Licence Informatique 2013-2014 **J1INAW02 - Programmation 1**
Université Bordeaux 1 **Devoir Surveillé 1h20**

Tous documents interdits

1 Questions de cours

On considère un programme en C composé de trois fichiers `client_mot.c`, `mot.c` et `mot.h`. Le fichier `client_mot.c` contient la fonction `main` du programme. Le fonction `main` utilise des fonctions du module `mot`.

1. Combien de fichiers objets sont construits lors de la compilation de ce programme ? Comment s'appellent-ils et que contiennent-ils ?
2. Donner les commandes permettant de construire ces fichiers. Pourquoi faut-il utiliser les options `-g` et `-Wall` ?
3. Comment s'appelle l'étape consistant à construire le fichier exécutable à partir des fichiers objet et en quoi consiste-t-elle ? Donner la commande permettant de mettre en œuvre cette étape pour le programme des questions précédentes.
4. Comment appelle-t-on le fichier `mot.h`, quelles informations contient-t-il, et pourquoi doit-il impérativement être inclu dans les deux autres fichiers source ?

2 Chaînes de caractères

Soit le code suivant.

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

char *mystere(char *s, char c) {
  assert (s != NULL);
  for ( ; *s != '\0'; s++)
    if (*s == c)
      return s;
  return NULL;
}

int main(void) {
  char *s = "il etait une fois une histoire";
  char *s1 = mystere(s, 'u');
  if ( s1 != NULL )
  {
    printf("%s\n", s1);
    char *s2 = mystere(s1 + 1, 'u');
    if ( s2 != NULL )
       printf("%s\n", s2);
  }
  return EXIT_SUCCESS;
}
```

Questions :

1. Qu'affiche ce programme ? Précisez (éventuellement, illustrez) les valeurs prises par les variables `s1` et `s2` dans la fonction `main`.
2. Expliquez ce que retourne la fonction `mystere` dans le cas général.
3. Ecrire la fonction `char *my_strrchr(char *s, char c);`
   qui renvoie un pointeur sur la dernière occurrence du caractère `c` dans la chaîne `s`. La fonction retourne `NULL` si `c` n'est pas trouvé dans `s`.

Rappel : le format `%s` de `printf` permet d'afficher une chaine de caractères (c'est à dire tous les caractères jusqu'au caractère nul `'\0'`).

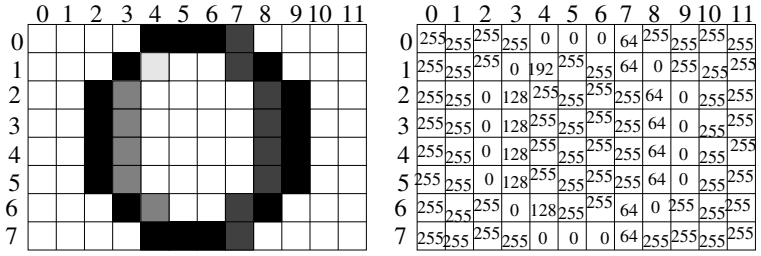
3 Module Image

Nous nous intéressons au traitement d'images numériques. Une image est décrite par une matrice rectangulaire de pixels. Nous ne considérons dans la suite que les images en niveau de gris : les pixels sont codés par un octet, c'est-à-dire par un nombre positif compris entre 0 (noir) et 255 (blanc).

Le but de cet exercice est d'écrire un module `Image` de manipulation d'images en niveaux de gris.

On représentera des images en niveaux de gris en ne retenant que les dimensions de l'image et les valeurs correspondantes à chaque pixel :
– la taille d'une image est définie par un couple de nombres entiers positifs `(largeur,hauteur)`.
– chaque pixel est défini par un triplet `(col,lig,c)` où `col` est l'indice de la colonne, `lig` est l'indice de la ligne de la matrice et `c` est le niveau de gris du pixel.

Par exemple l'image suivante a pour taille (12,8), l'indice de colonne varie de 0 à 11 et l'indice de ligne de 0 à 7 :



	0	1	2	3	4	5	6	7	8	9	10	11
0	255	255	255	255	0	0	0	64	255	255	255	255
1	255	255	255	0	192	255	255	64	0	255	255	255
2	255	255	0	128	255	255	255	255	64	0	255	255
3	255	255	0	128	255	255	255	255	64	0	255	255
4	255	255	0	128	255	255	255	255	64	0	255	255
5	255	255	0	128	255	255	255	255	64	0	255	255
6	255	255	255	0	128	255	255	64	0	255	255	255
7	255	255	255	255	0	0	0	64	255	255	255	255

FIGURE 1 – Image représentant un cercle discret et codage associé.

3.1 Implémentation du module

Ci-dessous l'interface du module `Image` :

