SOUBIGOU Antoine **GE2**
PAILLARD Jean-Noël

Rapport
Projet Informatique

Concaténation de fichier TIFF

ensais

Sommaire

Introduction

Un des avantages d'un fichier TIFF est qu'il peut contenir plusieurs images de bonne qualité. Une commande en langage C permet alors de rassembler toutes les images de plusieurs fichiers en une seule sur un autre fichier, il s'agit de la commande **tiffp**.

Exemple

- **tiffp** a.tif(2,4-7) b.tif dest.tiff

  ↳ copie les images 2 et 4 à 7 de a.tiff et celle de b.tiff dans dest.tif

- **tiffp** -o a.tif(1,7) b.tif dest.tiff

  ↳ copie les images 1 et 7 de a.tiff et celle de b.tiff dans dest.tif avec pour chaque image, d'abord l'IFD, les valeurs des tags puis les données concernant l'image

☞ **Le but du projet** qui nous est demandé en Génie logiciel est donc ici de programmer en langage C la commande tiffp. Nous allons d'abord dans une première partie regarder comment est construit un fichier TIFF afin de mieux entreprendre la programmation de la fonction tiffp.

Dans la deuxième partie, nous allons présenter l'organigramme du programme, et plu s particulièrement des différentes fonctions

I Les fichiers TIFF

I.1. Description

Dans cette partie nous allons voir la description hexadécimale d'un fichier tiff. Il contient un **header** (premier segment du fichier), des **IFD** (Image File Directory) et des **tags**

Prenons un exemple

Header = 49 49 2A 00 6C EF 04 00

- 49 49 signifie que l'on est en Little Endian (LE)
- 2A 00 en LE vaut 00 2A soit 42, il s'agit donc d'un fichier TIFF
- 6C EF 04 00 est l'offset de l'IFD 0 (adresse = 4EF6C)

IFD 0 = 0D 00 FE 00 04 00 01 00 00 00 00 00 00 00

- 0D 00 vaut en fait 000D = 13, c'est-à-dire qu'il y a ici 13 tags ou bien 13 informations sur l'image
- FE 00 Tag FE = 254 ce qui correspond à New Subfile Type
- 04 00 Type 4 ce qui signifie que c'est un long
- 01 00 00 00 Nombre d'élément = 1 seul élément
- 00 00 00 00 Valeur du tag = 0

IFD 1 = 01 01 03 00 00 01 00 00 37 02 00 00

- 01 01 = 257, c'est un tag d'information sur la hauteur de l'image
- 03 00 type = 03 il s'agit d'un short ici
- 00 01 00 00 Nombre d'élément est égal à 1
- 37 02 00 00 = 237 qui vaut en décimal 567 (valeur de la hauteur de l'image)

A noté aussi que si par exemple pour l'IDF 1 le nombre d'élément était égal à 3, vu qu'il s'agit d'un short, il aurait fallu 6 octets (un short nécessite 2 octets).Or la place disponible pour la valeur du tag est de 4 octets. Dans ces conditions, ce n'est plus la valeur qui y est indiquée mais une adresse où se trouvera alors la valeur recherchée sur 6 octets.

I.2. Les commandes à effectuer

D'après l'organigramme (**voir annexe 1**), plusieurs étapes essentielles sont nécessaires pour réaliser ce programme.

- Tout d'abord, un sous-programme va devoir analyser la ligne de commande de tiffp (exemple dans l'onglet "le projet")). Les options telles que L, B, h, o et autres vont être lues. Ensuite il s'agit de compter le nombre de fichiers à concaténer. Ensuite il faut enfin créer le fichier de destination qui recevra toutes les images. Enfin, on détermine les images du fichiers sources, on ouvre le premier fichier source, on le parcourt jusqu'à atteindre le premier IFD à concaténer.

- Il s'agit maintenant d'analyser tous les IFD et les Tags. Dans un premier temps, il faut lire l'offset du premier IFD, le nombre total de Tag, l'adresse spécifiée par l'offset de l'IFD et la définition de celui-ci. Une fois ce Tag décrit, il est recopié dans un tableau, puis il faut passer au tag suivant ainsi de suite jusqu'à l'absence de Tag. Le tableau obtenu possède alors tous les Tags du fichier source.

