Projet Informatique – Sections Electricité et Microtechnique

Printemps 2018 : *Decontaminators* © R. Boulic

Rendu1

1. Introduction

Le rendu1 se concentre sur le module **utilitaire** dont nous fournissons le fichier d'interface **utilitaire.h,** visible en annexe, à implémenter dans **utilitaire.c**.

2. Interface fournie à implémenter (annexe)

Ce module travaille avec deux types concrets S2D et C2D dont il montre la définition dans son interface.

Le module **utilitaire** étant une bibliothèque, il ne doit pas contenir de fonction main(). C'est un module de **test** indépendant qui va contenir une fonction main() destinée à appeler toutes les fonctions du module utilitaire pour les valider sur des exemples dont on connait le résultat. Une fonction qui renvoie un booléen doit être testée pour les deux cas VRAI et FAUX. Chaque appel de test doit afficher un message identifiant le test et s'il est validé ou pas.

Nous fournissons un module objet **test.o** contenant ce type de fonction main() pour tester votre module utilitaire. Bien sûr vous devez respecter scrupuleusement les noms de fonction et les modèles de structures fournis dans utilitaire.h pour pouvoir tirer parti de ce module objet. C'est avec ce type de programme de test que nous évaluerons votre rendu1.

Le rendu1 rapporte **8 points** sur un total de 52 points pour les 3 rendus. La répartition des points sera détaillée dans le document **Bareme** ; elle inclut une composante de style et de lisibilité du code source selon les conventions de programmation.

3. Forme du rendu1

Pour chaque rendu **UN SEUL membre d'un groupe** (noté **SCIPER1** ci-dessous) doit télécharger un fichier **zip** sur moodle (pas d'email). Le non-respect de cette consigne sera pénalisé. Le nom de ce fichier **zip** a la forme :

SCIPER1_ SCIPER2.zip

Compléter le fichier fourni **mysciper.txt** en remplaçant les numéros présents par les numéros SCIPER1 et SCIPER2 des 2 membres du groupe.

Le fichier archive du rendu1 doit contenir (aucun répertoire) :

- Fichier texte édité **mysciper.txt**
- Votre fichier **makefile** produisant un executable **test.x**
- votre module **utilitaire** = utilitaire.c et utilitaire.h (qui inclut ce que nous avons fourni)
- le fichier source **tolerance.h**
- le fichier objet **test.o**.

*On doit pouvoir produire l'exécutable **test.x** à partir du makefile après décompression du contenu du fichier zip.*

Auto-vérification : Après avoir téléchargé le fichier zip de votre rendu sur moodle (upload), récupérez-le (download), décompressez-le et assurez-vous que la commande make produit bien l'exécutable et que celui-ci fonctionne correctement.

Exécution sur la VM: votre projet sera évalué sur la VM du second semestre.

Backup : Vous êtes responsable de faire votre copie de sauvegarde du projet. Il y a un backup automatique seulement sur votre compte myNAS. Sur la VM, vous devez activer vous-même le backup (icone « engrenage » en haut à droite, choisir system settings, choisir backup et activer cette fonction en précisant les paramètres). Une alternative est de s'envoyer la dernière version du code source par email.

Gestion du code au sein d'un groupe : si vous décidez d'utiliser Git sur c4science, n'oubliez pas de supprimer l'accès public à votre code. Donnez accès seulement au partenaire du projet (tutorial Git sur moodle).

Annexe : contenu du fichier utilitaire.h

```
#ifndef UTILITAIRE_H
#define UTILITAIRE_H

#include <stdbool.h>
#include "tolerance.h"

//
// Types concrets exportés par le module utilitaire
//

// type et structure permettant de représenter un point ou un vecteur 2D
typedef struct S2d S2D;
struct S2d
{
	double x;
	double y;
};

// type et structure représentant un cercle dans le plan 2D
typedef struct C2d C2D;
struct C2d
{
	S2D centre;
	double rayon;
};

// ensemble des fonctions exportées

// renvoie la distance entre les points a et b
double	util_distance(S2D a, S2D b);

// renvoie l'angle que fait le bipoint ab et l'axe X du monde.
// L'angle doit être en radians et compris dans l'intervalle ]-pi, +pi]
double	util_angle(S2D a, S2D b);

// modifie si nécessaire l'angle pointé par p_angle
// pour qu'il soit compris dans l'intervalle ]-pi, +pi]
void	util_range_angle(double * p_angle);

// renvoie VRAI si le point est en dehors du domaine [-max, max]
bool	util_point_dehors(S2D a, double max);

// renvoie VRAI si l'angle alpha est en dehors de l'intervalle [-pi, pi]
bool	util_alpha_dehors(double alpha);

// renvoie VRAI si le point a est dans le cercle c
// plus précisément: si la distance de a au centre de c  < rayon - EPSIL_ZERO
bool	util_point_dans_cercle(S2D a, C2D c);

// renvoie VRAI en cas de collision des cercles a et b selon l'Equ. 4
// le paramètre de sortie p_dist est la distance entre les centres de a et b
bool	util_collision_cercle(C2D a, C2D b, double * p_dist);

// renvoie la position obtenue après déplacement du point p d'une distance dist
```

```
// dans la direction définie par l'angle alpha
S2D          util_deplacement(S2D p, double alpha, double dist);

// renvoie VRAI si la distance de a à b > EPSIL_ZERO et renvoie FAUX sinon.
// DE PLUS, dans le cas VRAI on utilise p_ecart_angle (qui doit être défini)
// pour récupérer l'écart angulaire entre le bipoint ab et un vecteur d'angle alpha.
// La valeur de l'écart angulaire doit être dans l'intervalle [-pi, pi].
bool        util_ecart_angle(S2D a, double alpha, S2D b, double *p_ecart_angle);

// renvoie VRAI si un vecteur d'angle alpha est aligné avec le vecteur ab. Pour
// déterminer cela on obtient l'écart angulaire avec la fonction util_ecart_angulaire
// et on revoie VRAI si cette fonction renvoie VRAI et si la valeur absolue de
// cet écart angulaire < EPSIL_ALIGNEMENT. Renvoie FAUX pour tous les autres cas.
bool        util_alignement(S2D a, double alpha, S2D b);

// renvoie VRAI si on peut calculer la nouvelle longueur du coté a lorsqu'on change
// la longueur du coté b, la longueur du coté c restant constante. Les longueurs des
// cotés a,b,c sont notées la, lb, lc. La nouvelle longueur du coté b est lb_new.
// le paramètre de sortie p_la_new doit être défini. Renvoie VRAI si:
// les 3 longueurs la, lb et lc > EPSIL_ZERO et lb_new se trouve dans l'intervalle
// autorisé [lb, lc]. Le calcul de la_new est donné par l'Equ.5 qui résoud le cas
// particulier de la Fig 5c avec:
//      la     = delta_d, lb = D, lc = L, lb_new = r1+r2
//      la_new = delta_d'
//
bool        util_inner_triangle(double la, double lb, double lc, double lb_new,
                                double * p_la_new);

#endif
```