INFO0030 : Projet de Programmation
Chasse au Trésor

B. Donnet, E. Marechal, M. Goffart, E. Wansart

Alerte : Évitez de perdre bêtement des points...

- Nous vous conseillons **vivement** de relire et d'appliquer les **guides de style et de langage**, mis à votre disposition sur eCampus (INFO0030, Sec. Supports pour le Cours Théorique). Cela vous permettra d'éviter de nombreuses erreurs et de perdre des points inutilement.
- Nous vous conseillons de consulter la **grille de cotation** utilisée pour la correction des projets afin d'éviter de perdre bêtement des points. Elle est disponible sur eCampus (INFO0030, Sec. Procédures d'Évaluation).
- Prenez le temps de lire attentivement l'énoncé. La plupart des réponses aux questions que vous vous posez s'y trouvent.
- Votre code sera compilé et testé sur les **machines CANDI qui servent de référence**. La procédure à suivre pour se connecter en SSH aux machines CANDI est disponible sur eCampus. Veillez donc à ce que votre code fonctionne dans cet environnement. Si vous n'avez pas de compte sur ces machines, veuillez contacter Marc Frédéric.
- Votre solution ne pourra pas être générée entièrement ou partiellement à l'aide d'un outil d'Intelligence Artificielle (e.g., ChatGPT, Blackbox, etc.).

1 Contexte

Jack Sparrow, pirate bien connu, a caché son trésor dans un coffre. Afin que le trésor ne soit pas trop facilement découvert, il a enterré, dans une même île, trois coffres. Un seul d'entre eux contient le trésor, les deux autres étant vides. Les coffres ne peuvent être ouverts que lorsqu'ils sont côte à côte. Vous avez découvert l'île et déterré les trois coffres. Cependant, il est impossible d'ouvrir les trois coffres. Seuls deux coffres pourront être ouverts. Votre objectif est de découvrir le coffre contenant le trésor (et vous en emparer).

Le jeu, Chasse au Trésor, se déroule comme suit : les trois coffres, fermés, sont affichés à l'écran. L'un d'eux contient le trésor, les deux autres étant vides. Le joueur choisit un des coffres. L'esprit de Jack (i.e., le programme) ne montre pas le contenu du coffre choisi par le joueur. À la place, il ouvre un des deux autres coffres en s'assurant d'abord que le coffre qu'il va ouvrir est vide. Le joueur choisit alors un nouveau coffre parmi les deux restés fermés [1]. Il peut donc maintenir son choix initial ou non. L'esprit de Jack ouvre le coffre finalement choisi par le joueur et déclare la victoire (i.e., le trésor a été découvert) ou défaite du joueur (i.e., le trésor n'a pas été découvert).

L'objectif de ce projet est de vous permettre d'apprendre à manipuler la programmation événementielle et, en particulier, la librairie Gtk+2 vue au cours. [2] 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+2 est aussi à la base de l'environnement graphique Gnome.

2 Exemples de Déroulement du Jeu

Cette section donne des exemples de déroulement du jeu, Chasse au Trésor.

1. Cliquer sur le coffre déjà ouvert ne doit pas avoir d'effet.
2. cfr. http://www.gtk.org.

2.1 Partie Perdue

(a) situation initiale (b) 1ère tentative (c) 2ème tentative

FIGURE 1 – Exemple de partie perdue.

La Figure 1 donne un exemple de partie perdue, depuis la situation initiale jusqu'à la défaite. Lors de la première tentative, nous avons cliqué sur le coffre à gauche. Nous avons maintenu notre choix lors de la deuxième tentative.

2.1.1 Partie Gagnée

(a) situation initiale (b) 1ère tentative (c) 2ème tentative

FIGURE 2 – Exemple de partie gagnée.

La Figure 2 donne un exemple de partie gagnée, depuis la situation initiale (qui correspond à la situation finale de l'exemple précédent – cfr. le score) jusqu'à la victoire. Lors de la première tentative, nous avons cliqué sur le coffre à gauche. Lors de la deuxième tentative, nous avons cliqué sur le coffre du milieu.

3 Compléments GTK+2

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

3.1 Label

Pour afficher un résultat, 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 d'une addition dans un programme. Les paramètres `n1` et `n2` représentent les deux nombres introduits par l'utilisateur. Le label à mettre à jour est, lui-aussi, passé en argument de la procédure.

```
1 void mise_a_jour(GtkWidget *pLabel, int n1, int n2){
2    if(pLabel == NULL)
3       return;
4    char message[100];
5    sprintf(message, "La␣somme␣vaut:␣%d", n1+n2);
```

```
    gtk_label_set_text(GTK_LABEL(pLabel), message);
}//fin mise_a_jour()
```

3.2 Image et Bouton

Il est possible, en GTK+2, 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. Éventuellement, 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(){
  //1a. Charger l'image
  GdkPixBuf *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;
  }
  //1b. Redimensionner l'image en 100*100 pixels
  GdkPixBuf *pb = gdk_pixbuf_scale_simple(pb_temp, 100, 100, GDK_INTERP_NEAREST);
  if(pb == NULL){
   printf("Erreur lors de la redimension de l'image!\n");
   return NULL;
  }

  //2. Créer le bouton
  GtkWidget *pBouton = gtk_button_new();
  if(pBouton == NULL){
   printf("Erreur lors de la création du bouton\n");
   return NULL;
  }

  //3. Placer l'image
  GtkWidget *image = gtk_image_new_from_pixbuf(pb);
  if(image == NULL){
   printf("Erreur lors de la création de l'image\n");
   return NULL;
  }
  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`[3] 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+2 libère la mémoire occupée par la `GtkImage` précédemment affichée.

