

 Unité de formation des Sciences de l'Ingénieur	ANNEE UNIVERSITAIRE 2016-2017 Session 2 Mention : Sciences pour l'ingénieur UE : 4TSI402U Epreuve : Examen d'informatique modulaire Date : Jeudi 22 juin 2017 Heure : 11h30 Durée : 1h30 Tout document papier autorisé
---	--

Géométrie

Après avoir développé un début de module de géométrie à deux dimensions (à l'occasion de l'examen de la session 1), on souhaite étendre le développement de ce module à de nouvelles formes géométriques. La première version du module permet de gérer des points et des droites. La deuxième version permettra de gérer en plus des cercles. Le fichier d'en-tête actuel `geometrie.h` du module est donné en annexe. On veut pouvoir y ajouter les fonctions nécessaires à des manipulations de cercles.

Fonctions mathématiques utiles

Pour manipuler les équations d'un cercle, on est amené à résoudre des équations du premier et du second degré. Avant de travailler sur le module `geometrie`, nous allons donc développer un petit module `resolution`. Ce module comportera pour l'instant deux fonctions : une fonction pour résoudre une équation du premier degré, une pour résoudre une équation du second degré. Le but de cette partie est d'écrire ce module. On utilisera le type `double` pour représenter les réels.

Pour les équations du second degré ou de degré inférieur, le nombre de solutions peut être égale à 0, 1, 2 ou une infinité (pour l'équation $0 = 0$). Afin de gérer ces 4 cas au retour de la fonction, on déclare la fonction `racines_polynome_2nd_degre` sur le principe suivant :

- la fonction retourne le nombre de solutions,
- 3 arguments `a2`, `a1` et `a0` représentant les coefficients du polynôme $a_2x^2 + a_1x + a_0$ sont passés **par valeur**,
- deux arguments `x1` et `x2` passés **par référence** permettant de récupérer les solutions si elles existent.

Pour les équations du premier degré, la fonction s'appelle `racines_polynome_1er_degre`, on utilise le même principe mais bien sûr, sans `a2` ni `x2`.

Par convention, les fonctions retournent -1 pour indiquer une infinité de solutions, et ne modifient ni `x1` ni `x2` dans ce cas.

Chaque fonction doit traiter le cas des polynômes de degré inférieur.

- 1) Écrivez le code (définition) de la fonction `racines_polynome_1er_degre`.
- 2) Dans quel fichier doit-on écrire ce code ?
- 3) Écrivez le code complet de `resolution.h`.
- 4) Écrivez le code de la fonction `racines_polynome_2nd_degre`. Cette fonction utilisera la fonction précédente pour les cas appropriés.

On écrit un programme de test du module nommé `test_resolution.c` donné en annexe.

- 5) Remplacez les lignes 3 et 13 par le code approprié.
- 6) Quelle ligne faut-il taper dans une console pour compiler ce programme ?
- 7) Quelle ligne faut-il taper dans une console pour exécuter ce programme ?

Module `geometrie`

Le code de `geometrie.h` est donné en annexe. On suppose que les définitions de toutes les fonctions déclarées avant la ligne `/* nouvelles fonctions */` sont déjà écrites.

Remarque : le choix de la représentation d'une droite par l'équation à trois paramètres a été fait pour autoriser les droites verticales et horizontales. Par conséquent, une même droite peut avoir plusieurs équations différentes, dont les coefficients sont proportionnels.

8) Écrivez le code (définition) de la fonction `cercle_creer`.

9) Écrivez le code (définition) de la fonction `distance_entre_points`.

10) Écrivez le code de la fonction `cercles_nombre_intersections` qui calcule le nombre d'intersection de deux cercles (sans calculer ces intersections).

Les fonctions `cercle_droite_calculer_intersections` et `cercles_calculer_intersections` utilisent le même principe de communication que les fonctions de résolution des polynômes de la première partie : elle renvoient le nombre de points d'intersection des deux formes géométriques passées par valeur, et donne ces points (s'il existent) par passage par référence.

11) Écrivez le code de la fonction `cercles_calculer_intersections` en supposant que la fonction `cercle_droite_calculer_intersections` est déjà écrite¹

1 ANNEXES : codes

```
1 /* test_resolution.c */
2
3 /* ICI IL MANQUE QUELQUE CHOSE */
4
5 int main(void)
6 {
7     double a,b,c ;
8     double s1,s2 ;
9     int nb_racines;
10    printf("saisissez les coefficients A, B, C du polynome A.x^2 + B.x + C") ;
11    scanf("%lf%lf%lf",&a, &a, &c) ;
12
13    /* ICI :APPEL de la fonction racines_polynome_2nd_degre */
14
15    if (nb_racines== -1)
16        printf("Tous les reels sont solutions\n") ;
17    else if (nb_racines==0)
18        printf("pas de racine reelle\n") ;
19    else if(nb_racines==1)
20        printf("une racine reelle : %f\n",s1) ;
21    else
22        printf("deux racines reelles : %f et %f\n",s1,s2) ;
23
24    return 0 ;
25 }
```

1. La solution mathématique des équations des points s'obtient en résolvant le système de deux équations quadratiques à deux inconnues. En développant ces équations et en en faisant la différence, vous obtenez une équation de droite. Les points recherchés sont donc l'intersection d'un des cercles et de cette droite.

```

1  /* geometrie.h */
2  #ifndef GEOMETRIE_H
3  #define GEOMETRIE_H
4
5  typedef struct {
6      double x;
7      double y;
8  } Point;
9
10 typedef struct {
11     double a;          /*eq : a * x + b * y + c = 0 */
12     double b;
13     double c;
14 } Droite;
15
16 typedef struct {
17     Point centre;
18     double rayon;
19 } Cercle;
20
21 Point point_creer(double x, double y);
22
23 Droite droite_creer_par2points(Point M, Point N);
24 Droite droite_creer_parallele(Droite D, Point P);
25 Droite droite_creer_perpendiculaire(Droite D, Point P);
26
27 Point droite_calculer_intersection(Droite D1, Droite D2);
28
29 /* nouvelles fonctions */
30 double distance_entre_points(Point M, Point N);
31
32 Cercle cercle_creer(Point centre, double rayon);
33
34 int cercles_nombre_intersections(Cercle C1, Cercle C2);
35
36 int cercle_droite_calculer_intersections(Cercle C, Droite D, Point *P1, Point *P2);
37 int cercles_calculer_intersections(Cercle C, Droite D, Point *P1, Point *P2);
38
39
40 #endif

```

FIN