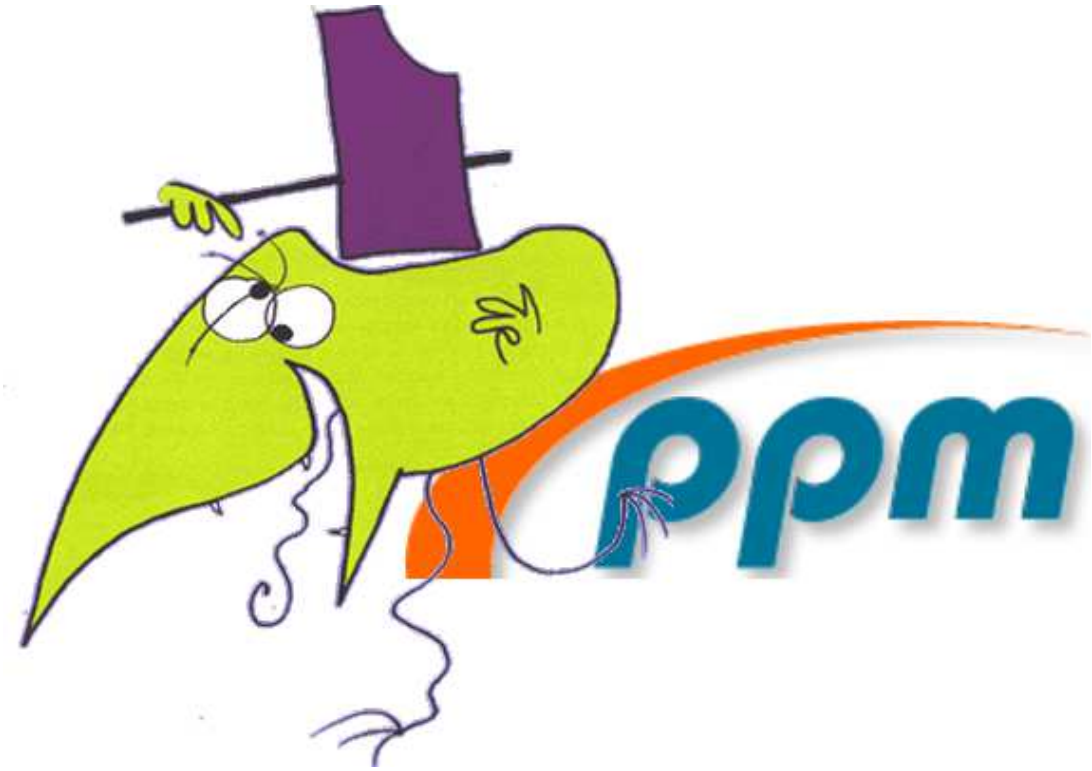
Niveau 4

TP
S4. Développement logiciel
S4.1. Principes de base : Gestion mémoire, allocation dynamique
Flux d'entrée et de sortie de base : fichiers
S4.3. Structure et gestion des données : Formats de fichier : texte, binaire, multimédia

Dévelop'Two

TABLE DES MATIÈRES



1 - PRÉSENTATION

1.1 - IMAGES PPM

Dans ce TP, dont le but est de gérer des fichiers au format texte, nous travaillerons sur des images codées en ASCII. Elles sont les plus simples à utiliser, car elles sont supportées par un fichier texte, directement lisible par le développeur et ne nécessitent donc pas de dé/compression pour les lire et/ou écrire.

Ce format est le PPM (***Portable PixMap***), et le contenu du fichier ressemble à :

```
P3
# Created by IrfanView
8 8
255
255 255 255 255 255 255 255 0 0 255 0 0 255 255 255
255 0 0 255 0 0 255 255 255 255 255 255 255 0 0
255 255 255 255 0 0 255 0 0 255 0 0 255 0 0
…
```

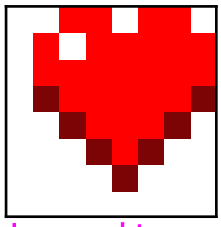


Image obtenue

- La première ligne est le codage du fichier. `P3` signifie que chaque pixel est codé par une couleur RVB.
- La seconde, débutant par `#`, est un commentaire indiquant généralement l'outil utilisé pour la création de l'image. Le nombre de caractères de cette ligne n'est pas fixe.
- La troisième ligne indique les dimensions (largeur puis hauteur) de l'image, exprimées en pixel. On travaille sur une image `8` × `8` pixels.
- La quatrième ligne indique la valeur maximale à utiliser pour chaque composante de couleur RVB (valeur comprise entre 0 et `255` dans cet exemple).

Les lignes suivantes sont une succession des valeurs de chaque pixel, codées selon le format indiqué en ligne 1. Ici le premier pixel est de couleur R=255, V=255, B=255. Le second pixel a la même couleur. Le troisième est en R=255, V=0, B=0…

> Notez qu'aucune ligne de ce fichier ne peut excéder 70 caractères !

2 - TRAVAIL À RÉALISER

2.1 - OUVERTURE ET LECTURE DE L'EN-TÊTE

Madame Entaiteux doit trouver un logo couleur pour sa nouvelle société Dévelop' Two. Elle dispose d'une flopée de fichiers image mais n'a pas le temps de toutes les visualiser une par une.

Elle vous demande d'automatiser certaines étapes afin de lui faciliter la tâche et de lui éviter une perte de temps inutile.

