Projet MAP401

Lord Pascal Isak & Weber Loïc

Table of Content

Paquetage geometrie

geometrie.h

```c
/*****************************************************************************
  implementation module geometrie 2D
*****************************************************************************/

#ifndef _GEOMETRIE_H_
#define _GEOMETRIE_H_



typedef struct {
    double x;
    double y;
} Vecteur;

typedef struct {
    double x;
    double y;
} Point;

////////////////////////////////////////////////
//                Initialisation              //
////////////////////////////////////////////////
// crée un nouveau vecteur de coordonée (x,y):  V(x,y)
Vecteur nouveau_vecteur(double x, double y);
// crée un nouveau point de coordonée (x,y):  P(x,y)
Point nouveau_point(double x, double y);
////////////////////////////////////////////////
//                 Conversion                 //
```

```
////////////////////////////////////////////////

//convertie le point P(x,y) en le vecteur V(x,y)
Vecteur point_to_vecteur(Point p);

//convertie le vecteur V(x,y) en le point P(x,y)
Point vecteur_to_point(Vecteur v);

//crée le vecteur allant du point p1 au point p2
Vecteur couple_point_to_vecteur(Point p1, Point p2);

////////////////////////////////////////////////
//                  Operations                //
////////////////////////////////////////////////
//vecteurs
//renvoie v1+v2
Vecteur addition_vecteur(Vecteur v1, Vecteur v2);

//renvoie λ*v
Vecteur produit_vecteur(double λ, Vecteur v);

//renvoie le produit scalaire de v1 et v2
double produit_scalaire(Vecteur v1, Vecteur v2);

//renvoie la norme de v aussi notée |v|
double norme(Vecteur v);
// point
// renvoie p1+p2
Point addition_point(Point p1, Point p2);
// renvoie λ*p
Point produit_point(double λ, Point p);
//renvoie la distance en =tre le point p1 et p2
double distance_point(Point p1, Point p2);


#endif
```

geometrie.c

```
/*****************************************************************************
  implementation module geometrie 2D
*****************************************************************************/

//#include<stdio.h>
//#include<stdlib.h>
//#include<string.h>
#include<math.h>
#include"geometrie.h"
```

```
////////////////////////////////////////////////////
//                Initialisation                  //
////////////////////////////////////////////////////

//crée un nouveau vecteur de coordonée (x,y):  V(x,y)
Vecteur nouveau_vecteur(double x,double y){
    Vecteur v;
    v.x = x;
    v.y = y;
    return v;
}
//crée un nouveau point de coordonée (x,y):  P(x,y)
Point nouveau_point(double x,double y){
    Point p;
    p.x = x;
    p.y = y;
    return p;
}
////////////////////////////////////////////////////
//                  Conversion                    //
////////////////////////////////////////////////////

//convertie le point P(x,y) en le vecteur V(x,y)
Vecteur point_to_vecteur(Point p){
    return nouveau_vecteur(p.x, p.y);
}

//convertie le vecteur V(x,y) en le point P(x,y)
Point vecteur_to_point(Vecteur v){
    return nouveau_point(v.x, v.y);
}

//crée le vecteur allant du point p1 au point p2
Vecteur couple_point_to_vecteur(Point p1, Point p2){
    return nouveau_vecteur(p2.x - p1.x, p2.y - p1.y);
}

////////////////////////////////////////////////////
//                  Operations                    //
////////////////////////////////////////////////////
//vecteurs
//renvoie v1+v2
Vecteur addition_vecteur(Vecteur v1, Vecteur v2){
    return nouveau_vecteur(v1.x + v2.x, v1.y + v2.y);
}

//renvoie λ*v
Vecteur produit_vecteur(double λ, Vecteur v){
    return nouveau_vecteur(λ * v.x, λ * v.y);
}

//renvoie le produit scalaire de v1 et v2
double produit_scalaire(Vecteur v1, Vecteur v2){
```

```
    return v1.x * v2.x + v1.y * v2.y;
}

//renvoie la norme de v aussi notée |v|
double norme(Vecteur v){
    return sqrt(produit_scalaire(v, v));
}
//point
//renvoie p1+p2
Point addition_point(Point p1, Point p2){
    return nouveau_point(p1.x + p2.x, p1.y + p2.y);
}
//renvoie λ*p
Point produit_point(double λ, Point p){
    return nouveau_point(λ * p.x, λ * p.y);
}
//renvoie la distance en =tre le point p1 et p2
double distance_point(Point p1, Point p2){
    return norme(couple_point_to_vecteur(p1, p2));
}
```

