Université Paris 6 - CNAM
DESS DLS

Module de test du logiciel
P. Ayrault - V. Ménissier-Morain

Examen du 25 Janvier 2002

Tous documents autorisés. Durée 2h.

1 Tests Structurels

Exercice 1

Soit un segment orienté de façon bi-directionnelle (par exemple, un tronçon de voie ferrée). Chaque point du segment est caractérisé par sa direction et l'abscisse de ce point suivant sa direction.

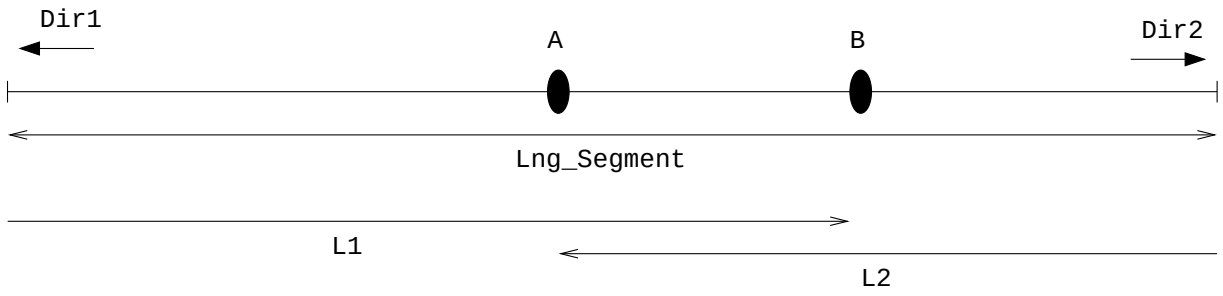


FIG. 1 – Segment bidirectionnel de longueur `Lng_Segment`

Par exemple sur la figure 1, le point A a la coordonnée $(DIR1, L2)$ et le point B a la coordonnée $(DIR2, L1)$.

La fonction `relation` fournie à la fin de l'exercice détermine la relation entre deux points du segment suivant une direction donnée :

- AMONT : si le point A est devant le point B dans la direction donnée
- AVAL : si le point A est derrière le point B dans la direction donnée
- EGAL : si les deux points sont à la même coordonnée.

Questions :

1. Déterminer les entrées et sorties du programme ;
2. Dessiner le graphe de contrôle de programme (organigramme) ;
3. Déterminer les jeux de tests pour obtenir la *couverture des instructions* du programme ;
4. Déterminer les jeux de tests pour obtenir la *couverture des branches* du programme ;
5. Déterminer les jeux de tests pour obtenir la *couverture des chemins* du programme.

Rappel : Les commentaires associés à chaque test doivent être explicites et faire référence aux parties testées.

Programme

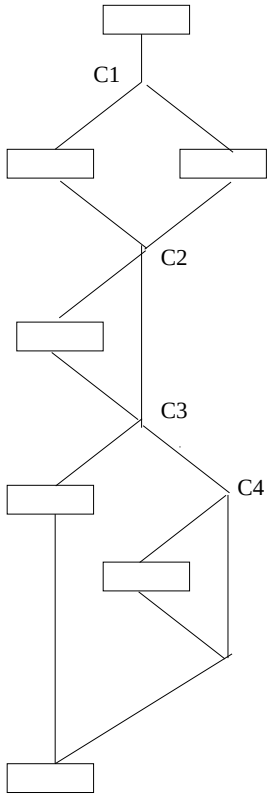
```
Lng_Segment : constant := 200;

type T_RELATION is ( AMONT, AVAL, EGAL);
type T_DIRECTION is ( DIR1, DIR2);
type T_POINT is
  record
    direction : T_DIRECTION;
    abscisse    : unsigned_integer_16;
  end record;

function Relation (P1, P2 : T_POINT; Dir : T_DIRECTION) return T_RELATION is
  res : T_RELATION;
  Absc_A, Absc_B : unsigned_integer_16;
begin
  Absc_B := P2.abscisse;
  Res := EGAL;

  -- Calcul abscisse premier point dans la direction Dir
  if( P1.direction = Dir ) then                              -- C1
    Absc_A := P1.abscisse
  else
    Absc_A := Lng_Segment - P1.abscisse;
  end if;

  -- Calcul abscisse second point dans la direction Dir
  if( P2.direction <> Dir ) then                             -- C2
    Absc_B := Lng_segment - P2.abscisse;
  end if;

  -- Comparaison des abscisses orientées
  if( Absc_A > Absc_B) then                                    -- C3
    res := AMONT
  elsif( Absc_A < Absc_B) then                                 -- C4
    res := AVAL;
  end if;

  return res;
}
```