- Dans cette partie, le fichier destination est crée et va recevoir les informations correspondant à la première image. Chaque Tag du tableau est lu. Si le nombre d'éléments du Tag est inférieur à 4 celui ci est identiquement recopié dans le fichier de destination. Sinon, le tag est recopier à la suite, mais les données elles sont écrites à la fin du fichier de destination (en prenant soin de remplacer la valeur courante du pointeur par l'adresse où l'on vient d'écrire. Tous les Tags sont passés en revu.

- On passe ensuite à l'IFD suivant, et le même protocole est effectué. Une fois la fin du fichier source atteint, on le ferme et on passe au suivant jusqu'à qu'il n'y ait plus de fichier à lire.

L'ensemble du programme peut-être décomposé en 3 fichiers

- **Tiffp.h** est le fichier qui contient l'énoncé de tous les prototypes, les variables externes etc...

- **Tiffp.c** sera le fichier qui contiendra la fonction principale *main( )*, raccourcie grâce aux appels aux fonctions.

- **Tiffp1.c**, c'est le fichier qui contient le détails de tous les prototypes. On y trouvera les fonctions *sym_suivant( )*, *tiff_analyse( )*, etc...

I.3. Conclusion

Nous avons donc précédemment compris la structure d'un fichier TIFF, ainsi que les manipulations qui vont nous être nécessaire afin de réaliser la concaténation de fichiers TIFF.

Dans la partie qui suit, nous allons expliquer clairement comment nous sommes arrivés à écrire les fonctions. Nous allons les expliquer et commenter leur action.

II Le programme

II.1. Cahier des charges

Nous avions besoin dans ce programme de fonctions précises. Tout d'abord, il nous fallait lire la commande de départ contenant les images de fichiers à concaténer ainsi que le fichier de destination, il s'agit de la commande **tiff_analyse**. Les fonctions suivantes sont nécessaires pour le décodage des images.

Nous allons maintenant en expliquer le contenu.

II.2. Les différentes fonctions

Nous allons voir dans cette partie les fonctions que nous avons mises en place. Chaque ligne de code est traduite en français afin de bien comprendre la procédure. *(programmation en C disponible en annexe)*

fonction sym_suivant

- **Cette fonction permet de passer au caractère suivant. (Annexe 2). Pcar représente ici le dernier symbole lu.**

```
tant que pcar est un espace
        on lie le caractère suivant
        si pcar est un caractère alphanumérique
                on met sym à fichier
                puis on met pcar dans p et on continu à lire le caractère suivant
                        on lit tant que pcar n'est pas un nombre
                        si pcar est un point
                                on met pcar dans p et on continu à lire le caractère suivant
                        fin si
                fin tant que
        fin si
        sinon si le symbole suivant est un entier
        on met sym à entier
        sinon on analyse le symbole
fin tant que
```

fonction intLE_intBE et shortLE_shortBE

- **Ces fonctions permettent de convertir en entier la valeur hexadécimale, sur 4 ou 8 octets, de « donnee ». Une transformation est parfois nécessaire car l'image est en LittleEndian. (Annexe 3)**

Exemple

Donnee = 6A 8B

↳ La transformation donne 00 6A 'ou logique' de 8B 00 soit 8B 6A

Exemple

Donnee = A1 B2 C3 D4

↳ La transformation donne D4 00 00 00 'ou logique' de 00 C3 00 00 'ou logique' de 00 00 B2 00 'ou logique' de 00 00 00 A1 soit D4 C3 B2 A1

Appel aux fonctions : printf("%d\n",intLE_intBE(donnee));

fonction format

- **Cette fonction permet de savoir si le fichier TIFF que l'on veut concaténer est codé en LittleEndian ou bien en BigEndian (Annexe 4)**

- On déclare les variables f, na, *p (pointeur du tableau a), a[]
- Si le fichier que l'on ouvre est vide
- Alors on écrit : c'est un échec, il est vide
- Fin si