test_geometrie.c

```
#include <stdio.h>

#include "geometrie.h"
#include "types_macros.h"

#define ANSI_COLOR_RED     "\x1b[31m"
#define ANSI_COLOR_GREEN   "\x1b[32m"
#define RESET_COLOR        "\x1B[0m"

//foncyions pratiques:
void afficher_resultat_test(int b){
    if (b){
        printf(" %s SUCCESS%s\n", ANSI_COLOR_GREEN, RESET_COLOR);
    }
    else{
        printf(" %s FAILED%s\n", ANSI_COLOR_RED, RESET_COLOR);
    }
}

int egale_vecteur(Vecteur v1, Vecteur v2){
    return v1.x == v2.x && v1.y == v2.y;
}

int egale_point(Point p1, Point p2){
    return p1.x == p2.x && p1.y == p2.y;
}

//fonctions de test
```

```c
void test_nouveau_point(){
    printf("Test fonction nouveau_point\n");

    printf("Test 1/3");
    Point p = nouveau_point(0, 0);
    afficher_resultat_test((p.x == 0 && p.y == 0));

    printf("Test 2/3");
    Point p1 = nouveau_point(5, 27.5);
    Point p2 = nouveau_point(5, 27.5);
    afficher_resultat_test((p1.x == p2.x && p1.y == p2.y));

    printf("Test 3/3");
    p = nouveau_point(-315.648135, 74535.1);
    afficher_resultat_test((p.x == -315.648135 && p.y == 74535.1));
}
void test_nouveau_vecteur(){
    printf("Test fonction nouveau_vecteur\n");

    printf("Test 1/3");
    Vecteur v = nouveau_vecteur(0, 0);
    afficher_resultat_test((v.x == 0 && v.y == 0));

    printf("Test 2/3");
    Vecteur v1 = nouveau_vecteur(5, 27.5);
    Vecteur v2 = nouveau_vecteur(5, 27.5);
    afficher_resultat_test((v1.x == v2.x && v1.y == v2.y));

    printf("Test 3/3");
    v = nouveau_vecteur(-315.648135, 74535.1);
    afficher_resultat_test((v.x == -315.648135 && v.y == 74535.1));
}

void test_point_to_vecteur(){
    printf("Test fonction point_to_vecteur\n");

    printf("Test 1/3");
    Point p = nouveau_point(0, 0);
    Vecteur v = point_to_vecteur(p);
    afficher_resultat_test((v.x == 0 && v.y == 0));

    printf("Test 2/3");
    Point p1 = nouveau_point(5, 27.5);
    Point p2 = nouveau_point(5, 27.5);
    Vecteur v1 = point_to_vecteur(p1);
    Vecteur v2 = point_to_vecteur(p2);
    afficher_resultat_test((v1.x == v2.x && v1.y == v2.y));

    printf("Test 3/3");
    p = nouveau_point(-315.648135, 74535.1);
    v = point_to_vecteur(p);
    afficher_resultat_test((v.x == -315.648135 && v.y == 74535.1));
}
```

```c
void test_vecteur_to_point(){
    printf("Test fonction vecteur_to_point\n");

    printf("Test 1/3");
    Vecteur v = nouveau_vecteur(0, 0);
    Point p = vecteur_to_point(v);
    afficher_resultat_test((p.x == 0 && p.y == 0));

    printf("Test 2/3");
    Vecteur v1 = nouveau_vecteur(5, 27.5);
    Vecteur v2 = nouveau_vecteur(5, 27.5);
    Point p1 = vecteur_to_point(v1);
    Point p2 = vecteur_to_point(v2);
    afficher_resultat_test((p1.x == p2.x && p1.y == p2.y));

    printf("Test 3/3");
    v = nouveau_vecteur(-315.648135, 74535.1);
    p = vecteur_to_point(v);
    afficher_resultat_test((p.x == -315.648135 && p.y == 74535.1));
}

void test_couple_point_to_vecteur(){
    printf("Test fonction couple_point_to_vecteur\n");

    printf("Test 1/3");

afficher_resultat_test(egale_vecteur(couple_point_to_vecteur(nouveau_point(0,0),no
uveau_point(5,5)), nouveau_vecteur(5,5)));


    printf("Test 2/3");

afficher_resultat_test(egale_vecteur(couple_point_to_vecteur(nouveau_point(5,5),no
uveau_point(0,0)), nouveau_vecteur(-5,-5)));


    printf("Test 3/3");

afficher_resultat_test(egale_vecteur(couple_point_to_vecteur(nouveau_point(0,0),no
uveau_point(0.25,2)),
couple_point_to_vecteur(nouveau_point(-0.25,-2),nouveau_point(0,0))));
}

void test_addition_vecteur(){
    printf("Test fonction addition_vecteur\n");

    printf("Test 1/3");

afficher_resultat_test(egale_vecteur(addition_vecteur(nouveau_vecteur(0,0),nouveau
_vecteur(5,5)), nouveau_vecteur(5,5)));
```

```
    printf("Test 2/3");

afficher_resultat_test(egale_vecteur(addition_vecteur(nouveau_vecteur(-5,5),nouvea
u_vecteur(2,-8)), nouveau_vecteur(-3,-3)));


    printf("Test 3/3");// v1+v2 == v2+v1

afficher_resultat_test(egale_vecteur(addition_vecteur(nouveau_vecteur(5,2.5),nouve
au_vecteur(0.25,2)),
addition_vecteur(nouveau_vecteur(0.25,2),nouveau_vecteur(5,2.5))));
}

void test_produit_vecteur(){
