TD 8
Comment vérifier une multiplication

Algorithmique
Semestre 1

1 Fonction *sommeChiffres*

1.1

```
1 -- Calcul la somme des chiffres de l'entier e
2 fonction sommeChiffre (entree e <Entier>)
3             retourne <Entier>;
```

Listing 1 – En-tête de sommeChiffres

1.2

```
1  importer entreeSortie;
2
3  -- definition et corps de la fonction sommeChiffres
4
5  programme preuvePar9
6
7  glossaire
8    op1 <Entier>;
9    op1 <Entier>;
10   resteHaut <Entier>;
11   resteBas <Entier>;
12   resteGauche <Entier>;
13   resteDroit <Entier>;
14   resultat <Entier>; --produit calcule par l'utilisateur => a verifier
15
16 debut
17   ecrire("Valeur du multiplicande? ");
18   lire(op1);
19
20   ecrire("Valeur du multiplicateur? ");
21   lire(op2);
22
23   ecrire("Resultat calcule? ");
24   lire(Resultat);
25
26   resteHaut <- sommeChiffres(op1) mod 9;
27   resteBas <- sommeChiffres(op2) mod 9;i
28   resteDroit <- sommeChiffres(resteHaut * resteBas) mod 9;
29   resteGauche <- sommeChiffres(resultat) mod 9;
30
31   si resteDroit = resteGauche alors
```

```
32      ecrire("Rien a signaler");
33    sinon
34      ecrire("Multiplication eronnee");
35    fin si;
36  fin
```

Listing 2 – Programme de la preue par 9

1.3

```
1  -- Calcul la somme des chiffres de l'entier e
2  fonction sommeChiffre (entree e <Entier>)
3               retourne <Entier>
4
5  glossaire
6    quotient <Entier>;
7    sommeReste <Entier>;
8
9  debut
10   quotient <-e ;
11   sommeReste <- 0;
12
13   tantque quotient /= 0 faire
14     sommeReste <- sommeReste + (quotient mod 10);
15     quotient <- quotient div 10;
16   fin tantque;
17   retourner(sommeReste);
18 fin
```

Listing 3 – Corps de sommeChiffres

2 Fonction preuvePar9Valide

2.1

$\boxed{op_1^1, op_2^1, res^1}$, $\boxed{op_1^2, op_2^2, res_2}$, $\boxed{op_1^2, op_2^2, res^2}$, $\boxed{op_1^n, op_2^n, res^n}$, 0

2.2

on est pas capable de traiter $0 * x$

2.3

```
1  debut
2    ecrire("Taper le premier operande ou 0 pour arreter");
3    lire le premier operande;
4
5    tantque l entier lu est different de l entier zero faire
6      lire le second operande ;
7      lire le resultat calcule;
8
9      si la preuve par 9 des 3 derniers entiers lu est valide alors
10       ecrire que la multiplicatoin semble correct
11     sinon
12       ecrire qu il y a un probleme;
13     fin si;
14
```

```
    lire le premier operande;
  fin tantque;
fin
```

Listing 4 – Algorithme de vérification de multiplications

2.4

```
-- retourne VRAI si prod semble etre le produit de ope1 et de ope2 retourne FAUX sinon
fonction preuvePar9Valide (entree ope1 <Entier>,
                entree ope2 <Entier>,
                entree prod <Entier>)
                retourne <Booleen>;
```

Listing 5 – En-tête de la fonction preuvePar9Valide

2.5

```
importer entreeSortie;

-- retourne VRAI si prod semble etre le produit de ope1 et de ope2 retourne FAUX sinon
fonction preuvePar9Valide (entree ope1 <Entier>,
                entree ope2 <Entier>,
                entree prod <Entier>)
                retourne <Booleen>
debut
  retourner ((sommeChiffres(ope1)*sommeChiffres(ope2))mod9 = sommeChiffres(
      prod)mod9); --cf question 2-6
fin

programme preuvePar9

glossaire
  op1 <Entier>;
  op2 <Entier>;
  res <Entier>;

debut
  ecrire("Quel est le premier operande (0 pour arreter)");
  lire(op1);

  tantque op1 /= 0 faire
    ecrire("2nd operande");
    lire(op2);

    ecrire("Produit calcule?");
    lire(res);

    si preuvePar9Valide(op1,op2,res) alors
      ecrire("Le resultat semble correct");
    sinon
      ecrire("Le resultat est incorrect");
    fin si;
    ecrire("Quel est le premier operande (0pour arreter)");
    lire(op1);
```

```
37    fin tantque;
38  fin
```

