Compte rendu Tache 2

Dans ce compte rendu on a inclus les fichier de la tache 2 qui inclut aussi le code correspondant en tache 1 comme il était demandé de faire un programme de test. Les résultats de test sont affichés sur le terminal.

Fichiers .h

Geom2d.h

```c
#ifndef _GEOM2D_H_
#define _GEOM2D_H_

#include "types_macros.h"

//Type Vecteur
typedef struct Vecteur_
{
   double x,y;
} Vecteur;

//Type Point
typedef struct Point_
{
   double x,y;
} Point;

Point set_point(double x, double y);

Vecteur set_vecteur(double x, double y);

Point add_point(Point P1, Point P2);

Vecteur add_vecteur(Vecteur V1, Vecteur V2);

Vecteur vect_bipoint(Point A, Point B);

Point produit(Point P1, double a);

Vecteur produit_vect(Vecteur V1, double a);

double produit_scalaire(Vecteur V1, Vecteur V2);

double norme(Vecteur V1);

double distance(Point P1, Point P2);

#endif /* _GEOM2D_H_ */
```

Image.h

```c
/*****************************************************************************
  Interface du module image
```

```
*****************************************************************************/

#ifndef _IMAGE_H_
#define _IMAGE_H_

#include "types_macros.h"


/*
 Type enum@r@ Pixel @quivalent au char avec BLANC=0 et NOIR=1
 */
typedef enum {BLANC=0,NOIR=1} Pixel;

/*
 Type Image
 */
typedef struct Image_
{
   UINT la_largeur_de_l_image;
   UINT la_hauteur_de_l_image;
   Pixel* pointeur_vers_le_tableau_de_pixels;
} Image;

/* cr@ation d'une image PBM de dimensions L x H avec tous les pixels blancs */
Image creer_image(UINT L, UINT H);

/* suppression de l'image I = *p_I */
void supprimer_image(Image *p_I);

/* renvoie la valeur du pixel (x,y) de l'image I
   si (x,y) est hors de l'image la fonction renvoie BLANC */
Pixel get_pixel_image(Image I, int x, int y);

/* change la valeur du pixel (x,y) de l'image I avec la valeur v
   si (x,y) est hors de l'image la fonction ne fait rien */
void set_pixel_image(Image I, int x, int y, Pixel v);

/* renvoie la largeur de l'image I */
UINT largeur_image(Image I);

/* renvoie la hauteur de l'image I */
UINT hauteur_image(Image I);

/* lire l'image dans le fichier nomm@ nom_f
   s'il y a une erreur dans le fichier le programme s'arrete en affichant
   un message
   version acceptant les fichiers avec
   - ligne 1 : P1
   - zero, une ou plusieurs lignes commen@ant toutes par #
   - zero, un ou plusieurs s@parateurs
   - la largeur
   - un ou plusieurs s@parateurs
   - la hauteur
   - un ou plusieurs s@parateurs
```

```
   - les pixels de l'image
   */
Image lire_fichier_image(char *nom_f);

/* crire l'image I  l'cran */
void ecrire_image(Image I);

/* calculer l'image "negatif" de l'image I */
/* l'image I n'est pas modifiee et */
/* la fonction renvoie l'image "negatif" de I */
Image negatif_image(Image I);

#endif /* _IMAGE_H_ */
```

Fichiers .c

Geom2d.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>

#include "geom2d.h"

Point set_point(double x, double y)
{
    Point P = {x,y};
    return P;
}

Vecteur set_vecteur(double x, double y)
{
    Vecteur V = {x,y};
    return V;
}

Point add_point(Point P1, Point P2)
{
    return set_point(P1.x+P2.x, P1.y+P2.y);
}

Vecteur add_vecteur(Vecteur V1, Vecteur V2)
{
    return set_vecteur(V1.x+V2.x, V1.y+V2.y);
}

Vecteur vect_bipoint(Point A, Point B)
{
    Vecteur V = {B.x-A.x, B.y-A.y};
    return V;
}
```

