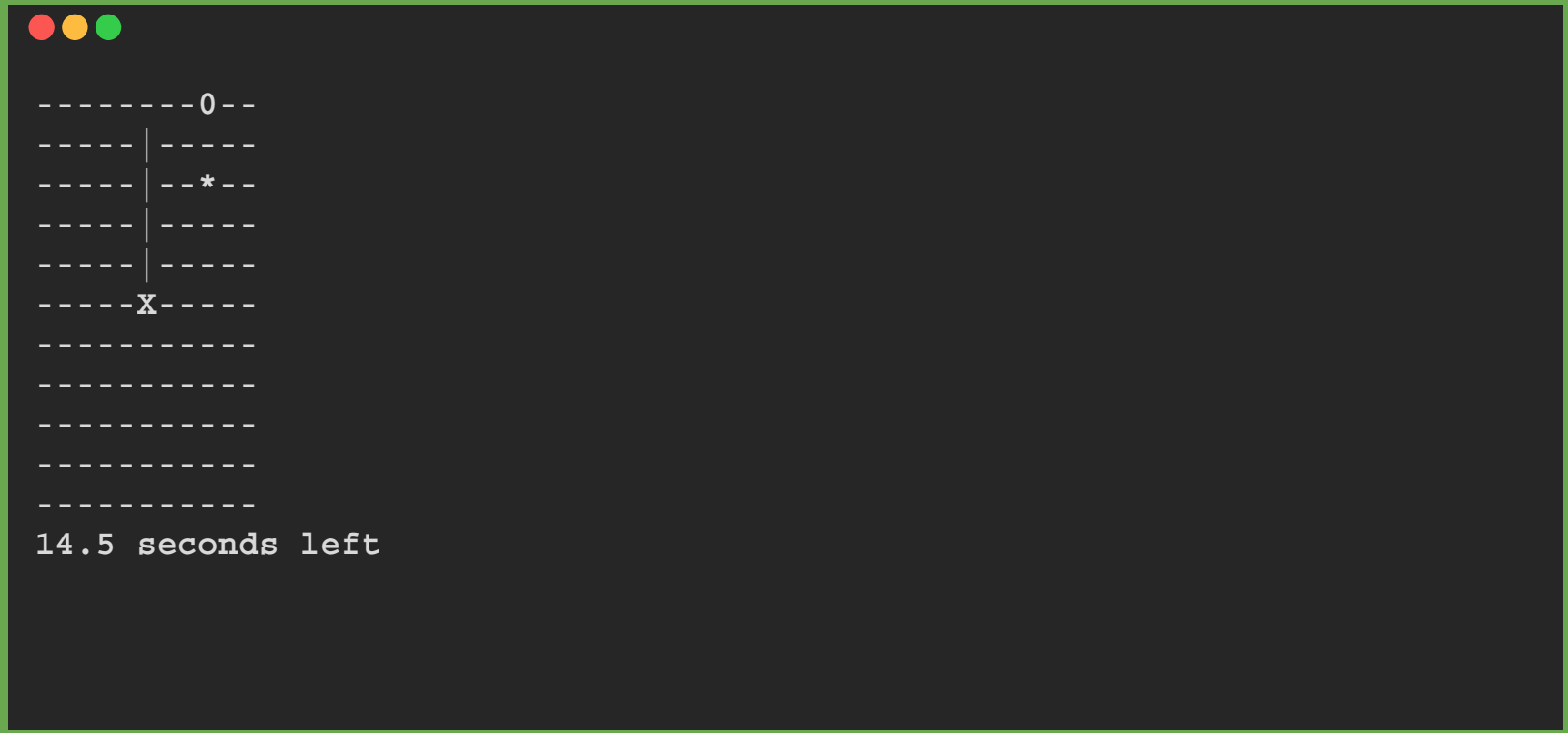
PROJECT ASCII TIMER \[0-0]/

[o.o]/ OBJECTIF

Réaliser une animation ASCII temps réel en C++ via la console

[O.o] EXEMPLE D'APPLICATION



[?_?] FONCTIONS A IMPLEMENTER

`AskDuration()`

```
Saisis la durée du chrono (en secondes) :
> 1
```

- Objectifs :
  - Affiche le message de demande.
  - Vérifie que l'utilisateur a bien saisi un nombre.
  - Redemande à nouveau si ça n'est pas le cas.
- Paramètre : aucun
- Valeur de retour : `int`