Solution :

Entrées : P1, P2, Dir ou P1.Dir, P1.abscisse, P2.Dir, P2.Abscisse, Dir

Sorties : Relation

Graphe de contrôle :



Les tests des instructions :

Commentaire	P1.Dir	P1.Abs	P2.Dir	P2.Abs	Dir	Relation
A devant B avec même direction	Dir1	100	Dir2	120	Dir1	AMONT
A derrière B avec directions différentes	Dir1	100	Dir1	80	Dir2	AVAL

Les tests des branches :

Commentaire	P1.Dir	P1.Abs	P2.Dir	P2.Abs	Dir	Relation
A devant B avec même direction (C1,C2,C3)	Dir1	100	Dir2	120	Dir1	AMONT
A derrière B avec directions différentes (!C1, !C2, !C3,C4)	Dir1	100	Dir2	120	Dir2	AVAL
A = B avec même direction (C1,C2, !C3, !C4)	Dir1	100	Dir2	100	Dir1	EGAL

Les tests des chemins :

Commentaire	P1.Dir	P1.Abs	P2.Dir	P2.Abs	Dir	Relation
A devant B (C1,C2,C3)	Dir1	80	Dir2	140	Dir1	AMONT
A devant B (C1, !C2,C3)	Dir1	80	Dir1	50	Dir1	AMONT
A devant B (!C1,C2,C3)	Dir2	80	Dir2	140	Dir1	AMONT
A devant B (!C1, !C2,C3)	Dir2	80	Dir1	40	Dir1	AMONT
A derrière B (C1,C2, !C3,C4)	Dir1	80	Dir2	140	Dir1	AVAL
A derrière B (C1, !C2, !C3,C4)	Dir1	80	Dir1	140	Dir1	AVAL
A derrière B (!C1,C2, !C3,C4)	Dir2	80	Dir2	40	Dir1	AVAL
A derrière B (!C1, !C2, !C3,C4)	Dir2	80	Dir1	140	Dir1	AVAL
A égal B (C1,C2, !C3, !C4)	Dir1	80	Dir2	120	Dir1	EGAL
A égal B (C1, !C2, !C3, !C4)	Dir1	80	Dir1	80	Dir1	EGAL
A égal B (!C1,C2, !C3, !C4)	Dir2	40	Dir2	40	Dir1	EGAL
A égal B (!C1, !C2, !C3, !C4)	Dir2	60	Dir1	140	Dir1	EGAL

Exercice 2

Nous considérons un programme qui met à jour une entrée binaire de sécurité (valeur 0 ou 1). Cette entrée physique est lue via un bus réseau et doit être rafraîchie au moins toutes les minutes. Si l'entrée n'est pas rafraîchie, elle passe dans un état sûr (valeur restrictive).

L'algorithme est le suivant :

- si un message a été reçu durant le cycle courant,
  - vérifier la validité du message (temps écoulé entre son émission et l'heure courante).
  - si le message est valide, mettre à jour l'entrée avec les valeurs du message, sinon mettre l'entrée dans son état restrictif ;
- si aucune message n'a été reçu durant le cycle courant,
  - décrémenter la validité de l'entrée
  - si l'entrée est en *time-out*, la mettre dans son état restrictif.

Programme

```
#define MAX_VALIDITE 60 /* validité d'une entrée : 1 minute */

typedef T_TIME = unsigned int_16;

typedef struct {
  T_TIME emission;
  bool valeur;
} T_MESSAGE;

typedef struct {
  int_16 validite;
  bool valeur;
} T_VAL;

void mise_a_jour_entree (bool message_recu; T_MESSAGE msg;
                         T_TIME current_time; T_VAL * entree)
{
  if message_recu {                                          /* C1 */
    if ((current_time - msg.emission < 0) ||                 /* C2 */
        (current_time - msg.emission > MAX_VALIDITE)) {
      /* message reçu hors délai */
      *entree.valeur = false;  /* valeur par défaut */
      *entree.validite = 0;
    }
    else {
      /* mise à jour entrée avec la donnée du message */
      *entree.valeur = msg.valeur;
      *entree.validite = MAX_VALIDITE - (current_time - msg.emission);
    }
  }
  else {
    /* décrémenter la validité */
    *entree.validite = *entree.validite - 1;
    if( entree.validite < 0 ) {                              /* C3 */
      entree.valeur = false;
      entree.validite = 0;
```

```
    }
  }
}
```

Questions :

1. Déterminer les entrées et sorties du programme ;
2. Déterminer les jeux de tests pour obtenir la *couverture des données aux limites* du programme ;
3. Déterminer les jeux de tests pour obtenir la *couverture des données hors limites* du programme.

