Objectif

Créer un jeu qui vous permettra d'amasser des **$$$$** et replonger dans votre dépendance au développement de logiciel, et de maîtriser le concept de tableaux multidimensionnels. L'objectif est aussi de vous initier aux définitions de type énuméré et de type structure. Ce travail devrait aussi vous faire réaliser l'importance de bien isoler, dans son code, l'input du programme, le traitement des règles du jeu et l'affichage graphique.

Contexte de réalisation et de remise

Ce travail individuel sera effectué pendant les périodes de laboratoire. Les remises seront faites par LÉA à la date mentionnée pour chaque livrable.

Livrables

1. *C21-TP1-Nom-Prenom-V1.cpp*
   - Un seul fichier contenant tout le code source. Utiliser le fichier ***C21-TP1-Base.cpp*** et renommer le. Vous devez absolument utiliser les définitions (enum, struct et variable) de ce fichier.

2. *C21-TP1-Nom-Prenom-V1.exe*
   - Le programme.

Spécifications

Le jeu est composé d'un damier de 96 cases (8 lignes, 12 colonnes) sur lesquelles le joueur déplace un curseur jaune à l'aide des flèches du clavier. Chaque case du jeu est visible sous l'un ou l'autre des 4 aspects suivants: *case bleue*, *case dollars*, *case rose,* ou *case noire.* Au début, toutes les cases sont soit une *case noire*, soit une *case bleue*.

Lorsque le curseur se pose sur une case bleue, celle-ci change et devient une case rose ou bien une case *dollars* ($$$$). Cette case deviendra visible seulement quand le curseur aura bougé de nouveau. Quand une case *dollars* devient visible à la suite d'un premier passage du curseur, le joueur doit passer sur la case à nouveau afin d'amasser les $$$$ et gagner 1 point. La case *dollars* $$$$ deviendra alors une case rose. Finalement, si le curseur se pose sur une case rose celle-ci devient une case noire et il ne sera plus possible d'y repasser par la suite.

Remarque importante : Le positionnement initial du curseur, au début de la partie, n'est pas considéré comme un déplacement, et n'a donc aucun effet sur l'état de la case de départ.

Le jeu se termine quand les 15 points ont été ramassés ou si le curseur ne pourra plus se déplacer. Vous devez détecter, immédiatement après un déplacement, si le curseur est bloqué.

Le fichier **C21-TP1-Thiboutôt-Alain.exe** représente le programme à réaliser. Il faut le reproduire fidèlement.

Spécifications techniques

1. Les touches valides sont les touches de déplacement (ex : ➔, ←, 🡻, etc.) y compris les déplacements diagonaux sur le clavier numérique. Voir le fichier arrow-keys.cpp comme exemple.

```
enum Arrow_keys // Code ascii décimal des touches fléchées du clavier.
{
    key_up            = 72,
    key_up_left       = 71,
    key_up_right      = 73,

    key_left          = 75,
    key_right         = 77,

    key_down          = 80,
    key_down_left     = 79,
    key_down_right    = 81
};
```

2. Le déplacement logique du curseur est mémorisé avec les structures suivantes :

```
struct Pos
{
    int lig;
    int col;
};

struct Curseur
{
    Pos dep;
    Pos arr;
};
```

3. Vous devez vérifier que le mouvement demandé par l'utilisateur reste à l'intérieur des frontières du damier. Vous devez aussi refuser de bouger le curseur vers une case vide (noire).

4. L'ensemble des états possibles pour les cases du damier est spécifié par le type énuméré suivant :

```
enum État { CO, CS, CD, CF, CV };

CO --> case ordinaire       bleue
CS --> case surprise        bleue
CD --> case dollars         verte
CF --> case fragile         rose
CV --> case vide            noire


Liste des transitions possibles d'un état à un autre:

1) CO >--> CF
2) CS >--> CD
3) CD >--> CF
4) CF >--> CV
```

5. Le jeu et son damier sont spécifiés avec la structure et la variable suivantes :

```
struct Jeu
{
    int ptsAccum;
    État damier [LIG][COL];
};

Jeu jeu;
```

6. Après chaque déplacement valide, deux cases seulement doivent être rafraîchies à l'écran : la case de départ qui vient d'être libéré du curseur, et le curseur au nouvel endroit.

7. Chaque case du damier à l'écran est composée de 8 caractères identiques : 2 lignes x 4 caractères superposés.

   - *case bleue* : '\xB2'
   - *case rose* : '\xB0'
   - *dollar vert* : '\x24'
   - *dollar noire* : '\x20' ou '\x0'

8. Les cases sont espacées de 1 caractère vide (sur l'axe des x) et 1 ligne vide (sur l'axe des y)

9. Le curseur est affiché avec 2 x 4 caractères spéciaux (codes ascci décimals)

   - *curseur (haut)* : 201 ╔ 203 ╦ 203 ╦ 187 ╗
   - *curseur (bas)* : 200 ╚ 202 ╩ 202 ╩ 188 ╝

10. Le coin supérieur gauche de la 1ère case du damier, dans la fenêtre, est à la position START_X = 10 et START_Y = 2.

11. Vous devez manipuler la position logique du curseur (lig,col) dans les décisions de votre programme et calculer, au besoin, la position graphique (x,y) pour rafraichir certains éléments du jeu à l'écran.

```
x = START_X + ( col * DELTA_X )       où DELTA_X = largeur_case + espacement_vide
y = START_Y + ( lig * DELTA_Y )       où DELTA_Y = hauteur_case + ligne_vide
```

ex : la position logique (0,0) dans le damier équivaut à la position graphique (10,2) à l'écran.
ex : la position logique (1,1) dans le damier équivaut à la position graphique (15, 5) à l'écran.

Diagramme d'action (ébauche)

```
Afficher le damier

Initialiser la position de départ du curseur en : ligne = 0 et colonne = 0
Afficher le curseur

Faire

   Lire Touche
   Si Touche est une flèche

      Calculer la position d'arrivée du curseur avec la direction de la flèche et
      sa position de départ

      Si la position d'arrivée est règlementaire

         Vérifier si des dollars sont disponibles
         Modifier l'état de la case d'arrivée selon les règles

         Afficher le curseur à la position d'arrivée
         Afficher la case de départ où était le curseur

Tant que Points < 15 ET curseur non bloqué

Écrire le rapport
```