TD 9 : compilation séparée, graphiques

Exercice 1 (Compilation séparée)**.**
Consulter les quatre fichiers suivants et préciser leurs rôles respectifs :

factorielle.h

```
/** La fonction factorielle
 *  @param n un nombre entier positif
 *  @return n!
 **/
int factorielle(int n);
```

factorielle.cpp

```
#include "factorielle.h"

int factorielle(int n) {
    int resultat = 1;
    for ( int k = 1; k <= n; k++ )
        resultat = resultat * k;
    return resultat;
}
```

factorielle-test.cpp

```
#include <iostream>
#include "factorielle.h"
using namespace std;

/** Infrastructure minimale de test **/
#define ASSERT(test) if (!(test)) cout << "Test failed in file " \
         << __FILE__ << " line " << __LINE__ << ": " #test << endl

/** Les tests de la fonction factorielle **/
void factorielleTest() {
    ASSERT( factorielle(0) == 1  );
    ASSERT( factorielle(1) == 1  );
    ASSERT( factorielle(2) == 2  );
    ASSERT( factorielle(3) == 6  );
    ASSERT( factorielle(4) == 24 );
}

/** Cette fonction main ne sert qu'à lancer les tests **/
int main() {
    factorielleTest();
    return 0;
}
```

factorielle-exemple.cpp

```cpp
#include <iostream>
#include "factorielle.h"
using namespace std;

int main() {
    int n;
    cout << "Entrez un entier n" << endl;
    cin >> n;
    cout << n << "! = " << factorielle(n) << endl;
    return 0;
}
```

Graphiques

En TP, nous utiliserons la bibliothèque MLV [1] qui permet de faire simplement du multimédia en C et C++, que l'on soit sous Linux, Windows, MacOS ou autre. Cette bibliothèque est développée par une équipe menée par Adrien Boussicault, originellement pour l'enseignement à l'Université de Marne-la-Vallée. Elle est basée sur la bibliothèque SDL [2].

Voici un exemple de programme utilisant cette bibliothèque :

mlv-exemple.cpp

```
#include "MLV.h"
using namespace mlv;

int main() {
    // Crée et affiche une fenêtre de taille 640x480
    window_t fenetre = window_t( "Premier essai", "essai", 640, 480 );
    // Dessine un point rouge de coordonnées x=120 et y=50
    fenetre.draw_point( {120, 50}, color::red );
    // Actualise la fenêtre
    fenetre.update();
    // Attend 10 secondes
    fenetre.wait_seconds( 10 );
}
```

Noter dans cet exemple le nouveau type `window_t` qui représente une fenêtre, et les fonctions `draw_point`, `update` et `wait_seconds`. La syntaxe pour appeler ces fonctions est similaire à celle pour appeler les fonctions `size` ou `push_back` sur un tableau. Il s'agit en fait de *méthodes* que l'on appelle sur un *objet*.

Exercice 2 (Premiers dessins)**.**
En vous inspirant de l'exemple fourni, écrire des instructions (fragments de programme) qui, respectivement,

(1) dessinent un point noir (*black*) de coordonnées $(418, 143)$;

(2) dessinent un segment marron (*saddlebrown*) reliant les points $(100, 200)$ et $(200, 200)$;

(3) dessinent un segment marron reliant les points $(200, 300)$ et $(200, 400)$;

(4) dessinent le rectangle horizontal vide de contour marron dont les sommets diagonaux sont $(200, 200)$ et $(400, 300)$;

(5) dessinent un rectangle horizontal plein marron dont les sommets diagonaux sont $(400, 150)$ et $(500, 200)$;

(6) dessinent un segment marron reliant les points $(400, 300)$ et $(500, 400)$;

(7) dessinent un cercle marron de centre $(415, 145)$ et de rayon 10 ;
Indication : utiliser les fonctions `cos` et `sin` ;

(8) dessinent un disque rouge (*red*) de centre $(700, 100)$ et de rayon 50 ;
Indication : utiliser la définition d'un disque.

1. `http://www-igm.univ-mlv.fr/~boussica/mlv/`
2. `https://www.libsdl.org/`

Exercice ♣ 3 (Fonctions de dessin)**.**
Généraliser l'exercice 2 en implantant les trois fonctions suivantes. À noter qu'elles prennent la fenêtre où dessiner comme premier paramètre (`fenetre`). Le symbole `&` indique que cette fenêtre est passée par référence afin que les fonctions puissent la modifier. Vous pouvez essentiellement ignorer ce détail technique dont vous verrez les tenants et les aboutissants au deuxième semestre.

```
// Dessine un cercle non rempli
void cercle( window_t &fenetre, point_t centre, int rayon );
// Dessine un cercle rempli
void disque( window_t &fenetre, point_t centre, int rayon );
// Dessine un segment
void segment( window_t &fenetre, point_t p1, point_t p2 );
```

Exercice 4 (Souris et Clavier)**.**
Voici maintenant un fragment de programme interactif utilisant la bibliothèque MLV pour réagir à des actions avec la souris ou le clavier.

```
// Crée et affiche une fenêtre de taille 640x480
window_t fenetre = window_t( "Premier essai", "essai", 640, 480 );

// Attend que l'utilisateur clique sur le bouton gauche de la souris
// et stocke les coordonnées du point où a cliqué l'utilisateur dans
// la variable point.
point_t point = fenetre.wait_mouse();

// Affiche les coordonnées du point
cout << point.x << " " << point.y << endl;

// Attend que l'utilisateur tape sur une touche
// et stocke la touche et d'autres informations dans
// la variable evenement
event::keyboard_t evenement = fenetre.wait_keyboard();
```

Noter le nouveau type `point_t`. C'est un *type composite* : un point `point` est composé de ses deux coordonnées x et y, auxquelles on accède avec la syntaxe `point.x` et `point.y`. Techniquement parlant, il s'agit d'un *enregistrement* (*struct* en anglais) ; vous en verrez les détails au deuxième semestre.

On suppose de plus disposer d'une fonction permettant de dessiner un segment comme dans l'exemple suivant :

```
fenetre.draw_line({20,20}, {500,580}, color::red);
```

(1) En vous inspirant du programme précédent, écrire un programme qui attend que l'utilisateur clique sur deux points de l'écran puis qui trace le segment les reliants.

(2) Écrire un programme qui attend que l'utilisateur clique sur quatre points puis qui dessine le quadrilatère ayant ces quatre points comme sommets.

(3) Écrire un programme qui dessine des polygones de la façon suivante : l'utilisateur clique sur des points successifs. À chaque clic, le programme relie les deux derniers points. Si l'utilisateur clique près du point initial, le polygone se ferme, et le programme commence un nouveau polygone.