Solution :

Entrées : `message_recu`, `msg` (ou `msg.emission`, `msg.valeur`), `current_time`, `entree` (ou `entree.validite`)

Sorties : `entree` (ou `entree.validite`, `entree.valeur`)

Valeurs fonctionnelles de la validité de l'entrée : `0 .. MAX_VALIDITE`

Valeurs fonctionnelles de la validité des timers : `0 .. Max_Int16`

Commentaire	msg_recu	msg.emission	msg.valeur	current_time	ent.validite	ent.validite	ent.valeur
limite basse : message émis le même cycle que la réception (!!)	True	5000	True	5000	?	False	60
limite basse : message émis au cycle précédent sa réception	True	5000	True	5001	?	True	59
limite basse : entrée en time-out	False	?	?	?	0	False	0
limite basse : entrée passant en time-out	False	?	?	?	1	False	0
limite haute : message reçu hors délai	True	4000	True	4061		False	0
limite haute : message reçu au dernier cycle de validité	True	4000	True	4060		True	0
limite basse : message émis et reçu au premier cycle de l'automate (!)	True	0	True	0		True	60
limite haute : message émis et reçu au dernier cycle de l'automate (!)	True	Max_int16	True	Max_Int16		True	60
limite haute : message reçu au dernier cycle de l'automate	True	Max_int16 -30	True	Max_Int16		True	30

Valeurs limites informatiques de la validité : `-Max_int16 .. Max_int16`

Valeurs limites informatiques des timers : `0 .. Max_Int16` (déjà testé)

Commentaire	msg_recu	msg.emission	msg.valeur	current_time	ent.validite	ent.validite	ent.valeur
limite haute informatique validité	False	?	?	?	Max_Int16	?	Max_Int16 - 1
limite basse informatique validité	False	?	?	?	-Max_Int16	False	0 (ou débordement)

2 Tests Fonctionnels

Exercice 3

Il s'agit de tester la commande Unix **basename** dont le manuel d'utilisation est décrit ci-dessous.

```
BASENAME(1L)      Manuel de l'utilisateur Linux      BASENAME(1L)


  NOM
       basename  -  Eliminer le chemin d'accès et le suffixe d'un
       nom de fichier.

    SYNOPSIS
       basename nom [suffixe]
       basename --help,--version

    DESCRIPTION
       Cette page de manuel documente la version GNU de basename.

       La  commande basename élimine les répertoires en tête du
       chemin d'accès nom.

       Si un suffixe est indiqué, et s'il est identique à la par
       tie  finale du nom, il est éliminé de celui-ci. basename
       affiche le nom obtenu sur la sortie standard.

    OPTIONS
       -help
              Afficher un message d'aide sur la sortie  standard,
              et terminer normalement.

       --version
              Afficher  un  numéro de version sur la sortie stan
              dard, et terminer normalement.


    TRADUCTION
       Christophe Blaess, 1997.
```

Questions :

1. Déterminer les entrées et sorties de la commande ;
2. Déterminer les jeux de tests pour obtenir la *couverture nominale* de la commande.

Solution :

Entrées :

- options : -help –version
- paramètres :
  - nom (simple ou avec répertoire)
  - suffixe

Sorties : Nom de fichier

Tests nominaux :

paramètre	option	Résultat	Commentaire
	-help	Affichage d'un message d'aide	option aide
	–version	Affichage du numéro de version	option version
repertoire/nom		nom	un seul paramètre : nom avec répertoire
nom		nom	un seul paramètre : nom simple (sans répertoire)
repertoire/nom.suffixe .suffixe		nom	2 paramètres : nom avec bon suffixe
repertoire/nom.suffixe1 .suffixe2		nom.suffixe1	nom avec autre suffixe
nom.suffixe .suffixe		nom	nom simple avec suffixe
nom.suffixe1 .suffixe2		nom.suffixe1	nom simple avec autre suffixe