```
Point produit(Point P1, double a)
{
    return set_point(a*P1.x, a*P1.y);
}

Vecteur produit_vect(Vecteur V1, double a)
{
    return set_vecteur(a*V1.x, a*V1.y);
}

double produit_scalaire(Vecteur V1, Vecteur V2)
{
    return (V1.x*V2.x + V1.y*V2.y);
}

double norme(Vecteur V1)
{
    return (sqrt(V1.x*V1.x + V1.y*V1.y ));
}

double distance(Point P1, Point P2)
{
    return(sqrt((P1.x-P2.x)*(P1.x-P2.x) + (P1.y-P2.y)*(P1.y-P2.y)));
}
```

Image.c

```
/*************************************************************************
  Implementation du module image_pbm
*************************************************************************/

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include"types_macros.h"
#include"image.h"

/* macro donnant l'indice du pixel de coordonnees (_x,_y) de l'image _I
   dans le tableau de pixels de l'image _I */
#define INDICE_PIXEL(_I,_x,_y) ((_x)-1)+(_I).la_largeur_de_l_image*((_y)-1)

/* creation d'une image PBM de dimensions L x H avec tous les pixels blancs */
Image creer_image(UINT L, UINT H)
{
    Image I;
    UINT i;

    I.la_largeur_de_l_image = L;
    I.la_hauteur_de_l_image = H;

    /* allocation dynamique d'un tableau de L*H Pixel*/
```

```
    I.pointeur_vers_le_tableau_de_pixels = (Pixel *)malloc(sizeof(Pixel)*L*H);

    /* test si le tableau a ete correctement alloue */
    if (I.pointeur_vers_le_tableau_de_pixels == (Pixel *)NULL)
    {
        ERREUR_FATALE("Impossible de creer une image");
    }

    /* remplir le tableau avec des pixels blancs */
    for (i=0; i<L*H; i++)
        I.pointeur_vers_le_tableau_de_pixels[i] = BLANC;

    return I;
}

/* suppression de l'image I = *p_I */
void supprimer_image(Image *p_I)
{
    free(p_I->pointeur_vers_le_tableau_de_pixels);
    p_I->la_largeur_de_l_image = 0;
    p_I->la_hauteur_de_l_image = 0;
}

/* renvoie la valeur du pixel (x,y) de l'image I
   si (x,y) est hors de l'image la fonction renvoie BLANC */
Pixel get_pixel_image(Image I, int x, int y)
{
    if (x<1 || x>I.la_largeur_de_l_image || y<1 || y>I.la_hauteur_de_l_image)
        return BLANC;
    return I.pointeur_vers_le_tableau_de_pixels[INDICE_PIXEL(I,x,y)];
}

/* change la valeur du pixel (x,y) de l'image I avec la valeur v
   si (x,y) est hors de l'image la fonction ne fait rien */
void set_pixel_image(Image I, int x, int y, Pixel v)
{
    if (x<1 || x>I.la_largeur_de_l_image || y<1 || y>I.la_hauteur_de_l_image)
        return;
    I.pointeur_vers_le_tableau_de_pixels[INDICE_PIXEL(I,x,y)] = v;
}

/* renvoie la largeur de l'image I */
UINT largeur_image(Image I)
{
    return I.la_largeur_de_l_image;
}

/* renvoie la hauteur de l'image I */
UINT hauteur_image(Image I)
{
    return I.la_hauteur_de_l_image;
}
```

```c
/* lire l'image dans le fichier nomme nom_f
   s'il y a une erreur dans le fichier le programme s'arrete en affichant
   un message
   version acceptant les fichiers avec
   - ligne 1 : P1
   - zero, une ou plusieurs lignes commen?ant toutes par #
   - zero, un ou plusieurs separateurs
   - la largeur
   - un ou plusieurs separateurs
   - la hauteur
   - un ou plusieurs separateurs
   - les pixels de l'image
   */

/* teste si le fichier d'identificateur f debute par un en-tete
   valide pour un fichier PBM :
   - ligne 1 : P1
   - suivie de zero, une ou plusieurs lignes commen?ant toutes par #
   La fonction se termine correctement si le fichier est correct,
   et le pointeur de fichier se trouve ? la suite de l'entete.
   Sinon, l'execution du programme est arretee avec l'affichage d'un message
   d'erreur (appel ? ERREUR_FATALE)
    */
void entete_fichier_pbm(FILE *f)
{
    char *ligne;
    size_t n;
    size_t l_ligne;