- On initialise na à 0 compteur qui peut arrêter la lecture du fichier
- Tant que fichier non fini et que l'on n'a pas lu toutes les données désirées
- On met le caractère lu dans p
- On incrémente le pointeur
- On incrémente na (si na = 4 alors fin de la boucle)
- Fin tant que
- On ferme le fichier

- Le tableau 'a' contient maintenant les 4 premières données du fichier
- Si elles correspondent à 49 49
- On retourne la valeur 1
- Fin si
- Si elles correspondent à 29 29
- On retourne la valeur 2
- Fin si
- Sinon on retourne la valeur 3

Appel à la fonction : printf("%d",format("a5.tif"));

fonctions storeInt et loadInt

- **Selon qu'il s'agisse d'un LE ou un BE, loadInt convertit sur 4 octets la valeur du tableau p, et storeInt à l'inverse, sauvegarde dans un tableau la chaîne (Annexe 5)**

fonction tiff_analyse

Cette fonction analyse la ligne de commande, elle mémorise les images à inclure et à exclure des fichiers TIFF que l'on veut concaténer. Ces images sont placées dans une matrice qui, pou chaque ligne, conserve les numéros des images.(Annexe 6)

```
on va au caractère suivant
tant que sym est à fichier
        on va au caractère suivant
        si c'est un crochet gauche
                on va au caractère suivant
                si c'est un accent circonflexe
                        on va au caractère suivant
                        on lance une procédure des images à exclure
                fin si
                tant que c'est un entier
                        on va au caractère suivant
                        on garde en mémoire le dernier chiffre
                        si c'est un tiret
                                on va au caractère suivant
                                si ce n'est pas un entier
                                        message d'erreur
                                fin si
                        fin si
                        on copie alors dans un tableau le numéro de l'image à copier
                        sinon
                        si c'est une virgule
                                on va au caractère suivant
                                        si ce n'est pas un entier
                                                message d'erreur
                                        fin si
                                on ajoute au tableau le numéro de l'image
                        fin si
                        si c'est un crochet droit
                                on va au caractère droit
                                k++ ?
                                on va au caractère suivant
                        fin si
                        sinon message d'erreur
                fin tant que
                sinon on copie toutes les images
        fin si
fin tant que
```

fonction idtaille_tag

Cette fonction retourne un entier correspondant à la taille en octets du tag. Elle lit l'identifiant du tag, le type de valeur, le nombre de valeur prises par le tag et elle détermine si c'est une valeur ou un offset qui suit. (Annexe 7)

Les trois variables externes de la fonction sont : les données, la position du tag dans les données et le type de fichier (LE ou BE). Voici l'organigramme :

- On se place d'abord au début du tag
- Selon qu'il s'agisse d'un LE ou d'un BE
- On rentre dans un tableau l'identifiant du tag
- Puis le type de valeur
- On convertit alors ces nombres en entier

- Puis, selon qu'il s'agisse d'un BE ou d'un LE
- On rentre à la suite du même tableau le nombre de valeur prises par le tag
- On convertit alors le résultat en entier

- Ensuite suivant la valeur prise par le type du tag
- On attribue à la variable *taille_type* la taille du type du tag

- Il faut maintenant déterminer si c'est une valeur ou un offset qui suit
- On calcule dans *taille_val* la place nécessaire pour les données du tag
- Si cette valeur est inférieure ou égale à 4
- On initialise la taille du tag à 12
- Selon qu'il s'agisse d'un BE ou d'un LE
- On ajoute au tableau la valeur du tag
- Fin si

- Si cette valeur est supérieure à 4
- On initialise la taille du tag à 12 plus sa valeur
- On convertit en entier la valeur de l'offset
- Sin si

On retourne à la fin la taille du tag

fonction *chargefichier

- **Cette fonction permet de stocker le contenu d'un fichier dans un tableau. Elle sera nécessaire lorsque l'on voudra copier les informations d'un fichier à un autre. (Annexe 5)**

