UGA - UFR IM^{2}AG **Master 2 CCI**

Devoir surveillé du 23 octobre 2018

Durée : une heure – **Documents autorisés : une feuille A4 recto-verso + fiche de traduction Algo ↔ C**

Exercice 2 (∼ 4 points)

On considère les fonctions `f1`, `f2` et `f3` suivantes :

```
int f1 (int a, int b) {
   return a == b ;
}

int f2 (int *a, int *b) {
   return a == b ;
}

int f3 (int a, int b) {
   a = b ;
   return a ;
}
```

Q1. Ecrivez en C un programme principal qui comporte deux appels à la fonction `f1` tel que l'un des appels renvoie la valeur 0 (pour "faux") et l'autre appel renvoie une valeur différente de 0 (pour "vrai"). Vous pouvez dans ce programme principal déclarer et initialiser les variables que vous souhaitez, mais vous ne pouvez pas modifier la fonction `f1`.

Q2. Même question que **Q1** mais en considérant cette fois-ci la fonction `f2` au lieu de `f1`.

Q3. Indiquez les valeurs de `a` et `b` après l'appel à `f3` sur le code suivant :

```
int a=3, int b=4;
a = f3(a, b) ;
/* valeurs de a et b ? */
```

Exercice 2 (∼ 4 points)

En langage C les chaînes de caractères sont représentées dans des tableaux de caractères (de type `char`) et se terminent toujours par le caractère `'\0'` (marque de fin de chaîne). On donne les spécifications de trois fonctions inspirées de la bibliothèque `<string.h>` :

— `void strcpy (char dest[], char src[]) ;`
Copie dans le tableau `dest` la chaîne de caractères (terminée par `'\0'`) qui est contenue dans le tableau `src` ; il n'y a pas de vérification sur la taille du tableau `dest`.
— `void strncpy (char dest[], char src[], int n) ;`
Cette fonction est identique à `strcpy`, sauf que seuls les `n` premiers caractères de `src` sont copiés. S'il n'y a pas de `'\0'` dans ces `n` premiers caractères, la chaîne résultante dans `dest` ne sera pas terminée par `'\0'`.
— `int strlen (char s[]) ;`
Renvoie le nombre de caractères de la chaîne `s`, sans compter le caractère `'\0'`.

Q1. Ecrivez en C le code de la fonction `strcpy`.

Q2. On veut copier dans le tableau `t` ci-dessous le plus long préfixe possible de la chaîne `"mystère"`. Que risque t-il se passer à l'exécution du code suivant :

```
#define TAILLE 4
char t[TAILLE] ;
strcpy(t, "mystere") ;
```

Q3. Ré-écrivez ce code de manière correcte (c'est-à-dire en copiant le plus long préfixe possible de `"mystère"` dans `t`, et en le terminant par `'\0'`) à l'aide des fonctions `strlen` et `strncpy`.

Exercice 3 (∼ 12 points)

On s'intéresse à des images de taille fixe représentées par des séquence de pixels [1]. Ces séquences de pixels sont mémorisées dans des tableaux, où chaque case du tableau code la couleur du pixel correspondant. On considèrera deux types d'image :
— des images à trois couleurs (Blanc, Gris ou Noir) ;
— des images à 256 niveaux de couleur ;
On utilisera pour cela le lexique suivant :

```
NbCouleur : la constante entiere 255
TailleImage : la constante entiere 1024
PixelBGN : le type {Blanc, Gris, Noir}
PixelC : le type entier sur [0,NbCouleur]
ImageBGN: le type tableau sur [0,TailleImage] de PixelBGN
ImageC: le type tableau sur [0,TailleImage] de PixelC
```

Q1. Donnez une traduction de ce lexique en langage C.

Q2. On souhaite transformer une image à 256 niveaux de couleurs (de type `ImageC`) en une image à trois couleurs (de type `ImageBGN`). On suppose pour cela que les valeurs faibles de PixelC (celles proches de 0) correspondent à des couleurs "claires", et les valeurs élevées (proche de 255) à des couleurs "foncées".