Listing 6 – Programme de vérification de multiplications

2.6

```
1  -- retourne VRAI si prod semble etre le produit de ope1 et de ope2 retourne
       FAUX sinon
2  fonction preuvePar9Valide (entree ope1 <Entier>,
3                 entree ope2 <Entier>,
4                 entree prod <Entier>)
5                 retourne <Booleen>
6  debut
7    retourner ((sommeChiffres(ope1)*sommeChiffres(ope2))mod9 = sommeChiffres(
         prod)mod9);
8  fin
```

Listing 7 – Corps de la fonction preuvePar9Valide

Remarque : sommeChiffres(x) mod 9 = x mod 9

3 Procédure construireCroixStAndre

3.1

```
1  --construit les chifres de la croix saint andre
2  procedure construireCroixStAndre (entree opd1 <Entier>,
3                    entree opd2 <Entier>,
4                    entree res <Entier>,
5                    sortie croix <EnrCroix>)
6  debut
7    croix.haut <- sommeChiffres(opd1)mod9;
8    croix.bas <- sommeChiffres(opd2)mod9;
9    croix.droit <- sommeChiffres(opd1 * opd2)mod9;
10   croix.gauche <- sommeChiffres(res)mod9;
11 fin
```

Listing 8 – Corps de la procédure construireCroixStAndre

3.2

```
1  importer entreSortie;
2
3  type EnrCroix = enregistrement
4    haut <Entier>,
5    bas <Entier>,
6    droit <Entier>,
7    gauche <Entier>;
8
9  fonction sommeChiffre (entree e <Entier>) retourne <Entier>
10 glossaire
11   quotient <Entier>;
12   sommeReste <Entier>;
13
14 debut
15   quotient <-e ;
```

```
16    sommeReste <- 0;
17
18    tantque quotient /= 0 faire
19      sommeReste <- sommeReste + (quotient mod 10);
20      quotient <- quotient div 10;
21    fin tantque;
22    retourner(sommeReste);
23  fin
24
25  --construit les chifres de la croix saint andre
26  procedure construireCroixStAndre (entree opd1 <Entier>, entree opd2 <Entier
        >, entree res <Entier>,  sortie croix <EnrCroix>)
27  debut
28    croix.haut <- sommeChiffres(opd1)mod9;
29    croix.bas <- sommeChiffres(opd2)mod9;
30    croix.droit <- sommeChiffres(opd1 * opd2)mod9;
31    croix.gauche <- sommeChiffres(res)mod9;
32  fin
33
34  procedure ecrireCroixStAndre(
35  --- a completer
36  --
37  --
38
39  programme preuvePar9
40  glossaire
41    opd1 <Entier>; --Premier operande d'une multiplication
42    opd2 <Entier>; --deuxieme operande d'une multiplication
43    res <Entier>;  --resultat attendu de la multiplication
44    croix <EnrCroix>; --croix de saint Andre pour opd1 * opd2 = res
45  debut
46    -- ecrire le message d'invite
47    ecrire("Entrer une suit ede multiplication terminee par 0: ");
48    lire(opd1);
49
50    --traiter toutes les multiplications de la suite
51    tantque opd1 /= 0 faire
52      --lire le deuxieme operande de la multiplicaton
53      lire(opd2);
54      --lire le resultat attend pour la multiplication opd1 * opd2
55      lire(res);
56      --construire la croix de saint andre pour opd1*opd2 = res puis l'
            afficher
57      construireCroixStAndre(opd1;opd2,res,croix);
58      ecrireCroixStAndre(croix);
59      --tester la valideite de la multiplication opd1*opd2=res
60      si croix.gauche = croix.droit alors
61        ecrire("*** RAS ***");
62      sinon
63        ecrire("*** Multiplication erronnee ***");
64      fin si;
65      -- lire l'operande opd1 de la multiplication suivante
66      lire(opd1);
67    fin tantque;
68  fin
```

Listing 9 – Programme