

Mini-projet de langage C - Anonymat photographique

Paul Gaborit

IFIA M1 – Avril 2019 – IMT Mines Albi

1 Consignes

Ce travail s'effectue en binôme. **Un seul des membres du binôme** déposera sur campus une archive (**.zip**, **.rar**, **.7z**, **.tgz...**) **portant les deux noms du binôme** et contenant :

1. un document (en PDF) contenant :
 - (a) le nom et le prénom de chacun des membres du binôme,
 - (b) le schéma bulle, l'algorithme et la description du fonctionnement de chacune des fonctions que vous aurez conçues,
2. le fichier source (le fichier **.c**) du programme que vous aurez réalisé.

2 Objectif

L'objectif de ce mini-projet est de concevoir et d'écrire en langage C un programme permettant de cacher plusieurs parties circulaires d'une image (pour préserver l'anonymat de certains sujets par exemple). Ce programme sera nommé **anonymat**.

3 Les étapes du fonctionnement du programme **anonymat**

L'exécution du programme **anonymat** se déroule en trois grandes étapes :

1. Il demande à l'utilisateur le nom (ou le chemin d'accès) d'un fichier image à traiter. Il charge cette image en mémoire (l'image **in**) et la duplique dans une seconde image (**out**).
2. Ensuite il demande la description d'un cercle à rendre anonyme : le rayon r puis les coordonnées x et y du centre (l'unité de ces trois valeur est le pixel). Il calcule tout d'abord la moyenne des couleurs des pixels couverts par ce cercle dans l'image d'origine (**in**) puis remplace par cette couleur moyenne la couleur de ces même pixels mais dans l'image résultat (**out**).

Il traite ainsi une suite de cercles en demandant à chaque fois de nouveaux paramètres. La saisie par l'utilisateur d'un rayon négatif ou nul signifie la fin de la suite de cercles.

3. Une fois tous les cercles traités, le programme demande le nom (ou le chemin d'accès) d'un fichier image dans lequel il enregistre l'image résultat (**out**).

4 L'ébauche du programme

Nous vous fournissons une première ébauche du programme **anonymat.c** (voir figure 3 page 5). Toutes les interactions avec l'utilisateur sont déjà programmées et fonctionnelles.

En revanche, ce programme affiche des messages **TODO** (via des appels à **printf**) indiquant une action à réaliser.

Chacun de ces messages devra être remplacé par une ligne de code effectuant l'action attendue (si l'action nécessite plusieurs lignes de code, vous pourrez appeler une fonction que vous aurez préalablement ajoutée dans le programme).

En pratique, cela signifie que vous devrez concevoir et coder au moins trois fonctions supplémentaires : **duplicat_image**, **moyenne_cercle** et **remplir_cercle**.

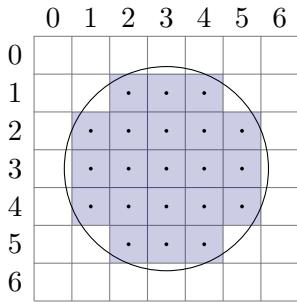
5 Éléments pratiques

Voici quelques éléments pratiques pour vous aider à réaliser votre programme.

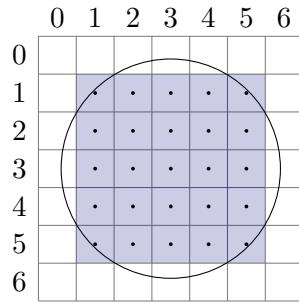
5.1 Quels pixels sont dans un cercle ?

On considère que les coordonnées d'un pixel sont le centre de ce pixel. Ainsi, le centre du pixel en haut à gacuhe d'une image (de coordonnées 0,0) est l'origine de notre repère.

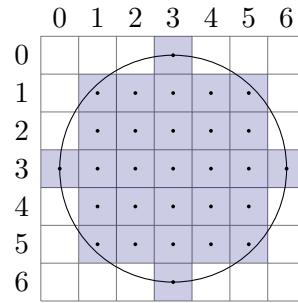
Pour qu'un pixel soit considéré comme faisant partie d'un cercle, son centre doit être dans le cercle. La figure 1 page suivante illustre ces explications.



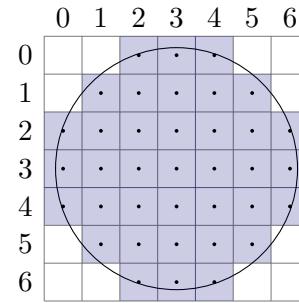
(a) Rayon 2.7



(b) Rayon 2.9



(c) Rayon 3.0



(d) Rayon 3.2

FIGURE 1 – Les pixels colorés font partie du cercle. Dans les quatre graphiques, le centre est en (3,3).

Il pourra s'avérer pratique de concevoir une fonction permettant de tester si un point est dans un cercle.

5.2 Images à traiter

Dans le dossier `images` de l'archive du projet, nous vous fournissons des fichiers `.txt` et des images `.ppm` à traiter. Chacun des fichiers `.txt` contient une série de réponses à fournir au programme `anonymat` pour traiter l'image correspondante.

La figure 2 page 4 affiche côté à côté l'image d'origine et l'image résultat (après application du fichier d'ordres associé).

5.3 Pour répondre plus vite

Lors de vos tests, il peut être long de répondre manuellement à toutes les questions du programme. Pour automatiser la prise en compte d'un fichier d'ordres, vous pouvez utiliser l'interpréteur de commandes de Windows :

1. Copier le répertoire `images` dans le répertoire du projet `Code::Blocks`.
2. Ouvrir l'interpréteur de commandes de Windows (`cmd.exe`).
3. Se déplacer dans le répertoire du projet `Code::Blocks`. En supposant que ce dossier est sur le bureau et que vous avez nommé votre projet `anonymat`, utiliser la commande suivante :

```
cd Desktop\anonymat
```
4. Dans `Code::Blocks`, utiliser le bouton `Build` pour s'assurer que le programme est bien compilé.
5. Pour tester le fichier d'ordres `test.txt`, utiliser la commande suivante :

```
bin\Release\anonymat.exe < images\test.txt
```

Vous devez reproduire les étapes 4 et 5 après chaque modification ou correction de votre programme.

