TD 5
Entiers de Guiseppe Peano

TAD
Semestre 2

1

Utilisateur Je suis l'utilisateur (ou le client) du TAD je connais les **en têtes**, les **propriétés** mais ne connais pas le corps de l'implémentation.

Concepteur Je suis le concepteur du TAD. Je spécifie les **opérations** du type et les **propriétés** du type, définie les **entêtes** des sous programmes et et je dis si l'affectation et la comparaison sont (ou pas) autorisés.

Programmeur Je suis le programmeur du TAD. J'ai en charge la définition du TAD (**implémentation**) définie la **représentation mémoire** du type. Il code les différents sous programmes du concepteur.

2 Spécification fonctionnelle du TAD Peano

Peano	**Entier**
zéro	0
succ	+1
add	+
mult	*
inf	<
egal	=

2.1

$$
\begin{aligned}
(p1) 0 + p &= p \\
(p2)(p+1) + q &= (p+q) + 1 \\
(p3) 0 * p &= 0 \\
(p4)(p+1) * q &= (p * q) + q \\
(p5)\neg(0<0)(p6)\neg 0 < p+1 & \\
(p7)\neg(p+1<0) & \\
(p7) p+1 < q+1 \equiv p < q & \\
(p8) p+1 < q+1 \equiv p < q & \\
(p9) 0 = 0 & \\
(p10)\neg(0 = p+1) & \\
(p11)\neg(p+1 = 0) & \\
(p12) p+1 = q+1 \equiv p = q &
\end{aligned}
$$

2.2

```
add(succ(succ(zero)), succ(succ(succ(zero)))) =
succ(add(succ(zero), succ(succ(succ(zero))))) =
succ(succ(add(zero, succ(succ(succ(zero)))))) =
succ(succ(succ(succ(succ(zero))))) =
```

Utile au client

3 Spécification algorithmique du TAD Peano

3.1

```
fonction zero()
  retourne <Peano>;

fonction succ(entree p <Peano>)
  retourne <Peano>
  declenche debordement;

fonction add(entree p <Peano>, entree q <Peano>)
  retourne <Peano>
  declenche debordement;

fonction mult (entree p <Peano>, entree q <Peano)
  retourne <Peano>
  declenche debordement;

fonction inf (entree p <Peano>, entree q <Peano>)
  retourne <Booleen>;

fonction egal (entree p <Peano>, entree q <Peano>)
  retourne <Booleen>;
```

3.2

En-tête (cf 3.1) + Propriétés (cf sujet, page 2) + en-tête de l'affectation

3.3

```
-- client
glossaire
  p1 <Peano>;
  p2 <Peano>;
debut
  p1 <- zero;
  p1 <- p2;
  si p1 = p2 alors
    --...
fin
```

4 Utilisation du TAD Peano

4.1

```
-- en iteratif
fonction fact (entree p <Peano>)
  retourne <Peano>
  declenche debordement
glossaire
  r <Peano>; --retour
  i <Peano>; --compteur
debut
  i <- succ(zero);
  r <- succ(zero);
  tantque inf(i, p) ou i = p faire
    r <- mult(r, i);
    i <- succ(i);
  fin tantque;
  retourner r;
fin

-- En recursif :-)
fonction pred(entree p <Peano>)
  retourne <Peano>
debut
  si p = zero alors
    retourner(succ(zero));
  sinon
    retourner(mult(fact(pred(p), p)));
  fin si;
fin
```

4.2

```

fonction infEgal(entree p1 <Peano>,
         entree p2 <Peano>)
retourne <Booleen>
debut
  retourner inf(p1, succ(p2));
fin
```

5 Implémentation du TAD Peano

5.1

```
constante N <Entier> = 10 000;
type Peano : tableau [1 .. N] de <Caractere>;
```

5.2

```
fonction zero ()
  retourne <Peano>
glossaire
  p <Peano>;
debut
  p[1] <- '!';
  retourner p ;
fin


fonction succ(entree p <Peano>)
  retourne <Peano>
  declenche debordement
glossaire
  s <Peano>;
  i <Entier>;
debut
  i <- 1;

  tantque p[i] < '!' faire
    s[i] <- '*';
    i <- i + 1;
  fin tantque;
    s[i] <- '*';
  si i = LG_MAX alors
    declencher(debordement);
  fin si;

  s[i+1] = '!';
  retourner(s)
fin
```

5.3 opération add

```
-- s = add(p,q)
recopier les caracteres '*' de p dans s;
recopier les caracteres '*' de q dans s;
ajouter le caractere '!' a s;
```

Listing 1 – Algorithme général

```
fonction add(entree p <Peano>, entree q <Peano>)
  retourner <Peano>
glossaire
  i <Entier>:
  j <Entier>;
  r <Peano>; -- retour

debut
  -- recopier p;
  i <- 1;
  tantque p[i] != '!' faire
    r[i] = '*';
    i <- i + 1;
  fin tantque;
  i <- i - 1;

  --recopier q dans r
  j <- 1;
  tantque q[j] != '!' faire
    si (i+j) > N alors
      declencher debordement;
    fin si;
    r[i+j] <- '*';
    j <- j+1;
  fin tantque;

  -- ajouter '!' a s
  si (i+j) > N alors
    declencher debordement;
  fin si; r[i+j] = '!'; retourner r;

fin
```

Listing 2 – Programme

5.4 opération mult

```
se positionner sur le premier caractere '*' de p;
tantque il reste une etoile dans p faire
  recopier les caractere '*' de q dans le produit;
  passer au caractere '*' suivant de p;
fin tantque;
```

Listing 3 – Algorithme général

```
fonction mult (entree p <Peano>, entree q <Peano)
  retourne <Peano>
  declenche debordement
glossaire
  i <Entier>;
  j <Entier>;
  k <Entier>;
  r <Peano>;

debut
  k <- 1;
  i <-1;
  tantque p[i] /= '!' faire
    j <-1;
    tantque q[j] /= '!' faire
      si k > N alors
        declencher debordement;
      fin si;
      s[k] <- '*';
      j <- j + 1;
      k <- k + j;
    fin tantque;
    i <- i + j;
  fin tantque;
  si k > N alors
    declencher debordement;
  fin si;
  s[k] <- '!' alors
  retourner s;
fin
```

Listing 4 – Programme

5.5 opération inf

Principe parcourir en parallèle les deux entiers p et q et s'arrêter dès qu'un des deux entiers est épuisé (!);

5.5.1

```
se positionner sur le premier caractère de p;
se positionner sur le premier caractère de q;
tantque il reste un caractére '*' de p et dans q faire
  passer au caractére suivant de p;
  passer au caractére suivant de q;
fin tantque;
determer si pest épuisé avant q;
```

Listing 5 – Algorithme général

5.5.2

```
1  -- p < q
2  fonction inf(entree p <Peano>, entree q <Peano>)
3    retourne <Booleen>
4  glossaire
5    i <Entier>;
6
7  debut
8    i <- 1;
9    tantque p[i] /= '!' et q[i] /= '!' faire
10     i <- i + 1;
11   fin tantque;
12
13   retourner(p[i] = '!' et q[i] /= '!');
14 fin
```

Listing 6 – Programme

5.6 Implémentation du TAD Peano

5.6.1

5.6.2 Définition du type Peano

- Corps des sous-programmes zero, succ, add, mult et inf
- Corps du sous-programme "=" (à écrire)
- Corps du sous-programme "<-" (à étudier)