[._.] FONCTIONS A IMPLEMENTER

`GetTime()`

- Objectif : Donne le temps en secondes depuis le lancement de l'application
- Paramètres : Aucun
- Valeur de retour : temps en secondes (float)

Utiliser le code suivant pour calculer le temps en secondes depuis le lancement de l'application :

```
float timeInSeconds = clock() / (float)CLOCKS_PER_SEC;
// nécessite #include <time.h>
```

[o.o] BOUCLE TEMPS REEL

Dans la fonction “main” écrire une boucle “while” s’arrêtant au bout de 5s.

Pour réaliser cet algorithme :

- Déclarer une variable “currentTime” et l’initialiser avec “obtenirTemps()”,
- Implémenter une boucle “while” qui s’arrête lorsque “GetTime() - currentTime” est supérieur à 5.

En fin de boucle, afficher un message de fin :

```
Fin du chrono ! \[^_^]/
```

[^_^] CHRONO

Ecrire une structure “Chrono” avec les sous-variables suivantes :

- duration : Une durée au bout de laquelle s’arrêtera le chronomètre
- elapsedTime : Le temps depuis le lancement du chronomètre

Créer un chrono avant la boucle “while” et l’initialiser avec le temps saisi par l’utilisateur.

A chaque tour de boucle, on veut calculer le temps qui s’est écoulé en un tour de boucle?
Dans le while, implémenter le code suivant :

- Déclarer une variable “newTime” et l’initialiser avec “GetTime()”,
- Déclarer une variable “deltaTime” et l’initialiser avec le temps écoulé entre deux boucles du while : “newTime - currentTime”,
- Ajouter la valeur de “deltaTime” à la sous-variable “elapsedTime” du chrono,
- Mettre à jour “currentTime” à la valeur de “newTime”.

Réécrire la condition de la boucle “while” avec “chrono.time < chrono.duration”.

[T_T] FONCTIONS A IMPLEMENTER

`UpdateChrono()`

- Objectifs : Mettre à jour le chrono si celui-ci n'est pas terminé
- Paramètres : référence vers le chrono, un delta-time
- Valeur de retour : aucune

Ajouter la sous-variable "isFinished" à la structure "Chrono".

Si "chrono.isFinished" est vrai, "UpdateChrono()" ne fait rien.
Sinon, la fonction ajoute le "deltaTime" à "chrono.elapsedTime" puis met à jour "chrono.isFinished".

Réécrire la boucle "while" avec "UpdateChrono()" et "isFinished".
Bonus : Retirer le paramètre "deltaTime" et faire le calcul du temps écoulé depuis le dernier appel à "UpdateChrono". Une variable "lastUpdateTime" doit être ajoutée à la structure.

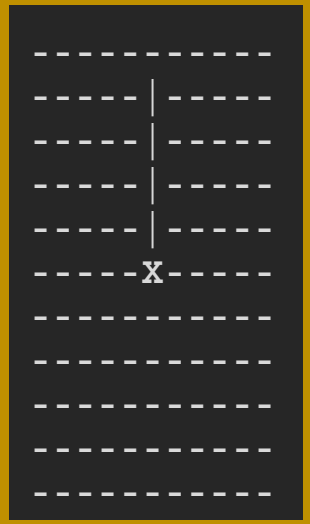
[-_-] TABLEAU D'AFFICHAGE

Créer un tableau "output" de 11 par 11 caractères (char) en variable globale (Note : On évitera autant que possible les variables globales).