On déclare le fichier f et les autres variables (b)
Tant que l'on est dans le fichier f
    On met le caractère courant dans le tableau b
    Puis on incrémente la valeur du pointeur sur le tableau
On ferme le fichier
On retourne la valeur du tableau

fonction creation

- **Cette fonction a pour simple but d'ouvrir un fichier temporaire dans lequel seront copiées les images des fichiers TIFF à concaténer. Il faut noter que, une fois toutes les copies effectuer, il faudra renommer ce fichier avec le nom prédéfini dans la ligne de commande. (Annexe 3)**

on déclare un tableau contenant le texte « temp.tif »
on déclare le fichier temp
on ouvre alors le fichier temp disponible seulement en écriture

fonction header

- **Cette fonction, une fois la ligne de commande lue, s'intéresse à l'header du fichier tiff ouvert et prêt à être concaténer. Elle détermine l'offset de l'IFD 0 et vérifie le format de l'image.(Annexe 8)**

Déclaration des variables
Si le fichier est vide
    On écrit « c'est un échec, il est vide »
    Puis on sort
Fin si
De la même façon que la fonction *chargefichier*, on charge l'hearder dans un tableau a
Selon qu'il s'agisse d'un LE ou un BE
    On rentre dans le tableau offset l'adresse de l'IFD 0
On écrit cet offset
Si les deux premiers octets du fichier sont 2A 00
    On écrit « c'est bien un fichier TIFF »
Fin si
Si les deux premiers octets du fichier ne sont pas 2A 00
    On écrit « il ne s'agit pas d'un fichier TIFF »
Fin si Sinon on retourne 3

Conclusion

Le programme n'est pas fini, les fonctions supplémentaires à ajouter sont : lecture du nombre d'IFD, lecture du nombre de TAG, création d'un fichier de destination, copie des informations dans le fichier destination.

Ce projet nous a permis de nous améliorer en programmation C malgré le peu d'heure consacré à cette matière qui en demande bien plus.

Ce qu'il reste à faire

Voici les fonctions qui manquent encore à notre projet, sur lesquelles nous avons réfléchi mais malheureusement nous n'avons pas eu assez de temps pour les concrétiser.

Dans « *tiff_analyse* » tout d'abord, nous aurions fait une **fonction permettant de récupérer les noms des différents fichiers** en vu de leur ouverture ultérieure. Pour cela nous les aurions stockés dans un tableau nommé « *nomdefichier* » dans lequel chaque nom aurait été séparé par un espace.

Ceci nous aurait permis ensuite de récupérer le dernier « *nomdefichier* » (en partant de la fin du tableau, décrémentant et s'arrêtant au premier espace rencontré). Celui-ci aurait été stocké dans une variable « *dernierfichier* ».

Cette dernière nous aurait servi à remplacer le nom du fichier temporaire de stockage final des données à la toute fin de la lecture de la commande.