6 Quels algorithmes fournir ?

Pour chacune des fonctions que vous aurez ajoutées dans le programme `anonymat.c`, vous devrez fournir :

- un schéma bulle,
- un algorithme,
- une description de son principe de fonctionnement.

7 Les calculs avec des entiers et des réels

En langage C, lorsqu'un opérateur s'applique à des opérandes entiers, le calcul s'effectue dans les entiers (ainsi $11/2$ vaut 5). Si au moins l'un des opérandes est un réels, le calcul s'effectue dans les réels (ainsi $11.0/2$ ou $11/2.0$ vaut 5.5).

Lorsqu'on affecte un résultat réel dans une variable entière, il est tronqué. Pour maîtriser la manière dont le réel est tronqué, vous pouvez utiliser les trois fonctions suivantes de la bibliothèque mathématique :

1. `ceil()` qui renvoie le plus petit nombre entier n'étant pas strictement inférieur à son argument réel.
2. `floor()` qui renvoie la plus grande valeur entière qui n'est pas strictement supérieure à son argument réel.
3. `round()` qui renvoie la valeur entière la plus proche de son argument réel.

Ces fonctions vous seront sûrement utiles pour le calcul de la couleur moyenne ou pour parcourir tous les pixels d'un cercle.

8 Remarques

N'oubliez pas que les cercles demandés par l'utilisateur peuvent être en partie (ou même *en totalité*) en dehors de l'image.

Ils peuvent aussi être tellement petits qu'ils n'en-globent *aucun* pixel.

Le répertoire **images-reference** contient le résultat de images après application des ordres. Ces images peuvent vous servir de référence : vous pouvez comparer vos propres résultats avec ces images.



(a) Application des ordres `chats.txt` au fichier image `chats.ppm`.



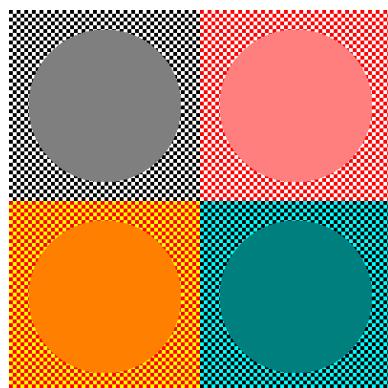
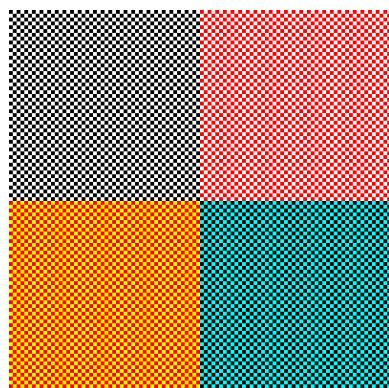
(b) Application des ordres `une-personne.txt` au fichier image `une-personne.ppm`.



(c) Application des ordres `garcon.txt` au fichier image `garcon.ppm`.



(d) Application des ordres `enfants.txt` au fichier image `enfants.ppm`.



(e) Application des ordres `test.txt` au fichier image `test.ppm`.

FIGURE 2 – Exemples de résultats d'utilisation du programme [anonymat](#) sur différentes images (avec leur fichier d'ordres associé).

```

#include <stdio.h>
#include "libimage.h"

void demande_double(char * message, double *p_double) {
    int res_scanf; /* le resultat du scanf */

    printf("%s", message);
    res_scanf = scanf("%lf", p_double);
    if (res_scanf != 1) {
        printf("Lecture d'un double impossible!\n");
        exit(1);
    }
}

void demande_chaine_1001(char * message, char *chaine) {
    int res_scanf; /* le resultat du scanf */

    printf("%s", message);
    res_scanf = scanf("%1000s", chaine);
    if (res_scanf != 1) {
        printf("Lecture d'une chaine impossible!\n");
        exit(1);
    }
}

int main() {
    char nom_image_in[1001]; /* nom du fichier image lu */
    char nom_image_out[1001]; /* nom du fichier image produit */
    double cr; /* le rayon d'un cercle */
    double cx, cy; /* les coordonnees du centre du cercle */
    image in; /* image d'origine */

    demande_chaine_1001("Nom du fichier image 'in' ?\n", nom_image_in);

    in = lire_image(nom_image_in);
    printf("TODO: dupliquer l'image 'in' dans l'image 'out'\n");

    demande_double("Rayon du cercle ?\n", &cr);
    while (cr > 0) {
        demande_double("Coordonnee X du centre ?\n", &cx);
        demande_double("Coordonnee Y du centre ?\n", &cy);

        printf("TODO: calculer la moyenne des couleurs du cercle (%g,%g) de rayon %g dans 'in'\n",
               cx, cy, cr);
        printf("TODO: appliquer cette moyenne dans le cercle (%g,%g) de rayon %g dans 'out'\n",
               cx, cy, cr);

        demande_double("Rayon du cercle ?\n", &cr);
    }

    demande_chaine_1001("Nom du fichier image 'out' ?\n", nom_image_out);
    printf("TODO: enregistrer l'image 'out' dans le fichier '%s'\n", nom_image_out);

    detruire_image(in);
    printf("TODO: detruire l'image 'out'\n");
}

```

FIGURE 3 – Source de l'ébauche du programme [anonymat.c](#)