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**

#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**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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**

#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;

}

/\* 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)

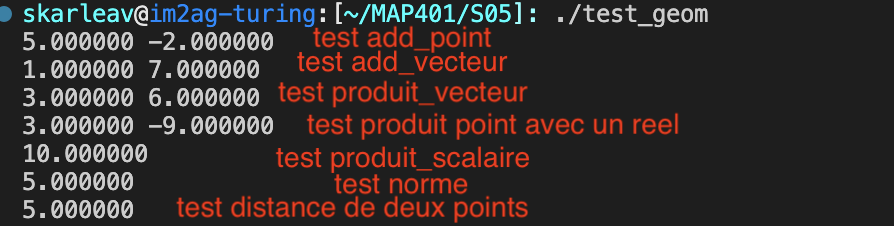
B = (4.0, 1.0)

Et les vecteurs utilises sont initialisés à:

V = (1.0, 2.0)

I = (0.0, 5.0)

Les resultats sont correctes:



**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

