[Math-L312] TP 9 : Complexifions les structures et structurons les complexités

Adrien Semin
adrien.semin@math.u-psud.fr

1 Notation pour ce TP

Aucun retard ne sera toléré pour ce TP. Tout étudiant qui enverra son TP après mardi soir minuit aura 0 à ce TP.

2 Nouvelle bibliothèque

La bibliothèque `image_ppm.h` a été remaniée de manière à pouvoir tester si l'image passée en argument a un nom correct. Des structures d'image ont également été rajoutées :

```
typedef struct {
  OCTET rouge;
  OCTET vert;
  OCTET bleu;
} pixel;

typedef struct
{
  int nH;
  int nW;
  pixel* tpixel;
} imageppm;

typedef struct
{
  int nH;
  int nW;
  OCTET* tpixel;
} imageppm;
```

Le programme suivant montre un exemple d'utilisation de cette structure, téléchargeable à l'adresse `/home/doc/semin/TP/c/s_image_copy.c`.

```
#include <stdio.h>
#include <image_ppm.h>
#include <allocation_tableaux.h>
#include <assert.h>

int main(int argc, char* argv[])
{

  char cNomImgLue[250], cNomImgEcrite[250];
  int nH, nW, nTaille, i, j;

  if (argc != 3)
    {
      printf("Usage: %s ImageIn.ppm ImageOut.ppm\n",argv[0]);
      exit(1) ;
    }
```

```
  assert(sscanf(argv[1],"%s",cNomImgLue)==1);
  assert(sscanf(argv[2],"%s",cNomImgEcrite)==1);
  imageppm* PictureIn = malloc(sizeof(imageppm));
  s_lire_image_ppm(cNomImgLue, PictureIn);
  imageppm* PictureOut = malloc(sizeof(imageppm));
  nouvelle_image_ppm(PictureOut, PictureIn->nH, PictureIn->nW);
  for(i=0;i<PictureIn->nH;i++)
    {
    for(j=0;j<PictureIn->nW;j++)
      {
      PictureOut->tpixel[PictureIn->nW*i+j]
        = PictureIn->tpixel[PictureIn->nW*i+j];
      }
    }
  s_ecrire_image_ppm(cNomImgEcrite, PictureOut);
  desallouer_ppm(PictureIn);
  desallouer_ppm(PictureOut);
  return 0;
}
```

`* **Question orale** *` Pourquoi initialise-t-on le pointeur `imageppm*` avec une instruction `malloc()` ?

`* **Question orale** *` Dans le programme, quelle est la différence entre `imageppm*` et `OCTET*` au niveau de l'allocation ?

3 Travail demandé

1. `fraction.c` `* **À Rendre!** *` A partir de la structure suivante

```
typedef struct {
  int num;
  unsigned int denom;
} fraction
```

implémenter dans un fichier `fraction.c` les fonctions suivantes (on donne uniquement leur prototypage) :

```
// Tester si une fraction est irréductible.
// La fonction renverra -1 si oui, et 0 sinon.
int is_irreductible(fraction)
// Additionne deux fractions, et retourne une fraction sous forme
// irréductible.
fraction add(fraction, fraction)
// Soustrait deux fractions, et retourne une fraction sous forme
// irréductible.
fraction sub(fraction, fraction)
// Multiplie deux fractions, et retourne une fraction sous forme
// irréductible.
fraction mult(fraction, fraction)
// Divise deux fractions, er retourne une fraction sous forme
// irréductible.
fraction div(fraction, fraction)
// Evalue une fraction
double eval(fraction)
```

Note : on pourra utiliser la fonction suivante

```
int pgcd(int a,int b)
{
  if (b==0) return a;
```

```
    return pgcd(b,a\%b);
}
```

2. `plans_rvb.c` **** À Rendre! **** Ecrire un programme `plans_rvb.c` qui prendra à l'exécution un nom d'image d'entrée, et construira trois images dont le nom sera construit à partir du nom de l'image d'entrée (exemple : si le nom d'image d'entrée est `lena.ppm`, les trois images de sortie créées seront `lena_R.ppm`, `lena_V.ppm` et `lena_B.ppm`). Ces images contiendront respectivement les plans rouge, vert et bleu de l'image d'entrée.
   Par exemple, pour l'image d'entrée `lena.ppm` (voir figure 1), le programme donnera les images correspondant aux figures 2 à 4.

FIG. 1 – Image initiale : `lena.ppm`

FIG. 2 – Plan rouge : `lena_R.ppm`

FIG. 3 – Plan vert : `lena_V.ppm`

FIG. 4 – Plan bleu : `lena_B.ppm`

**** Question orale **** A quoi peuvent correspondre ces plans ?