Les grands entiers

Motivation et objectifs du travail Le type `CARDINAL` de PASCAL ne permet de manipuler que les entiers de l'intervalle $[\![0, 2^{32} - 1]\!]$.

Le but de ce travail consiste à

- définir un type de données pour représenter des entiers quelconques sans aucune limitation. On se contentera de représenter les entiers positifs ou nuls (cf 1).
- programmer deux procédures d'interface entre cette représentation et celle de l'utilisateur (2).
- programmer les opérations arithmétiques de base (cf 4).
- programmer les opérateurs de comparaison (cf 5).

1 Représentation des grands entiers

Tout entier naturel peut être représenté par une liste de *chiffres*.

Par exemple, l'entier 2005 peut être représenté par

- la liste (2,0,0,5) si la base de représentation est 10 ;
- la liste (1,1,1,1,1,0,1,0,1,0,1) si la base de représentation est 2 ;
- la liste (20,5) si la base de représentation est 100 ;
- la liste (2,5) si la base de représentation est 1000 ;
- la liste (2005) si la base de représentation est 10000, etc.

L'intérêt de choisir une grande base est d'avoir des listes plus courtes qu'avec de petites bases.

Question 1. Quel est approximativement le rapport entre les tailles d'un entier écrit dans deux bases b et b' ?

L'intérêt de choisir une puissance de 10 comme base de représentation réside dans l'interface avec l'utilisateur : la conversion de la base 10 à la base 10^k, et réciproquement, ne nécessite aucun calcul.

On va donc choisir comme base b de représentation des entiers la plus grande puissance de 10 dont le carré est représentable par le type `CARDINAL`.

Question 2. Que vaut b ?

Pour des raisons algorithmiques qui apparaitront lors de la programmation des opérations arithmétiques, il est commode que la liste des chiffres d'un entier débute par le chiffre de poids faible, et se termine par le chiffre de poids fort.

Nous représenterons l'entier $n = 3141592 = 314b^1 + 1592b^0$ par la liste (1592, 314).

Quelle représentation pour l'entier 0 ? On pourrait bien entendu représenter 0 par la liste (0). Mais on pourrait tout aussi bien (ou mal ?) le représenter par les listes (0,0), (0,0,0),...

De même, si l est la liste des chiffres représentant un entier n, alors $l.(0)$ est aussi une liste représentant le même entier n.

Conclusion, 0 sera représenté par la liste vide, et tout entier non nul sera représenté par une liste non vide dont le dernier élément est un entier non nul.

Question 3. Définissez le type `GDENTIER`.

2 Procédures d'entrée/sortie

Question 4. Réalisez la fonction de conversion d'une chaîne de caractères en grand entier

```
// string2GDEntier() = liste des chiffres de
//        l'entier désigné par s
// CU : s ne contient que des chiffres décimaux
function string2GDEntier(s : STRING) : GDENTIER;
```

Exemple :

`string2GDEntier("3141592")` = (1592, 314)

Question 5. Réalisez les procédures d'entrée/sortie

```
procedure lireGDEntier(out n : GDENTIER);
```

et

```
procedure ecrireGDEntier(const n : GDENTIER);
```

3 Taille d'un grand entier

La taille d'un nombre entier en base b est le nombre de chiffres de l'écriture normale (sans 0 superflu) de cet entier en base b.

Question 6. Programmez la fonction

```
// tailleGDEntier(n) = nombre de chiffres de l'écriture décimale
// de n.
// La taille donnée peut dépasser légèrement la taille réelle.
function tailleGDEntier(n : GDENTIER) : CARDINAL;
```

4 Opérateurs arithmétiques

Question 7. Réalisez une fonction d'addition de deux entiers de type `GDENTIER`.

```
// add(n1,n2) = n1 + n2
function add(n1,n2 : GDENTIER) : GDENTIER;
```

Question 8. Réalisez de même une fonction de multiplication.

```
// mult(n1,n2) = n1 × n2
function mult(n1,n2 : GDENTIER) : GDENTIER;
```

Question 9. Réalisez de même une fonction de calcul de puissance.

```
// puissance(a,n) = a^n
function puissance(a : GDENTIER; n : CARDINAL) : GDENTIER;
```

Question 10. Pourquoi dans la programmation de la fonction `puissance` se limiter au cas où l'exposant est un `CARDINAL` ?

5 Opérateurs de comparaison

Question 11.

```
// compare(n1,n2) = -1 si n1 < n2
//                   0 si n - 1 = n2
//                   1 si n - 1 > n2
function compare(n1,n2 : GDENTIER) : INTEGER;
```

6 Suppléments

Question 12. Réalisez un programme illustant les différentes procédures et opérations. Votre programme pourra par exemple montrer le calcul de la factorielle de 100, qui est

```
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000
```

7 Méthodologie de travail

1. Vous devez absolument respecter la partie interface de l'unité demandée, en particulier ne rien y ajouter.
2. Mettez en commentaire toutes les fonctions de la partie interface. Vous les décomenterez au fur et à mesure de leur implantation.
3. Suivez l'ordre proposé par ce sujet : les procédures d'entrées/sorties d'abord, la fonction taille ensuite, les oprétaions arithmétiques demandées une par une, et enfin la fonction de comparaison.
4. Pour chaque fonction ou procédure de la partie interface réalisée, vous écrirez un programme de test. Ce programme de test prendra ses éventuels arguments sur la ligne de commande.

8 Annexe sur les chaînes de caractères

Quelques fonctions

Longueur d'une chaîne

```
// length(s) = nombre de caractères dans s
function length(s : STRING) : CARDINAL;
```

Sous-chaîne

```
// copy(s,i,l) = sous-chaîne de s de longueur l
//                    débutant à l'indice i
function copy(s : STRING; i,l : CARDINAL) : STRING;
```

Faîtes des tests avec cette fonction avant de l'utiliser pour étudier son comportement avec des valeurs des paramètres i et l dépassant les limites de la chaîne s.

Conversion d'un entier en une chaîne de caractères

```
// IntToStr(n) = chaine de caractères représentant l'écriture
//                décimale de l'entier n
function IntToStr(const n : INTEGER) : STRING;
```

Remarque : cette fonction fait partie de l'unité `SysUtils`.

Conversion d'une chaîne de caractères en un entier

```
// StrToInt(s) = l'entier représenté par
//                 la chaîne de caractères s.
function StrToInt(const s : STRING ) : INTEGER;
```

Remarque : cette fonction fait partie de l'unité `SysUtils`.