3.3 Invalider un Bouton

Sur les captures (cfr. Sec. 2), on peut remarquer que le bouton "Recommencer" est parfois grisé. Lorsque c'est le cas, l'utilisateur ne peut pas cliquer dessus.

Pour mettre un bouton `pBouton` dans un état grisé, on appelle :

```
gtk_widget_set_sensitive(pBouton, FALSE);
```

Pour lui redonner son état normal et le rendre cliquable :

```
gtk_widget_set_sensitive(pBouton, TRUE);
```

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

4 Contrôler l'Aléatoire

En C, on peut produire une suite pseudo-aléatoire d'entiers en utilisant la fonction `rand()` de la bibliothèque standard (`stdlib.h`). A chaque fois qu'un programme appelle cette fonction, elle retourne l'entier suivant dans la suite. Les entiers de la suite sont compris entre `0` et `RAND_MAX` (une constante prédéfinie).

Par exemple, le programme suivant :

```c
#include <stdlib.h>
#include <stdio.h>

int main(){
  for(int i = 0; i < 10; i++){
    int a = rand();
    printf("%d\t", a);
  }//fin for - i

  printf("\n");

  return EXIT_SUCCESS;
}//fin programme
```

produit la sortie suivante :

```
16807    282475249        1622650073       984943658        1144108930
470211272        101027544        1457850878       1458777923       2007237709
```

Dans le cas du Chasse au Trésor, on cherche à tirer aléatoirement un nombre entre 0 et 2 (compris), pour désigner le coffre qui contient le trésor. Dans ce cas, il suffit d'utiliser le reste de la division par 3 :

```c
int coffre_tresor = rand()%3;
```

Si on veut tirer un entier entre A et B compris, on peut faire :

```c
int x = rand() % (B-A+1) + A;
```

Pour produire un flottant entre A et B, on peut faire :

```c
double x = (double) rand() / RAND_MAX * (B-A) + A ;
```

La suite produite par `rand()` n'est pas vraiment aléatoire. En particulier, si on lance le programme précédent plusieurs fois, on obtiendra exactement la même suite.

Pour rendre le résultat apparemment plus aléatoire, on peut initialiser la suite en appelant la fonction `srand()` (définie aussi dans `stdlib.h`), à laquelle on passe en paramètre une valeur d'initialisation. La suite produite par `rand()` après cet appel dépendra de la valeur que l'on passe à `srand()` lors de l'initialisation. Une même valeur d'initialisation produira la même suite.

Par exemple, le programme suivant :

```c
#include <stdlib.h>
#include <stdio.h>

int main(){
  srand(2);

  for(int i = 0; i < 5; i++){
    int a = rand();
    printf("%d\t", a);
  }//fin for - i

  printf("\n");
  srand(2);

  for(int i = 0; i < 5; i++){
    int a = rand();
    printf("%d\t", a);
  }//end for - i

```

```
20    printf("\n");
21
22    return EXIT_SUCCESS;
23  }//fin programme
```

produit la sortie suivante :

```
33614    564950498        1097816499       1969887316       140734213
33614    564950498        1097816499       1969887316       140734213
```

Pour que la suite soit vraiment imprévisible, on peut passer à `srand()` une valeur dépendant de l'heure. Par exemple, en utilisant la fonction `time()` de la librairie standard :

```
1  srand(time(NULL));
```

5 Énoncé du Projet

Il vous est demandé d'écrire un programme implémentant Chasse au Trésor tel que décrit dans ce document.

Votre projet devra :

- Être soumis dans une archive `tar.gz`, appelée `tresor.tar.gz`, via la Plateforme de Soumission. 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. Votre archive doit être chargée sur la Plateforme de Soumission pour le **Mardi 01/04/2025, 08h00** au plus tard.
- Définir les fonctionnalités nécessaires à la mise en place de l'IHM du jeu Chasse au Trésor (cfr. Sec. 2).
- Être modulaire, i.e., nous nous attendons à trouver un (ou plusieurs) header(s) et un (ou plusieurs) module(s). Néanmoins, vous ne devez **pas** utiliser le pattern MVC.
- 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/énumération/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 `tresor`.
  2. Le fichier binaire généré devra pouvoir être exécuté de la façon suivante :
     ```
     $>./tresor
     ```
  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/`.
  4. Tout Makefile qui se respecte doit aussi inclure la commande
     ```
     $>make clean
     ```
     qui permet de supprimer tous les exécutables et librairies (et autres fichiers) générés antérieurement par le Makefile.

Toute question relative au projet peut être posée sur le forum de la page web du cours (eCampus).

Tout projet ne compilant pas se verra attribuer la note de 0/20.

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.

Tout étudiant ne soumettant pas à minima une archive vide se verra attribuer une note d'absence.