Feuille de TP n°7 :

Programmation événementielle : C++ / bibliothèque wxWidgets

Objectif pédagogique

Cette séance de TP a pour but de vous familiariser avec le framework wxWidgets (hiérarchie de classes et méthodes) permettant de créer un programme ave interface graphique.

TRAVAUX A RENDRE ….

… **Individuellement** sur le webCampus avant jeudi 19 mai 2016, 23h59, dans une archive .zip à votre nom et groupe de TP

1.- Réponses aux questions suivantes de la feuille de td n°6 :

- **Exercice A.-**
  - **(a)** Quel est l'événement wxEVT_*xxx* à intercepter ?
  - **(b)** Au sujet de la méthode événementielle prenant en charge cet événement :
    - Dans quelle portion de code sera-t-elle déclarée (écriture de son entête) ? Pourquoi ?
    - Dans quelle portion de code sera-t-elle définie (écriture du corps) ?
  - **(c)** Utiliser une table d'événements (EVENT_TABLE) pour associer l'événement et la méthode qui le traitera.
    - Dans quelle portion de code sera-t-elle déclarée ?
    - Dans quelle portion de code sera-t-elle peuplée (ajout d'une ligne l'association dentifiantObjet←→méthodeEvenementielle) ?
- **Exercice B.-**
  - **(a)** Au vu du comportement décrit ci-dessus, identifier les différents états du système et les événements qui font changer le système d'état.
    Identifier les gestionnaires d'événement à prévoir et préciser à quels événements ils sont associés : nom, but, événement qu'il traite.

2.- Sources du programme deux.

RESSOURCES A VOTRE DISPOSITION POUR REALISER LE TP

~ Cette feuille de TP n°7.
~ Ressources disponibles sur le WebCampus :
  ~ documentation wxWidgets : partielle (.pdf à utiliser en td), et complète (wx.chm à utiliser en tp ou bien la documentation en ligne : **http://docs.wxwidgets.org/2.8/**)

DIRECTIVES PARTICULIERES A CETTE SEANCE

Dans votre répertoire de travail habituel, **et dans le dossier `M2105-IHMs`**,

- créez un répertoire nommé **`tp7`**

A.- deux.exe : Additionneur simplifié (Deuxième partie et fin)

(a) Dans votre répertoire tp6, **créez** un projet CodeBlock intitulé **deux** avec les options ci-dessous :
- template wxWidgets, version 2.8.x
- Projet **SANS** générateur d'interface GUI, Application de type FRAMED
- Compilateur par défaut Gnu-gcc, mode **DEBUG uniquement**
- Bibliothèques **STATIQUES**, codage des caractères en mode **UNICODE** (meilleure portabilités des applications),
- Création d'un projet **VIDE** (vous associerez au projet les sources mis à votre disposition une fois le projet vide créé)
- Application sera en mode **GUI** (et non en mode CONSOLE)
- Associer les bibliothèques **_WXDEBUG_** et **Debug wxWidgets** au compilateur Gnu-gcc
- Application cible (type de l'exécutable .exe à fabriquer) : de type **GUI**

Résultat attendu :

Un projet nommé deux, vide (sans fichier source attaché) et sans générateur d'interface associé a été créé dans votre répertoire \tp7 .

(b) Dans le répertoire **`tp7\deux`**, **copiez** les 4 fichiers sources C++ contenus dans votre répertoire tp6\deux. **Compilez** avant de commencer toute modification. Si le fichier exécutable n'est pas généré, c'est sûrement que le projet n'a pas été créé avec les bonnes options : recommencez la création du projet en vous reportant aux copies d'écran fournies sur la feuille de tp n°6.

(c) Modifier/Compléter les codes sources selon la progression proposée sur la feuille de td n°6.

(d) Une fois l'Additionneur terminé, enlevez le message de Bienvenue indiquant que l'application est en construction.

(e) Ajouter à votre programme une boîte de messages demandant la confirmation de fermeture de l'application, comme cela a été traité dans la question 1.- de la feuille de td n°6.

Indications

1) Pour transformer un texte (type wxString) en nombre entier

Utiliser la méthode `ToLong()` de la classe `wxString,` de la manière suivante :

```
    wxTextCtrl *text_X;
    … en supposant que la zone de saisie text_X contient une valeur à tester…

    wxString texte;
    texte = text_X ->GetValue(); // cf méthode GetValue() de la classe wxTextCtrl

 if (texte.ToLong(&x, 10) == true)
  { // actions à faire dans le cas où la variable x de type long a été correctement
    // initialisée à partir du contenu de texte
  }
 else
  { // actions à faire dans le cas où la variable x de type long n'a pas été correctement
    // initialisée à partir du contenu de texte
  }

/*
   la méthode ToLong() tente de convertir le contenu textuel de la variable texte
   - en Nombre entier (long)
   - codé en base 10
   - qu'il range dans la variable x.
   Si l'opération réussit, retourne TRUE, sinon, retourne FALSE
*/
```

Consultez la documentation en ligne qui vous a été fournie pour comparer l'exemple utilisé avec l'entête / la signature de la méthode…..

2) Pour affecter une variable de type wxString avec une valeur

Utiliser l'opérateur **`<<`** de manière analogue à la redirection sur `cout,` comme indiqué ci-dessous :

```
    wxString texte;
    int nombre = 5;
    texte << (10 + nombre) ;  // texte contient la chaine wxString équivalente à "15"
    int rep = wxMessageBox(texte) ;

    texte << wxT("Bonjour") ;  // contient la chaîne wxString équivalente à "Bonjour"
    int rep = wxMessageBox(texte) ;
```

Pour vous en persuader, utiliser `wxMessageBox` pour afficher les valeurs intermédiaires de la variable `texte`.