

## Laboratoire de Programmation Concurrente

semestre printemps 2015 - 2016

### Bandit manchot

Temps à disposition : 4 périodes (travail débutant en semaine 4)

## 1 Objectifs pédagogiques

- Mettre en oeuvre une application multi-threadée.

## 2 Cahier des charges

Nous désirons réaliser un bandit manchot semblable aux machines que l'on peut trouver dans un casino.

L'interface graphique vous est fournie sous la forme de classes C++ exploitant la librairie Qt.

Elle propose les éléments suivants :

- une fente pour l'introduction d'une pièce (réalisée par un bouton) ;
- un bouton *stop* ;
- trois rouleaux affichant cycliquement les nombres de 0 à 9 ;
- un afficheur de la valeur du jackpot ;
- un afficheur pour les messages de la machine à l'intention de l'utilisateur.

Une partie commence après avoir introduit une pièce. Les rouleaux doivent alors se mettre en marche, et afficher cycliquement les valeurs 0 à 9. L'utilisateur doit alors appuyer trois fois sur stop. A chaque pression, un rouleau doit s'arrêter, et laisser sur l'affichage sa valeur courante. Lorsque le troisième rouleau a été stoppé, si les trois fenêtres affichent la même valeur, l'utilisateur remporte la moitié du jackpot, et si deux des fenêtres affichent la même valeur, il gagne le quart du jackpot. Pour ces gains, l'arrondi est fait à l'entier immédiatement supérieur.

Un délai maximum de `DelaiLocal` est autorisé avant que l'utilisateur ne presse la touche *stop*, et ce pour chaque pression du bouton. Si l'utilisateur ne presse pas le bouton *stop* avant ce délai, tous les rouleaux doivent se bloquer, et la partie se termine par une victoire ou un échec.

## 3 Code

Le code de l'interface graphique est entièrement fourni. Vous n'avez que le fichier *bmmanager.cpp* à modifier. Il contient le corps des fonctions suivantes, définies dans *bmmanager.h* (un exemple de lancement de thread y est déjà présent).

```

class BmManager
{
public:
    /**
     * \brief Cette fonction est appelée une seule fois, au lancement de
     *        l'application.
     */
    void start();

    /**
     * \brief Cette fonction est appelée une seule fois, à la terminaison
     *        de l'application.
     */
    void end();

    /**
     * \brief Cette fonction est appelée lorsqu'une pièce a été insérée.
     */
    void pieceIntroduite();

    /**
     * \brief Cette fonction est appelée lorsque le bouton stop est enfoncé.
     */
    void boutonStop();
};

```

Toutes ces fonctions sont appelées par l'interface graphique, et donc par le thread principal de l'application. Il sera à vous de créer les threads nécessaires au bon fonctionnement du système.

Pour ce qui est des rouleaux, la fonction `usleep()` vous permet de faire attendre un thread pendant un certain nombre de microsecondes. Il est dès lors suggéré, voir même imposé, d'avoir un thread par rouleau.

Afin de modifier les valeurs des rouleaux, ainsi que le jackpot et les messages, vous disposez des fonctions définies dans *bandmitmanchot\_interface.h* :

```

#ifndef BANDITMANCHOT_INTERFACE_H_
#define BANDITMANCHOT_INTERFACE_H_

/**
 * Cette fonction permet de donner la valeur d'un des rouleaux. Les rouleaux sont
 * numérotés de 0 à 2, et la valeur est un entier entre 0 et 9. L'affichage est
 * alors modifié avec la nouvelle valeur.
 * \param rouleau Le numéro du rouleau, entre 0 et 2
 * \param valeur La valeur à afficher par le rouleau, entre 0 et 9
 */
void setValeurRouleau(int rouleau, int valeur);


/**
 * Cette fonction affiche le message passé en paramètres. Il permet d'indiquer
 * des informations pour l'utilisateur.
 * \param mess Une chaîne de caractères à afficher
 */
void setMessage(const QString mess);

/**
 * Cette fonction permet de définir le jackpot disponible. Il est alors
 * affiché dans la fenêtre.
 * \param jackpot Le jackpot courant
 */
void setJackpot(int jackpot);

```

Cette API ne vous permet que de modifier l'affichage, il est de votre devoir de gérer les valeurs locales dont vous avez besoin.

Le code à disposition se trouve sur le site du cours.

 Il n'est pas permis de modifier l'interface graphique.

## 4 Travail à rendre

---

- Les modalités du rendu se trouvent dans les consignes qui vous ont été distribuées.
- La description de l'implémentation, ses différentes étapes, la manière dont vous avez vérifié son fonctionnement et toute autre information pertinente doivent figurer dans les programmes rendus. Aucun rapport n'est demandé.
- Inspirez-vous du barème de correction pour connaître là où il faut mettre votre effort.
- Vous pouvez travailler en équipe de deux personnes au plus.

## 5 Barème de correction

---

Conception	25%
Exécution et fonctionnement	20%
Codage	15%
Documentation et en-têtes des fonctions	30%
Commentaires au niveau du code	5%
Robustesse	5%