1) Commencez par vérifier que ce sont bien des images en couleur. Préparez-lui une fonction `EstPPM_RVB()` qui reçoit en paramètre le nom du fichier et qui retourne une valeur `vraie` si le fichier est bien au format **PPM, RVB 255.**
Voici le prototype à respecter :
```
bool EstPPM_RVB(char * sNomImage) ;
```

2) Le seul impératif que s'est donné Marthelle, est la taille de ce logo : 8 × 8 pixels ! Elle vous demande de lui fabriquer une procédure nommée `LirePPM_Dimensions()`, qui reçoit en paramètre le nom du fichier à ouvrir et qui lui retourne les dimensions `wHauteur` et `wLargeur` de l'image. Bien sûr, il faudra vérifier que le format de l'image est bien celui attendu !
Voici le prototype à respecter :
```
void LirePPM_Dimensions(    char *              sNomImage,
                            unsigned int &      wHauteur,
                            unsigned int &      wLargeur) ;
```

3) Martelle vous demande, par la suite, de lui fournir une autre fonction, `EstDeTaille()` prenant le nom du fichier en paramètre ainsi que les dimensions `wLargeur` et `wHauteur` qu'elle recherche. Ce module retourne `vrai` si les dimensions de l'image sont identiques à celles requises, et `faux` sinon.
Voici le prototype à respecter :
```
bool     EstDeTaille    (    char *          sNomImage,
                             unsigned int    wHauteur,
                             unsigned int    wLargeur) ;
```

2.2 - LECTURE DE L'IMAGE

M^me^ Entaiteux vous demande de lui présenter un aperçu de l'image. Mais pour ça, une petite préparation est nécessaire : la stocker dans votre programme !

> Reprenez le support de cours et le TP sur les tableaux créés dynamiquement !

4) Commencez par écrire un module `CreerTableau2D()` qui va créer **dynamiquement** le tableau `tabLogo` (déclaré dans le `main` sous forme de double pointeur), ayant deux dimensions de taille adaptée aux `wLargeur` et `wHauteur`, passées aussi en paramètre. La fonction utilise une variable locale, `wTab`, double pointeur elle aussi, lui permettant la création de ce tableau. Une fois alloué à `wTab`, le tableau est initialisé et **retourné** au `main`. Le type de retour est donc un double pointeur.
Voici le prototype à respecter :
```
unsigned int * * CreerTableau2D(unsigned int wHauteur, unsigned int wLargeur) ;
```

5) Écrivez une fonction `RVB()` retournant une valeur `wRvb` unique à partir des trois composantes `wR`, `wV` et `wB`, passées en paramètres. Ces couleurs sont de type `unsigned int`.
Voici le prototype à respecter :
```
unsigned int RVB(unsigned int wR, unsigned int wV, unsigned int wB) ;
```

6) Écrivez un module `LireImagePPM()` qui reçoit en paramètre le nom du fichier image à charger et le double pointeur `wTab`. Ce module remplit le tableau avec les couleurs RVB reconstituées d'après les valeurs R, V et B, lues dans le fichier.
Voici le prototype à respecter :
```
void     LireImagePPM( char *              sNomImage,
                       unsigned int * *    wTab,
                       unsigned int        wHauteur,
                       unsigned int        wLargeur) ;
```

7) Il vous est maintenant possible d'afficher votre image, en affichant le contenu du tableau wTab. Pour ça, vous pourrez définir des constantes symboliques pour chaque couleur et afficher un caractère adapté à chacune, comme montré dans la soluce d'un TP précédent.
Par exemple, pour le rouge :

```
#define ROUGE     (0x00FF0000)
```

et pour l'affichage :

```
if(Tab[x][y] == ROUGE)      cout << "R" ;
```

Voici le prototype à respecter :

```
void AfficherImage(    unsigned int * *   wTab,
                       unsigned int       wHauteur,
                       unsigned int       wLargeur) ;
```

8) Proposez un module logiciel, LibererTab2D, permettant de dés-allouer correctement le tableau créé dynamiquement à la question 4. Cette fonction prend en paramètre de double pointeur du tableau, ainsi que les tailles de ses deux dimensions wHauteur et wLargeur. Une fois dés-alloué, le double pointeur est retourné au main.
Voici le prototype à respecter :

```
unsigned int * * LibererTab2D(  unsigned int **    wTab,
                                unsigned int       wHauteur,
                                unsigned int       wLargeur) ;
```

3 - BONUS

9) Réaliser un module permettant de trafiquer les couleurs de l'image. La valeur de chaque composante R, V et B est intervertie. Par exemple : le R prend celle du B, le B celle du V et le V celle du R...

10) Créez un module permettant de sauvegarder cette image truquée dans un nouveau fichier au format **PPM RBV 255**.

4 - COULEURS UTILISÉES

Couleur	*Hexadécimal*	*Décimal*
Blanc	0x00FFFFFF	255 255 255
Gris	0x00AAAAAA	170 170 170
Marron foncé	0x007D0404	125 4 4
Marron clair	0x009B4A00	155 74 0
Rouge	0x00FF0000	255 0 0
Rose	0x00FFA987	255 169 135

Couleur	*Hexadécimal*	*Décimal*
Orange	0x00FF7300	255 115 0
Jaune	0x00FFFF00	255 255 0
Cyan	0x0000D1FF	0 209 255
Bleu	0x000000FF	0 0 255
Bleu foncé	0x00000071	0 0 113
Violet	0x00BE00BE	190 0 190