TP n° 3 :

Buts : fonctions, tableau
Durée : 1 semaine

Fonctions simples

Ecrire une fonction qui calcule la puissance n d'un réel x et un programme principal qui utilise cette fonction. Le prototype de cette fonction sera : `double puiss(double x, int n)` ;

Remarque 1: on peut optimiser cette fonction en remarquant que $x^n = x^{n/2} * x^{n/2} * x^{n\%2}$ et en appliquant directement cette forme.
Remarque 2: on peut optimiser encore plus cette fonction en remarquant que $n = \sum_{i=0}^{d} a_i 2^i \quad avec \quad a_i = \{0,1\} \quad et \quad x^n = \prod_{i=0}^{d} (x^{2^i})^{a_i}$. Le calcul des puissances de x utile s'effectue facilement par une seule multiplication à chaque itération, sans oublier que a_i valant 0 ou 1, x^{ai} puissance vaut x ou 1.

Master Mind : fonctions et tableau

L'objectif est ici de réaliser la gestion du jeu Master Mind. : vous devez deviner quelles sont les N (5) couleurs, choisies au hasard parmi M (7) couleurs possibles par votre programme. Une même couleur peut être tirée plusieurs fois. Le joueur fait une proposition de configuration de couleurs pour deviner la configuration initiale. Le programme doit répondre

- combien de couleurs sont à leur place,
- combien de couleurs sont effectivement présentes mais mal placées.

A partir de cette réponse, le joueur propose une nouvelle configuration à laquelle le programme répond et ainsi de suite jusqu'à ce que la bonne réponse soit trouvée par le joueur.
On ne demande pas de réaliser un programme qui devine la solution, mais simplement un programme qui indique quelles sont les couleurs correctement devinées ou placées.

Les couleurs sont représentées par un entier (1..M) et une configuration est un tableau de N entiers.
N et M sont des constantes définies initialement à l'aide des instructions du préprocesseur par :

```
#define N 5
#define M 7
```

Algorithme

Une configuration est donc un tableau d'entiers. Vous utiliserez deux tableaux : un pour la configuration initiale à deviner, un pour la configuration du joueur, qui changera donc à chaque tour de jeu. Vous décomposerez votre problème en plusieurs étapes plus simples. Les différentes étapes à réaliser (ou l'algorithme) sont les suivantes

Créer les variables utiles (2 tableaux : machine et joueur, 3 entiers : nbcoups, bp et mp)
Initialiser le tableau à deviner
Répéter
    Lire la configuration proposée par le joueur
    Incrémenter nbcoups
    Calculer le nombre de couleurs devinées et bien placées et mettre ce nombre dans bp
    Calculer le nombre de couleurs devinées et mal placées et mettre ce nombre dans mp
tant que le nombre de bien placées est différent de N.
Afficher le nombre de coups

Travail à réaliser

Pour ce premier programme utilisant plusieurs fonctions, nous n'utiliserons **qu'UN seul** fichier. Toutes les fonctions, y compris le programme principal, seront écrites dans cet unique fichier master.c.

Nous vous donnons plusieurs fonctions que vous n'avez pas à écrire et que vous pouvez récupérer sur le site et recopier dans le fichier master.c:

1. Une fonction qui remplit un tableau de n cases (couleurs) tirées au hasard parmi les M couleurs possibles pour créer la configuration initiale (celle que l'ordinateur choisi). L'entête de la fonction est `void init(int tab[], int n, int M).`
2. Une fonction qui affiche un tableau de n cases (couleurs) à l'ecran. L'entête de la fonction est `void affiche(int tab[], int n).`
3. Une fonction qui retourne combien de couleurs sont présentes mais mal placées dans une configuration proposée par le joueur par rapport à la configuration initiale vous est donnée sur le site. Elle retourne le nombre de couleurs mal placées. L'entête de la fonction est `int malplace(int joueur[], int machine[], int n, int M).`

Il vous reste à écrire les fonctions suivantes dans le fichier master.c et leurs prototypes dans le fichier master.h:

4. Ecrire une fonction qui lit une configuration au clavier (celle du joueur). Pour simplifier, cette fonction doit lire des entiers (<M) au clavier. C'est la fonction d'affichage qui fera la conversion entre un entier et les couleurs des pions. L'entête de la fonction est `void lire(int tab[], int n).`
5. Ecrire une fonction qui indique combien de couleurs sont bien placées dans une configuration proposée par le joueur par rapport à la configuration initiale. Elle retourne le nombre de couleurs bien placées. L'entête de la fonction sera `int bienplace(int joueur[], int machine[], int n).`

Il faut aussi écrire un programme principal, c'est à dire la fonction main(), dans le fichier master.c.

