INFO0030 : Projet de Programmation
Calculatrice

B. Donnet, E. Marechal

1 Contexte

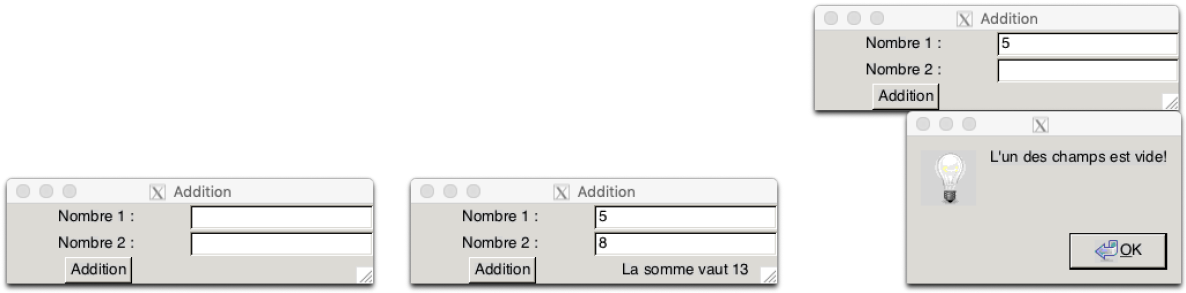
Il vous est demandé, dans ce projet, d'implémenter une *calculatrice* simple (i.e., elle ne permet que de faire des additions) sous la forme d'une interface graphique.

L'objectif de ce projet est donc de vous permettre d'apprendre à manipuler la programmation événementielle et, en particulier, la librairie GTK+2 vue au cours. [1] Cette librairie, écrite en C/C++ a été écrite pour réaliser le projet GIMP, qui est une très bonne alternative libre au logiciel de retouche d'image Photoshop. GTK est aussi à la base de l'environnement graphique GNOME

2 Interface Graphique

L'objectif de cette section est de vous présenter l'interface graphique générale de votre calculatrice.

La Fig. 1 présente l'interface graphique générale de votre calculatrice ainsi que le comportement attendu de votre application.



(a) Interface graphique générale (b) Addition réussie (c) Gestion des erreurs

FIGURE 1 – Interface graphique et cas d'utilisation de la calculatrice.

En particulier, la Fig. 1(a) présente la calculatrice. L'interface graphique est assez simple. Il y a deux champs pour insérer les nombres (le type est `GtkEntry` – voir Sec. 3.2) et un bouton pour effectuer l'addition. Si vous le désirez, vous pouvez customiser le bouton d'addition avec une image (voir Sec. 3.1).

L'idée du programme est que l'utilisateur entre deux nombres entiers dans les champs respectifs. Une fois que c'est fait, l'utilisateur clique sur le bouton "Addition" et le résultat s'affiche sous la forme d'une chaîne de caractères (voir Fig. 1(b)).

Votre programme doit pouvoir gérer les erreurs. En effet, rien n'empêche l'utilisateur de ne rien entrer dans les champs avant de cliquer sur le bouton "Addition" ou encore entrer des valeurs autres que entières (e.g., une chaîne de caractères). Dans ce cas, votre programme doit avertir l'utilisateur de son erreur via une fenêtre pop-up (voir Fig. 1(c)) contenant un message relatif à l'erreur (i.e., le message pour un champ vide ne sera pas le même qu'en cas d'entrée non entière).

1. cfr. `http://www.gtk.org`

3 Compléments GTK

Cette section a pour but de vous apporter des compléments à GTK+2 de façon à pouvoir réaliser ce projet.

3.1 Image et Bouton

Il est possible, en GTK, d'afficher une image dans un bouton. Supposons que nous disposions d'une image, `img1.jpg` et qu'on veut l'afficher dans un bouton `pBouton`. Ceci nécessite trois étapes :

1. charger l'image depuis le disque dur. Eventuellement, on peut redimensionner l'image.
2. créer le bouton.
3. placer l'image dans le bouton

Le bout de code ci-dessous illustre ces trois étapes.

```
GtkButton *charge_image_bouton(){
    GtkWidget *pBouton;
    GdkPixBuf *pb_temp, *pb;
    GtkWidget *image;

    //1. Charger l'image et la redimensionner (100*100 pixels)
    pb_temp = gdk_pixbuf_new_from_file("img1.jpg", NULL);
    if(pb_temp==NULL){
        printf("Erreur de chargement de l'image img1.jpg!\n");
        return NULL;
    }
    pb = gdk_pixbuf_scale_simple(pb_temp, 100, 100, GDK_INTERP_NEAREST);

    //2. Créer le bouton
    pBouton = gtk_button_new();

    //3. Placer l'image
    image = gtk_image_new_from_pixbuf(pb);
    gtk_button_set_image(GTK_BUTTON(pBouton), image);

    return pBouton;
}//fin charge_image_bouton()
```

Si on désire remplacer l'image existante d'un bouton, il suffit d'appeler, sur ce bouton, la procédure `gtk_button_set_image` avec la nouvelle image.

