TP5. Utilisation des tableaux

Dimitri Galayko

1 Traitement d'une liste de notes

Nous avons une fonction permettant de saisir les notes et de les mettre dans un tableau global notes:

```
#include <stdlib.h>
#define MAX 1000

float notes[MAX];

void saisir_notes(){

float note;
int i=0;
do{
printf("saisir note; entrez -1 pour terminer :");
scanf("%f", note);

notes[i]=note;
}while(note>=0);
return;
}
```

Ecrivez une fonction
int compter_mention_bien()
qui retourne le nombre de notes avec mention "bien", c.a.d., appartenant à l'intervalle [14,16].

2 Champ graphique quadrillé et damier

Le but de cet exercice est de préparer la construction d'une bibliothèque de fonction qui permet de manipuler des cases rectangulaires sur un champ graphique. Cela permettra de réaliser un grand nombre de programmes tels que "jeu de la vie", "labyrinte", etc.

A la base nous avons un champ graphique de N pixels sur x, M pixels sur Y.

Ce champs est quadrillé en blocs rectangulaires (cases) de taille LX sur x, LY sur y. Au total, il y a donc $XMAX = N/LX$ cases sur x, et $YMAX = N/LY$ cases sur y.

On peut imaginer que sur chaque case on peut afficher différents objets: rien (case blanche), rectangle plein, un motif représentant un mur de labyrinte, un dessin de bonhome, etc.

L'avantage de cette approche est que l'on adresse les cases non pas par leurs coordonées physiques liées aux pixels, mais par leurs coordonnées logiques qui vont de 1 à $XMAX$ pour x, et $YMAX$ pour y. Comme pour l'écran physique, l'axe y a l'origine en haut de l'écran, et les coordonnées y s'incrémentent en descendant.

Exo 2.1. Afficher un rectangle plein. Pour commencer, on crée une fonction qui permet d'afficher des rectangles pleins de couleur courante du stylo. La fonction ne s'occupe pas de la couleur du rectangle, ce sera le programme appelant qui positionnera la couleur du stylo à la bonne valeur. La fonction prend en argument les coordonnées logiques de la case.

```
void afficher_rectangle_plein_case(int x, int y){
...
}
```

Exo 2.2. Damier aléatoire. En utilisant la fonction écrite précédemment et la fonction générant des nombres entiers aléatoires, écrivez un programme qui dessine sur le champ graphique des rectangles pleins placés dans ces cases. Chaque rectangle a une couleur choisie aléatoirement dans la gamme suivante: blanc, bleu, rouge, jaune, vert.

Exo 2.3. Construction d'un labyrinte. On va maintenant refaire un exercice très proche du "damier aléatoire", avec une différence: les coordonnées des cases "remplies" de rectangles sont affichées donnés par les valeurs des éléments d'un tableau.

Pour cela, on déclare un tableau à deux dimensions de taille $XMAX$, $YMAX$, on l'initialise avec des 0 et 1, de sorte à définir un motif que l'on souhaite afficher.

Par exemple, si on a un champs avec 20x20 cases ($XMAX = YMAX = 20$), le tableau sera declaré comme ceci:

```
int cases[YMAX][XMAX]={\
          {0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, \
          {1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1}, \
          {1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1}, \
          {1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1}, \
          {1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1}, \
          {1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1}, \
          {1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1}, \
          {1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1}, \
          {1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};
```

On peut apercevoir que ce tableau définit des tunnels dans un labyrinte.

Ensuite, dans chaque case avec les coordonnées logiques [x,y] on dessinera un rectangle plein *seulement si* l'élément correspondant du tableau *cases* est 1.

Ecrivez une fonction *dessiner_labyrinte* qui réalisera ce dessin. La fonction ne prend et ne retourne aucun argument. Le tableau *cases* est déclaré comme un tableau global.