    /* se positionner en debut de fichier */
    fseek(f, 0, SEEK_SET);

    /* lecture et test de la ligne 1 */
    ligne = (char *)NULL;
    n = 0;
    l_ligne = getline(&ligne, &n, f);

    /* la ligne est correcte si et ssi
       cas - fichier cree sous Linux : ligne = {'P','1',10}
         soit une chaine de 3 caracteres (le dernier est le caractere nul)
       cas - fichier cree sous Windows : ligne = {'P','1',13, 10}
         soit une chaine de 4 caracteres (le dernier est le caractere nul)
       */
    if (l_ligne < 3)
    {
        ERREUR_FATALE("entete_fichier_pbm : ligne 1 incorrecte\n");
    }
    if ((ligne[0] != 'P') || (ligne[1] != '1'))
    {
        ERREUR_FATALE("entete_fichier_pbm : ligne 1 incorrecte\n");
    }
    free(ligne);
```

```
    /* lecture des eventuelles lignes commen@ant par # */
    bool boucle_ligne_commentaire = true;
    do
    {
        /* tester d'abord la fin de fichier */
        if (feof(f))
        {
            ERREUR_FATALE("entete_fichier_pbm : fin fichier inattendue\n");
        }

        /* lire un caractere et tester par rapport @ '#' */
        char c;
        fscanf(f, "%c", &c);
        if (c=='#')
        {
            /* lire le reste de la ligne */
            ligne = (char *)NULL;
            n = 0;
            l_ligne = getline(&ligne, &n, f);
            free(ligne);
        }
        else
        {
            /* reculer d'un caract@re dans f */
            fseek(f, -1, SEEK_CUR);
            boucle_ligne_commentaire = false;
        }
    } while (boucle_ligne_commentaire);

}

/* lire l'image dans le fichier nomme nom_f
   s'il y a une erreur dans le fichier le programme s'arrete en affichant
   un message */
Image lire_fichier_image(char *nom_f)
{
    FILE *f;
    UINT L,H;
    UINT x,y;
    int res_fscanf;
    Image I;

    /* ouverture du fichier nom_f en lecture */
    f = fopen(nom_f, "r");
    if (f == (FILE *)NULL)
    {
        ERREUR_FATALE("lire_fichier_image : ouverture du fichier impossible\n");
    }

    /* traitement de l'en-tete et arret en cas d'erreur */
    entete_fichier_pbm(f);

    /* lecture des dimensions */
```

```
    res_fscanf = fscanf(f, "%d", &L);
    if (res_fscanf != 1)
    {
        ERREUR_FATALE("lire_fichier_image : dimension L incorrecte\n");
    }
    res_fscanf = fscanf(f, "%d", &H);
    if (res_fscanf != 1)
    {
        ERREUR_FATALE("lire_fichier_image : dimension H incorrecte\n");
    }

    /* creation de l'image de dimensions L x H */
    I = creer_image(L,H);

    /* lecture des pixels du fichier
       seuls les caracteres '0' (BLANC) ou '1' (NOIR)
       doivent etre pris en compte */
    x = 1; y = 1;
    while (!feof(f) && y<=H)
    {
        char c;
        int res;

        /* lire un caractere en passant les caracteres differents de '0' et '1' */
        res = fscanf(f, "%c", &c);
        while (!feof(f) && !(c == '0' || c == '1'))
        {
            res = fscanf(f, "%c", &c);
        }
        if (!feof(f))
        {
            set_pixel_image(I,x,y,c=='1' ? NOIR : BLANC );
            x++;
            if (x>L)
            {
                x = 1; y++;
            }
        }
    }

    /* fermeture du fichier */
    fclose(f);

    return I;
}

/* ecrire l'image I  l'ecran */
void ecrire_image(Image I)
{
    int hauteur = I.la_hauteur_de_l_image;
    int largeur = I.la_largeur_de_l_image;
    Pixel A;
    int id=0;
```