Notez qu'à chaque fois que l'on remplace l'image du bouton, on crée un nouvel objet de type `GtkImage` [2] en appelant `gtk_image_new_from_pixbuf`. On pourrait supposer qu'il serait préférable de libérer l'image avant de créer la nouvelle. En réalité, lorsqu'on appelle `gtk_button_set_image`, GTK libère la mémoire occupée par la `GtkImage` précédemment affichée.

3.2 Entrée

Un moyen simple de capturer, avec GTK, l'input de l'utilisateur sous forme d'une chaîne de caractères est d'utiliser un `GtkEntry`, i.e., une entrée avec un champ texte d'une seule ligne.

Le bout de code ci-dessous vous illustre comment créer une variable de type `GtkEntry`.

```
GtkWidget *pEntry = gtk_entry_new();
```

Pour récuper l'input de l'utilisateur, il suffit de faire comme suit :

```
const char *s = gtk_entry_get_text(pEntry);
```

Notez bien que le résultat d'une récupération est une chaîne de caractères. Il vous faudra donc éventuellement la transformer en fonction de ce que vous désirez en faire.

Si vous désirez aller plus loin avec `GtkEntry` (e.g., limiter le nombre de caractères introduits par l'utilisateur, rendre invisible l'input de l'utilisateur via des caractères `'*'`, ...), n'hésitez pas à consulter la document disponible sur Internet.

2. A l'instar (par exemple) de `GtkWindow`, un object de type `GtkImage` est aussi de type `GtkWidget`.

3.3 Label

Pour afficher le résultat d'une addition, dans l'exemple de déroulement donné plus haut, on peut utiliser la fonction `sprintf()`, qui fonctionne comme la fonction `printf()`, mais qui prend en premier paramètre un tableau de `char` dans lequel le texte produit va être stocké.

Exemple : La procédure suivante est utilisée pour afficher le résultat de l'addition dans le programme correspondant aux captures de la Sec. 2. Les paramètres `n1` et `n2` représentent les deux nombres introduits par l'utilisateur. Le label est, lui-aussi, passé en argument de la procédure.

```
void addition_mise_a_jour(GtkWidget *pLabel, int n1, int n2){
  char message[100];
  sprintf(message, "La somme vaut : %d", n1+n2);
  gtk_label_set_text(GTK_LABEL(pLabel), message);
}//fin addition_mise_a_jour()
```

4 Enoncé du Projet

Il vous est demandé d'écrire un programme en langage C implémentant la calculatrice telle que décrite dans ce document.

En particulier, votre projet devra :

- être soumis dans une archive `tar.gz`, appelée `calculatrice.tar.gz`, via la plateforme de soumission.[3] La décompression de votre archive devra fournir tous les fichiers nécessaires dans le répertoire courant où se situe l'archive. Vous penserez à joindre tous les codes sources nécessaires.
- définir les fonctionnalités nécessaires à la mise en place de l'IHM de la calculatrice (cfr. Sec. 2).
- être modulaire, i.e., nous nous attendons à trouver un (ou plusieurs) header(s) et un (ou plusieurs) module(s).
- s'assurer que les structures de données proposées sont implémentées comme des types opaques (quand cela s'avère pertinent).
- être parfaitement documenté. Vous veillerez à spécifier correctement chaque fonction/procédure/structure de données que vous définirez. Votre documentation suivra les principes de l'outil doxygen.
- appliquer les principes de la programmation défensive (vérification des préconditions, vérification des mallocs, ...). Pensez à libérer la mémoire allouée en cours de programmation afin d'éviter les fuites de mémoire.
- comprendre un Makefile permettant au moins de

  1. compiler votre projet. L'exécution de la commande

     ```
     $>make
     ```

     permet de générer un fichier binaire, exécutable, appelé `calculatrice`.
  2. Le fichier binaire généré devra pouvoir être exécuté de la façon suivante :

     ```
     $>./calculatrice
     ```

  3. générer la documentation doxygen. L'exécution de la commande

     ```
     $>make doc
     ```

     devra produire de la documentation au format HTML dans le sous-répertoire `doc/`.

  La compilation doit se faire en spécifiant au minimum les options de compilation `--std=c99 --pedantic -Wall -W -Wmissing-prototypes`.

Il est impératif de respecter **scrupuleusement** les consignes sous peine de se voir attribuer une note de 0/20 pour non respect de l'énoncé

3. Pour rappel, l'adresse de la plateforme de soumission est `https://submit.montefiore.ulg.ac.be`