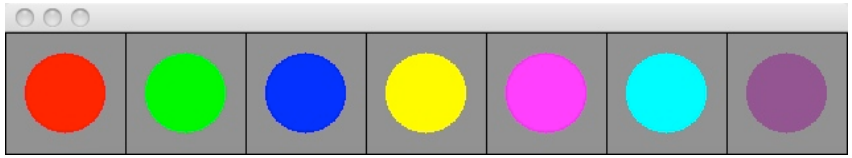
6. Ecrire un programme qui réalise les différentes opérations pour jouer au master mind : tirage initial, lecture de la configuration du joueur au clavier, réponse du programme, boucle et gestion de la fin de la partie. Inclure le fichier master.h.
7. Vous pouvez créer votre exécutable avec la commande unix : `gcc master.c –o master`

Attention : utilisez une approche incrémentale pour développer ce programme. Vérifiez chaque fonction une à une. Ecrivez une première fonction (affiche par exemple) et faites un programme pour vérifier son fonctionnement. Ajouter ensuite la fonction lire et vérifier là encore son fonctionnement, puis passer à la fonction bien placée, vérifier, etc..

Facultatif : version graphique : ne vous lancez dans cette partie que si vous etes en avance
Vous pouvez utiliser la bibliothèque graphique SDL pour réaliser une version graphique couleur de votre master mind. Il faut en plus de la SDL installer les bibliothèques SDL_draw, SDL_image.

Dans un premier temps, faites l'affichage dans une fenêtre graphique en vous inspirant du code ci dessous. Les configurations sont entrées au clavier comme précédemment.

Vous pourrez vous inspirer du programme suivant qui affiche la fenêtre :



Fichier grmastermind.c

```
#include <stdio.h>
#include "SDL_phelma.h"
/*      Nombre de point du master mind */
#define N 7
/*      Nombre de couleurs du master mind */
#define M 7
/*      Rayon d'un pion du master mind. La taille d'une case est alors 3R+1 */
#define R 30

int main() {
  int i, t[N] ; /* Tableau representant le master mind */

       /* Variable permettant de manipuler une fenetre graphique */
  SDL_Surface* f1=NULL;

       /* Creation d'une fenetre graphique de taille
  f1=newfenetregraphique((N)*(3*R+1)+2,3*R+3);
  if ( f1==NULL) {printf("Creation de fenetre impossible \n"); exit(1); }

        /* On remplit la fenetre avec un rectangle gris */
  SDL_FillRect(f1, NULL, SDL_MapRGB(f1->format,128,128,128));

        /* On dessine une grille */
  Grille(f1,1,N+1,3*R+1,3*R+1,0,0);

        /* Initialisation du tableau du master mind*/
  for (i=0; i<N; i++) t[i]=i;

        /* Affichage graphique du master mind et attente         */
  Affichemaster(f1,t,N,R);
  printf("Tapez une touche pour continuer \n"); getchar();

        /* Deuxieme essai mais couleur au hazard*/
  for (i=0; i<N; i++) t[i]=rand()%M;
  Affichemaster(f1,t,N,R);
  printf("Tapez une touche pour continuer \n"); getchar();
}
```

Pour compiler, il faut copier le fichier Makefile (voir le site des tp) suivant et utiliser la commande unix make pour créer un exécutable.
fichier Makefile :

```
DIRSDL=/users/prog1a/C/librairie/2011

CFLAGS=-g -O2 -I$(DIRSDL)/include
LDFLAGS=$(LFLMAC)  -g  -L$(DIRSDL)/lib  -L/usr/local/lib  -lSDLmain  -lSDL  -lSDL_ttf  -
lSDL_image -lSDL_draw -lSDL_phelma

grmastermind : grmastermind.o
        gcc -o grmastermind grmastermind.o  $(LDFLAGS)
grmastermind.o : grmastermind.c
        gcc -c $(CFLAGS)  grmastermind.c
```