concepteur et y placer les contrôles suivants (il est conseillé de nommer les contrôles conformément à la liste suivante) :

numericUpDown : saisieDuree

boutons : bouton de gauche : boutonReset ; bouton de droite : boutonStartStop

label : labelTrestant (affichage du temps restant), labelFin (message « C’est fini »), labelDebug (affichage de la variable état).

Timer : timer1 : c’est ce composant qui réalisera le cadencement du minuteur/compte à rebours. Il sera réglé à une période de 1s.

Configurer chaque contrôle dans le respect du cahier des charges.

Vous aurez aussi besoin de créer d’autres labels « fixes » mais leur nom importe peu car ils n’interagissent pas avec le programme.

Coder

Un « squelette » de l’application à coder est donné en annexe.

Dans Visual studio, en mode code, pour coder l’application :

- vous créerez deux variables locales entières nommées etat (la variable d’état de la machine à états) et tRestant (qui représente le temps restant à décompter).

- vous créerez quatre méthodes nommées RAZ(), Marche(), Arret(), Fin(). Ces méthodes correspondent aux actions effectuées dans l’état correspondant, conformément au diagramme d’état.

- vous coderez ensuite les actions correspondant à chaque événement, conformément au diagramme d’état et au tableau évenements/conditions/méthodes

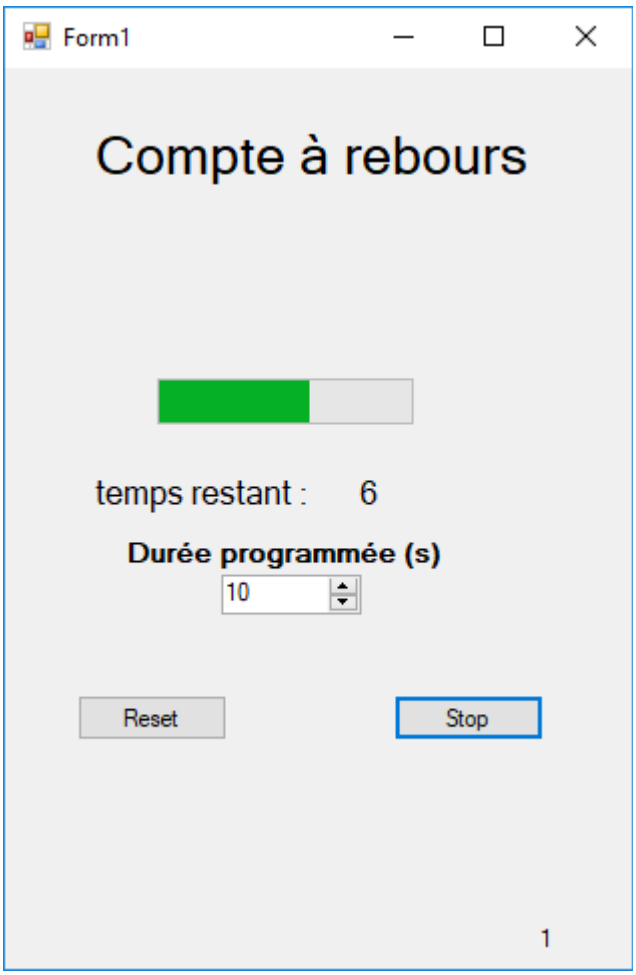
Conseil n°1 : pour que la machine à état démarre dans l’état initial désiré, il faut définir les actions dans la méthode Form1, juste après l’instruction InitializeComponent() ;

Conseil n° 2 : la structure « switch case » est particulièrement efficace pour coder une machine à états. Celle-ci pourra être implémentée dans la méthode boutonStartStop_Click().

Les bonus

Pour ceux qui ont terminé en avance, voici quelques pistes d'améliorations en questions bonus.

Utiliser une barre de progression diminuant progressivement durant le décompte, conformément à l'image suivante.



Implémenter un beep en fin de décomptage ou un fichier wav.

Effectuer les finitions d'usage : icônes, etc.

Annexe : « squelette » de l'application à coder

```
namespace chronoA
{
    public partial class Form1 : Form
    {
        int etat ;
        int tRestant;

        public Form1()
        {
            InitializeComponent();
           // à compléter
        }

        public void RAZ()
        { // les actions que l'on fait en entrant dans l'etat
            // à compléter
              etat = 2 ;
        }

        public void Marche()
        {
            // à compléter
        }

        public void Arret()
        {
           // à compléter
        }

        public void Fin()
        {
            // à compléter
        }

        private void boutonReset_Click(object sender, EventArgs e)
        {
            // à compléter
        }

        private void boutonStartStop_Click(object sender, EventArgs e)
        {
           // à compléter
        }

        private void saisieDuree_ValueChanged(object sender, EventArgs e)
        {
           // à compléter
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            // à compléter
         }
       }

    }
}
```