```
    for (int i=1; i<=hauteur; i++)
    {
        for (int j=1;j<=largeur; j++)
        {
        A = I.pointeur_vers_le_tableau_de_pixels[id];
        switch(A)
            {
                case(BLANC):
                    printf("0");
                    break;
                case(NOIR):
                    printf("1");
                    break;
            }

            id++;
        }
        printf("\n");
    }
}

/* calculer l'image "negatif" de l'image I */
/* l'image I n'est pas modifiee et */
/* la fonction renvoie l'image "negatif" de I */

Image negatif_image(Image I)
{
    int hauteur = I.la_hauteur_de_l_image;
    int largeur = I.la_largeur_de_l_image;
    Pixel A;
    int id=0;

    //Initialisation image neagtive
    Image negative;
    negative = creer_image(largeur, hauteur);


    for (int i=1; i<=hauteur; i++)
    {
        for (int j=1;j<=largeur; j++)
        {
        A = I.pointeur_vers_le_tableau_de_pixels[id];
        switch(A)
            {
                case(BLANC):
                    set_pixel_image(negative, j, i, NOIR);
                    break;
                case(NOIR):
                    set_pixel_image(negative, j, i, BLANC);
                    break;
            }

            id++;
        }
```

```
    }
    return negative;
}
```

Test_image.c

```
#include <stdint.h>
#include <string.h>
#include<stdlib.h>
#include"types_macros.h"
#include"image.h"

int main(int argc, char **argv)
{
    //Test no 1
    printf("Test 1\n");
    Image I;
    char fichier[100];
    strcpy(fichier, argv[1]);
    I = lire_fichier_image(fichier);
    ecrire_image(I);

    printf("\n");
    //Test no 2
    Image neg;
    printf("Test 2\n");
    neg = negatif_image(I);
    ecrire_image(neg);

    return 0;
}
```

Test_geom2d.c

```
#include <stdint.h>
#include <string.h>
#include<stdlib.h>

#include "geom2d.h"

int main(int argc, char **argv)
{
    //Test no 3
    Point A, B, C;
    A = set_point(1.0, -3.0);
    B = set_point(4.0, 1.0);
    C = add_point(A, B);
    printf("%f %f \n", C.x, C.y);

    Vecteur V, I, Z;
    V = set_vecteur(1.0, 2.0);
    I = set_vecteur(0.0, 5.0);
    Z = add_vecteur(V, I);
    printf("%f %f \n", Z.x, Z.y);
```

```
    double a = 3;
    Z = produit_vect(V, a);
    printf("%f %f \n", Z.x, Z.y);

    C = produit(A, a);
    printf("%f %f \n", C.x, C.y);

    double res;
    res = produit_scalaire(V, I);
    printf("%f \n", res);

    res = norme(I);
    printf("%f \n", res);

    res = distance(A, B);
    printf("%f \n", res);
}
```

Resultats des tests

Géométrie 2D

Les points sont initialisés à :

```
A = (1.0, -3.0)
B = (4.0, 1.0)
```

Et les vecteurs utilises sont initialisés à:

```
V = (1.0, 2.0)
I = (0.0, 5.0)
```

Les resultats sont correctes:

```
skarleav@im2ag-turing:[~/MAP401/S05]: ./test_geom
5.000000 -2.000000    test add_point
1.000000 7.000000     test add_vecteur
3.000000 6.000000     test produit_vecteur
3.000000 -9.000000    test produit point avec un reel
10.000000             test produit_scalaire
5.000000              test norme
5.000000              test distance de deux points
```

Image

L'image négatif est bien enregistrée à une autre image que l'originale et on utilise la fonction ecrire_image pour afficher le résultat sur le terminal

```
skarleav@im2ag-turing:[~/MAP401/S04-Tache1]: ./test_image caractere2.pbm
Test 1
0000000
0011100
0100010
0000010
0000100
0001000
0010000
0111110
0000000

Test 2
1111111
1100011
1011101
1111101
1111011
1110111
1101111
1000001
1111111
```

test affichage image .pbm

test affichage image negatif .pbm