On propose donc d'utiliser l'action suivante pour faire réaliser cette transformation :

1. un pixel désigne un "point" de l'image

seuilBlanc : la constante entiere 75
seuilNoir : la constante entiere 200

Transformer : action (la donnee Src : une ImageC, le resultat Dest : une ImageBGN)
debut
    pour i parcourant 0..TailleImage
        selon Src[i]
            cas Src[i] $\leq$ SeuilBlanc : Dest[i] $\leftarrow$ Blanc
            cas Src[i] $\geq$ SeuilNoir : Dest[i] $\leftarrow$ Noir
            sinon : Dest[i] $\leftarrow$ Gris

Traduisez cette action en C.

Q3. Ecrivez en C un programme principal qui :

1. déclare une image de nom `imgSrc` et de type ImageC et une image de nom `imgDst` et de type ImageBGN ;
2. lit au clavier le contenu de `imgSrc` ;
3. transforme `imgSrc` en `imgDst`
4. affiche à l'écran le contenu de `imgDst`
   (on affichera la valeur 0 pour `Blanc`, la valeur 1 pour `Gris` et la valeur 2 pour `Noir`).

Q4. On dispose d'une fonction `affPixel(unsigned int x, unsigned int y, PixelC pix)` qui permet d'afficher au point de coordonnées `(x,y)` le pixel `pix`.

A l'aide de cette fonction écrivez les deux fonctions suivantes (cf. Figure) :

1. `affImageLigne (unsigned int x, unsigned int y, ImageC P)`, qui affiche l'image `P` sur une seule ligne, avec `P[0]` aux coordonnées `(x,y)`, `P[1]` aux coordonnés `(x,y+1)`, `P[2]` aux coordonnés `(x,y+2)`, etc.
2. `affImageCarre (unsigned int x, unsigned int y, ImageC P)`, qui affiche l'image `P` sous forme de carré de 32 x 32 pixels (sachant que $32 \times 32 = 1024$), telle que `P[0]` soit aux coordonnées `(x,y)`, `P[1]` aux coordonnés `(x+1,y)`, `P[32]` aux coordonnés `(x,y+1)`, etc., `P[1023]` aux coordonnés `(x+31,y+31)`,

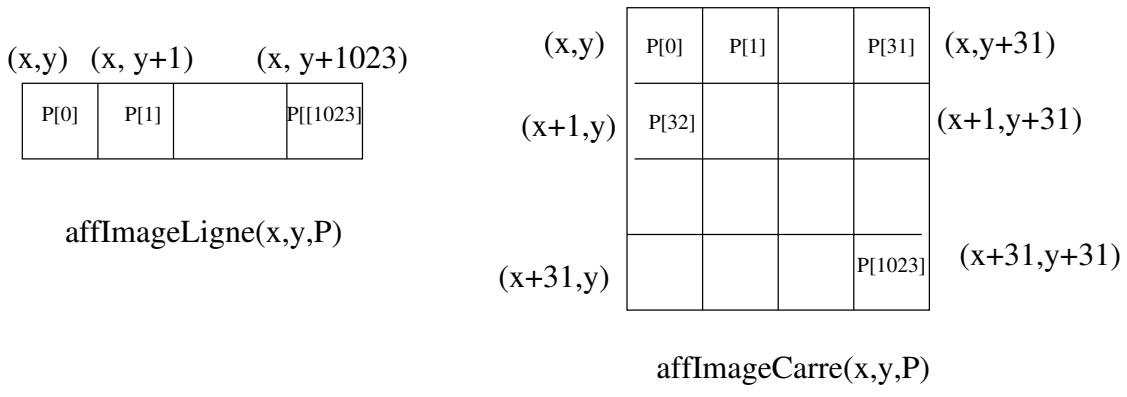

FIGURE 1 – Affichage de l'image `P` en "ligne" et en "carré"