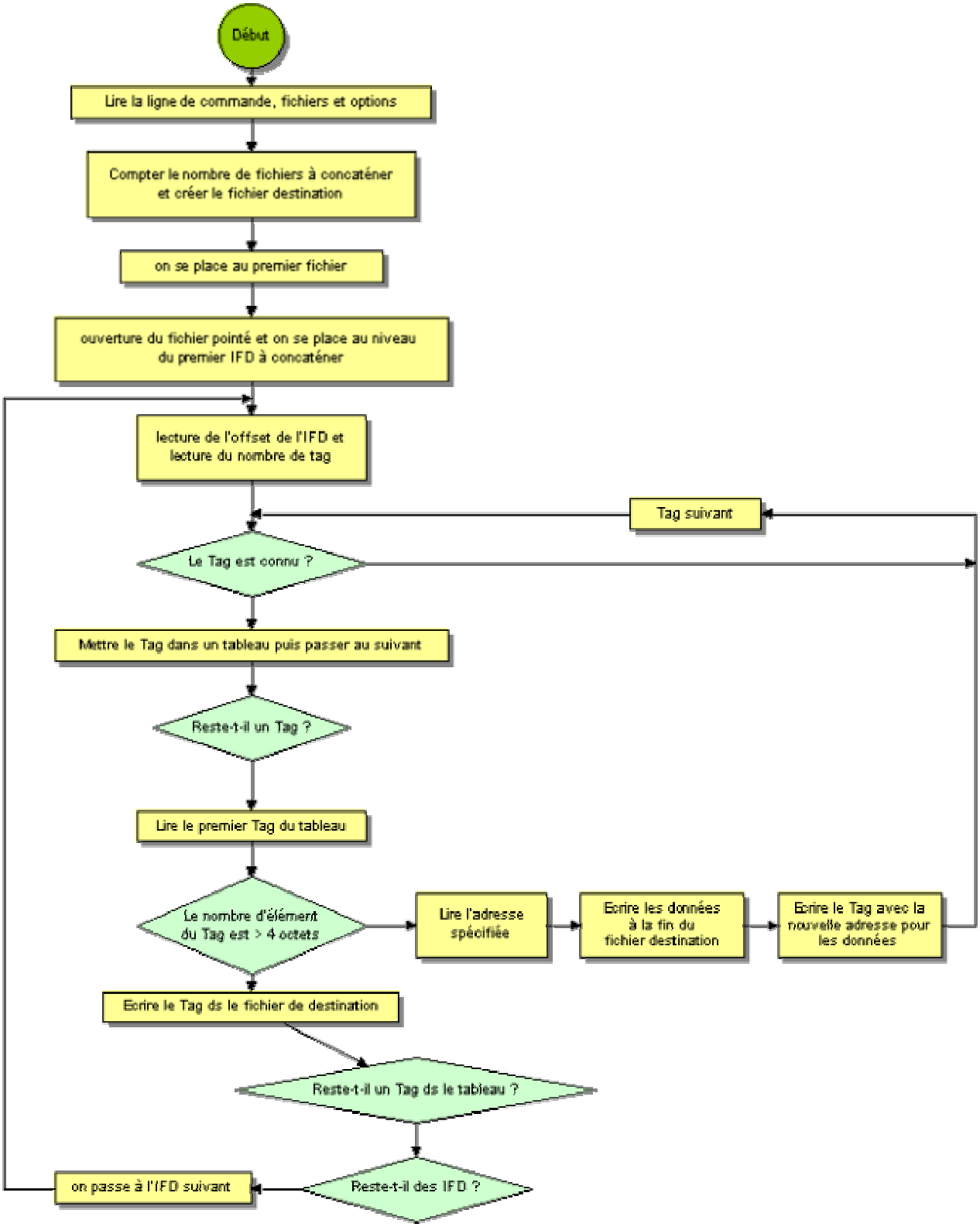
Nous aurions également construit une **procédure « *copie* »** à deux paramètres : x et y qui aurait recopié le contenu d'un tableau « a » de l'adresse x à l'adresse y à la suite du contenu de « temp.tif ».

Enfin nous avions pensé à une procédure qui trouverait y tel que :

adresse de y = adresse de l'offset – 1

Annexes

Annexe 1


Début
Lire la ligne de commande, fichiers et options
Compter le nombre de fichiers à concaténer et créer le fichier destination
on se place au premier fichier
ouverture du fichier pointé et on se place au niveau du premier IFD à concaténer
lecture de l'offset de l'IFD et lecture du nombre de tag
Tag suivant
Le Tag est connu ?
Mettre le Tag dans un tableau puis passer au suivant
Reste-t-il un Tag ?
Lire le premier Tag du tableau
Le nombre d'élément du Tag est > 4 octets
Lire l'adresse spécifiée
Ecrire les données à la fin du fichier destination
Ecrire le Tag avec la nouvelle adresse pour les données
Ecrire le Tag ds le fichier de destination
Reste-t-il un Tag ds le tableau ?
Reste-t-il des IFD ?
on passe à l'IFD suivant

Annexe 2

Annexe 3

Annexe 4

Annexe 5

Annexe 6

Annexe 7

Annexe 8

Annexe 9