    printf("Test fonction produit_vecteur\n");

    printf("Test 1/3");
    afficher_resultat_test(egale_vecteur(produit_vecteur(5,nouveau_vecteur(5,5)),
nouveau_vecteur(5*5,5*5)));


    printf("Test 2/3");

afficher_resultat_test(egale_vecteur(produit_vecteur(-1,nouveau_vecteur(2,-8)),
nouveau_vecteur(-2,8)));


    printf("Test 3/3");

afficher_resultat_test(egale_vecteur(produit_vecteur(0,nouveau_vecteur(64568,2.513
65)), nouveau_vecteur(0,0)));
}

void test_produit_scalaire(){
    printf("Test fonction produit_scalaire\n");

    printf("Test 1/3");

afficher_resultat_test(produit_scalaire(nouveau_vecteur(5,0),nouveau_vecteur(0,784
523)) == 0);


    printf("Test 2/3");

afficher_resultat_test(produit_scalaire(nouveau_vecteur(5,1006),nouveau_vecteur(0,
0)) == 0);


    printf("Test 3/3");

afficher_resultat_test(produit_scalaire(nouveau_vecteur(5,16),nouveau_vecteur(657,
77)) == 4517);
}
```

```c
void test_norme(){
    printf("Test fonction norme\n");

    printf("Test 1/3");
    afficher_resultat_test(norme(nouveau_vecteur(5,0)) == 5);


    printf("Test 2/3");
    afficher_resultat_test(norme(nouveau_vecteur(3,4)) == 5);


    printf("Test 3/3");
    afficher_resultat_test(norme(nouveau_vecteur(0,0)) == 0);
}
void test_addition_point(){
    printf("Test fonction addition_point\n");

    printf("Test 1/3");

afficher_resultat_test(egale_point(addition_point(nouveau_point(0,0),nouveau_point
(5,5)), nouveau_point(5,5)));


    printf("Test 2/3");

afficher_resultat_test(egale_point(addition_point(nouveau_point(-5,5),nouveau_poin
t(2,-8)), nouveau_point(-3,-3)));


    printf("Test 3/3");// v1+v2 == v2+v1

afficher_resultat_test(egale_point(addition_point(nouveau_point(5,2.5),nouveau_poi
nt(0.25,2)), addition_point(nouveau_point(0.25,2),nouveau_point(5,2.5))));
}

void test_produit_point(){
    printf("Test fonction produit_point\n");

    printf("Test 1/3");
    afficher_resultat_test(egale_point(produit_point(5,nouveau_point(5,5)),
nouveau_point(5*5,5*5)));


    printf("Test 2/3");
    afficher_resultat_test(egale_point(produit_point(-1,nouveau_point(2,-8)),
nouveau_point(-2,8)));


    printf("Test 3/3");

afficher_resultat_test(egale_point(produit_point(0,nouveau_point(64568,2.51365)),
nouveau_point(0,0)));
```

```
}

void test_distance_point(){
    printf("Test fonction distance_point\n");

    printf("Test 1/3");
    afficher_resultat_test(distance_point(nouveau_point(0,0),nouveau_point(3,4))
== 5);

    printf("Test 2/3");
    afficher_resultat_test(distance_point(nouveau_point(-5,8),nouveau_point(-5,8))
== 0);

    printf("Test 3/3");
    afficher_resultat_test( 2 *
distance_point(nouveau_point(0,0),nouveau_point(-5,8)) ==
distance_point(nouveau_point(5,-8),nouveau_point(-5,8)));
}

int main(int argc, char** argv){
    test_nouveau_point();
    test_nouveau_vecteur();
    test_point_to_vecteur();
    test_vecteur_to_point();
    test_couple_point_to_vecteur();
    test_addition_vecteur();
    test_produit_vecteur();
    test_produit_scalaire();
    test_norme();
    test_addition_point();
    test_produit_point();
    test_distance_point();
    return 1;
}
```

Resultats des tests du paquetage geometrie

```
Test fonction nouveau_point
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
Test fonction nouveau_vecteur
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
Test fonction point_to_vecteur
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
Test fonction vecteur_to_point
```

```
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
Test fonction couple_point_to_vecteur
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
Test fonction addition_vecteur
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
Test fonction produit_vecteur
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
Test fonction produit_scalaire
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
Test fonction norme
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
Test fonction addition_point
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
Test fonction produit_point
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
Test fonction distance_point
Test 1/3  SUCCESS
Test 2/3  SUCCESS
Test 3/3  SUCCESS
```