```
#ifndef _IMAGE_H
#define _IMAGE_H

typedef struct image *Image;

extern Image   creerImage(int largeur, int hauteur);
/* cree une image de taille (largeur, hauteur) dont les pixels sont a zero */

extern void    libererImage(Image this);
/* libere la memoire utilisee par l'image this */

extern Image   copieImage   (Image this);
/* retourne une copie de l'image this: les pixels de l'image retournee */
/* ont la meme valeur que les pixels de l'image this */

extern unsigned short   lirePixelImage(Image this, int colonne, int ligne);
/* retourne le niveau de gris du pixel dont les coordonnees sont (colonne, ligne) */

extern void   ecrirePixelImage(Image this, int colonne, int ligne, unsigned short c);
/* ecrit le niveau de gris c dans le pixel dont les coordonnees sont (colonne, ligne) dans l'image this */

extern int    lireLargeurImage(Image this);
/* retourne la largeur de l'image this */

extern int    lireHauteurImage(Image this);
/* retourne la hauteur de l'image this */

extern Image  extractSubImage(Image this, int colonne, int ligne, int largeur, int hauteur);
/*cree une nouvelle image en extrayant la sous image de this a partir des pixels qui sont situes */
/* entre [colonne, colonne + largeur] et [ligne, ligne+hauteur] */


extern void   afficheImage(Image this);
/* affiche l'image this ligne par ligne */

#endif
```

1. Rappeler en quelques lignes ce qu'est une interface minimale. Est-ce que cette interface est minimale ? Pourquoi ?
2. Les pixels d'une image seront représentés par un tableau d'entiers à deux dimensions alloué dynamiquement. **Ce tableau est un tableau de tableaux**. En plus du tableau représentant les pixels la structure `image` contiendra `largeur` et `hauteur`. Expliciter cette structure (on utilisera le type entier suffisant à représenter un nombre entre 0 et 255).
3. Écrire la fonction `creerImage` qui crée une image dont la taille est définie par les paramètres `largeur` et `hauteur` puis l'initialise en appellant une méthode statique
   `void initImage(Image this,int largeur, int hauteur)` (chaque pixel sera initialisé à 0).
4. Écrire la fonction `libererImage` qui libère tout l'espace mémoire alloué par la fonction `creerImage`.
5. Écrire les fonctions `lireLargeurImage` et `lireHauteurImage`.
6. Écrire les fonctions `lirePixelImage` et `ecrirePixelImage`. On veillera à vérifier la pertinance des coordonnées du pixel dont on veut respectivement connaître ou modifier la valeur.
7. Écrire la fonction `afficheImage` qui affiche l'image à l'écran.
8. Il ne faut pas que votre implémentation contienne de duplication de code.
   Expliquer en quoi les fonctions `copierImage` et et `extractSubImage` sont presque idéntiques.
   Prendre en compte la consigne ci-dessus lors de l'écriture de leur code.

3.2 Programme de test

Ecrire un programme de test qui crée une image de taille (100,50) où les pixels d'une même ligne auront pour niveaux de gris l'indice de la ligne.