Rapport de Projet en Programmation Orientée Objet C++

Sallio Romane and Fontaine Pierre

21 avril 2017

Table des matières

Première partie

Introduction

Pour ce projet nous avons choisis d'implémenter un "Dashboard", cela se présente sous forme d'une fenêtre graphique qui est développer en C++ grâce au framework QT largement utilisé dans le monde professionel de l'informatique.

Nous avons choisis QT car ce framework comparé à d'autres comme la SDL, SFML ... permet une implémentation objet. Chaque éléments repose sur le principe de l'objet, ainsi une fenetre peut se dériver en une classe spéciale ainsi que tous les autres éléments. D'autre part il est 'Javascript Like' c'est à dire la gestion des évènements pour l'IHM est semblable au code Javascript.

Il implémente une todolist, la recherche de la météo grâce à une API, la recherche d'actualité grâce à un flux RSS, une horloge et un convertisseur de vitesse et de température.

Afin de pouvoir introduire les notions vues en TP nous avons implémenter la classe List qui fait intervenir

Deuxième partie

Implémentation d'une liste avec template

0.1 liste

list.h

Nous avons ré-implémenté le container liste pour les besoins du projet. Grâce aux templates que nous avons introduit il est possible de l'utiliser pour stocker n'importe quel type d'éléments.

Troisième partie

Coeur du projet

0.2 ToDoList

Ce module permet de choisir une date à partir de la date actuelle en cliquant sur celle choisi dans le calendrier, cela parmet l'ouverture d'une nouvelle fenêtre dans laquelle l'utilisateur doit rentrer les données titre, note et heure pouis appuyer sur le bouton valider. Si l'utilisateur ne souhaite pas enregistrer les données il lui suffit de choisir le bouton quitter.
Lorsque l'on clique sur le bouton valider les différentes informations sont enregistrées dans le fichier TDLdata.
Ensuite, après avoir créé une ou plusieurs tâche, on peut cliquer sur le bouton Mise à jour liste ce qui affiche une liste des titres des différentes notes. A droite de tous les titres se trouve un bouton open qui permet d'afficher dans une nouvelle fenêtre les information associées à chaque titre.

0.3 Météo

Ce module à pour objectif d'afficher la météo pour les 5 prochains jours. Il suffit de cliquer sur l'onglet du jour que l'on souhaite obtenir.

0.4 Actualité

Ce module à pour objectif de fournir les titres principaux de l'actualité fournis par le flux RSS du site LeMonde.

0.5 Horloge

Ce module a pour objectif de d'afficher la date et l'heure en ce mettant à jours automatiquement toute les 0,1s.

0.6 Convertisseur

Ce module permet le conversion de vitesse (miles, mètres, kilomètres) et de température (celsius, fahrenheit, kelvin). Pour cela il suffit de choisir dans les différents menus déroulants le genre de conversion (vitesse ou température), l'unité du chiffre que l'on souhaite convertir et celle qu'on veut obtenir et l'utilsateur doit également saisir le réel à convertir.

Ensuite il lui suffit de cliquer sur le bouton et la conversion s'affiche dans la case en dessous du menu déroulant de l'unité que l'on souhaite convertir.

0.7 Annexe

Classe abstraite, fonction fiend, fonction virtuel :

```
class AbstractMesureUnite{
  protected:
    double _value;
  public:
    AbstractMesureUnite();
    AbstractMesureUnite(double);
    AbstractMesureUnite(const AbstractMesureUnite&);
    ~AbstractMesureUnite();
    AbstractMesureUnite &operator=(const AbstractMesureUnite&);

    void setValue(double);
    double getValue()const;

    virtual void afficher(ostream &flux) const = 0;

    friend ostream& operator<<(ostream& flux,const AbstractMesureUnite& u){
      u.afficher(flux);
      return flux;
    }
};
```

Template :

```
template <class T>
  class List{
    protected:
      struct cellule{
        cellule *suivant;
        T valeur;
      };
      typedef cellule* liste;
```

```
    liste _l;
  public:
    List();
    ~List();
    List(const List<T> &l);
    List<T> operator=(const List<T>);

    void insertElemAtPos(int pos,T elt);
    void deleteElemAtPos(int pos);
    T getElemAtPos(int pos)const;
    int getLength()const;
    bool empty()const;
};
```

Une classe mère (Temperature) et ses deux classes filles (Celsius, Kelvin) :

```
class Temperature : public AbstractMesureUnite{
  public:
    Temperature();
    Temperature(double);
    virtual void afficher(ostream &flux) const = 0;
    virtual double getCelsius()const = 0;
    virtual double getFahrenheit()const = 0;
    virtual double getKelvin()const = 0;
};

class Celsius : public Temperature{
  public:
    Celsius();
    Celsius(double);
    void afficher(ostream &flux)const;
    double getCelsius()const;
    double getFahrenheit()const;
    double getKelvin()const;
};

class Kelvin : public Temperature{
  public:
    Kelvin();
```

```
    Kelvin(double);
    void afficher(ostream &flux)const;
    double getCelsius()const;
    double getFahrenheit()const;
    double getKelvin()const;
};
```