Implémenter la fonction suivante :

`InitializeOutput()`

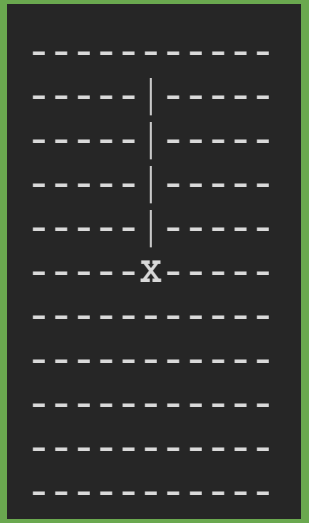
- Objectifs :
  - Remplit le tableau de caractères '-',
  - Ecrit 'X' en position centrale (5,5) du tableau,
  - Écrit '|' en positions (5,1) à (5,4) du tableau
- Paramètre : `aucun`
- Valeur de retour : `aucune`



[?_?] FONCTIONS A IMPLEMENTER

`DisplayOutput()`

- Objectifs :
  - Affiche le tableau lettre par lettre
- Paramètres : aucun
- Valeur de retour : aucune



La fonction commence par déclarer et remplir une chaîne de caractères (std::string) correspondant au texte à afficher à partir des valeurs du tableau.

La fonction ne réalisera qu'un seul appel à "std::cout".

[.^.] FONCTIONS A IMPLEMENTER

`ResetDisplay()`

- Objectifs :
  - Réinitialise la position du curseur d'écriture console
  - Réinitialise le tableau (appeler "InitializeOutput()")
- Paramètres : aucun
- Valeur de retour : aucune

Utiliser le code suivant pour réinitialiser la position du curseur console :

```
HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
COORD pos = { 0, 0 };
SetConsoleCursorPosition(hConsole, pos);
// nécessite #include <windows.h>
```

[@_@] AFFICHAGE TEMPS REEL

Avant la boucle “while” appeler le code suivant pour nettoyer une première fois l’ensemble de la console :

```
system("cls");
// nécessite #include <windows.h>
```

A chaque tour de boucle, appeler :

- La fonction “ResetDisplay”
- Puis la fonction “DisplayOutput”

[0_0] FONCTIONS A IMPLEMENTER

`WriteSecond()`

- Objectifs : Écrire ‘0’ dans une case du tableau en fonction de l’avancement de la seconde actuelle.
- Paramètres : référence vers le chrono
- Valeur de retour : aucune

Le caractère ‘0’ marquera l’écoulement d’une seconde. Il parcourra la largeur du tableau en 1 seconde puis reviendra au début.

La position du ‘0’ dans le tableau sera {posX, 0} :

```
int posX = ((int)(chrono.time * 11)) % 11;
```

[*_*] FONCTIONS A IMPLEMENTER

`WriteTimeLeft()`

- Objectifs : Ecrit '*' dans le tableau en fonction du temps restant
- Paramètres : référence vers le chrono
- Valeur de retour : aucune

Le caractère '*' marquera le temps restant avant la fin du chrono. Il fera un tour de cadran complet autour de X sur la durée totale du chrono.

La position du '*' dans le tableau sera :

```
int posX = (int)((cos(chrono.time * 2 * PI / chrono.duration - PI / 2) * 3)+ 5.5f);
int posY = (int)((sin(chrono.time * 2 * PI / chrono.duration - PI / 2) * 3)+ 5.5f);
```

[-_^] FONCTIONS A IMPLEMENTER

`WriteChrono()`

- Objectifs : Appelle “WriteSecond()” et “WriteTimeLeft()”
- Paramètres : référence vers le chrono
- Valeur de retour : aucune

Appeler “WriteChrono()” entre “ResetDisplay()” et “DisplayOutput()” dans la boucle “while”