

**SMP**

TP2 Entiers Long

-

NOËL Antoine

## Table des matières

1.	Spécifications fonctionnelles et algorithmes .....	2
1.1	Fichier Utilitaire .....	2
1.1.1	Int2intLong .....	2
1.1.2	Égalité et comparaison .....	2
1.2	Fichier opérations .....	3
1.2.1	Addition .....	3
1.2.2	Soustraction .....	4
1.2.3	Multiplication.....	6
1.3	Fibonacci .....	7

# 1. Spécifications fonctionnelles et algorithmes

## 1.1 Fichier Utilitaire

Ce fichier utilitaire a pour but de contenir les différentes fonctions nécessaires à l'utilisation de la structure *entierLong*.

### 1.1.1 Int2intLong

Cette fonction convertit un entier standard en entier long. Elle prend en entrée un entier classique et le transforme en structure *entierLong* tout en conservant son signe original.

Le principe de conversion repose sur une décomposition décimale du nombre. Une boucle parcourt chaque position du tableau de chiffres. À chaque itération, l'opération modulo 10 permet d'extraire le chiffre des unités courant, qui est stocké dans la case correspondante du tableau. Le nombre est ensuite divisé par 10 pour passer au chiffre suivant.

La gestion du signe s'effectue parallèlement : un test vérifie si la valeur d'entrée est négative, et le résultat détermine la valeur de la variable booléenne qui marque le signe dans la structure.

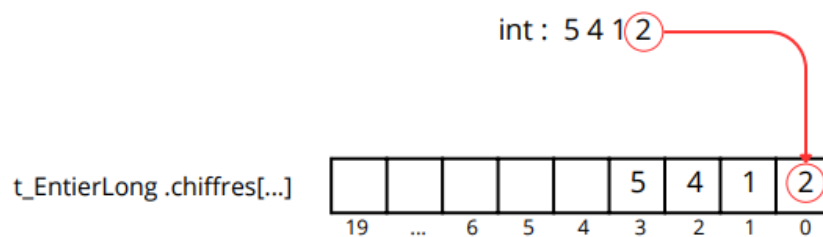


Figure 1 – Schéma du principe de la conversion d'un entier vers un entier long

Pour tester cette fonction, l'utilisation de la fonction *afficheEntierLong* est utilisée afin de vérifier la bonne écriture de l'entier standard dans la structure.

Afin de simplifier les essais tout au long du TP un fichier de configuration a été ajouté permettant de modifier les valeurs des entiers standard facilement sans à avoir à compiler l'entièreté du programme à chaque test.

Grâce à ces implantations, tous les tests ont pu être réalisés facilement.

### 1.1.2 Égalité et comparaison

Ces deux fonctions permettent de comparer deux entiers longs selon deux modalités différentes. La première fonction détermine l'égalité parfaite entre les deux nombres, tandis que la seconde établit une comparaison en valeur absolue pour savoir si le premier est inférieur ou égal au second.

Les deux fonctions prennent deux *entierLong* en entrées et renvoie un booléen. Une boucle est intégrée dans les deux fonctions.

- Pour la fonction d'égalité, une boucle parcourt chaque position des tableaux de chiffres des deux nombres. Si à chaque position les chiffres correspondent, la fonction poursuit la vérification, si une divergence est détectée, elle retourne immédiatement false. Seule une parfaite concordance sur tous les chiffres et sur le signe conduit à un retour true.

- Pour la fonction de comparaison, l'algorithme procède par soustraction des chiffres en partant des poids forts vers les poids faibles. Dès qu'une soustraction révèle que le chiffre

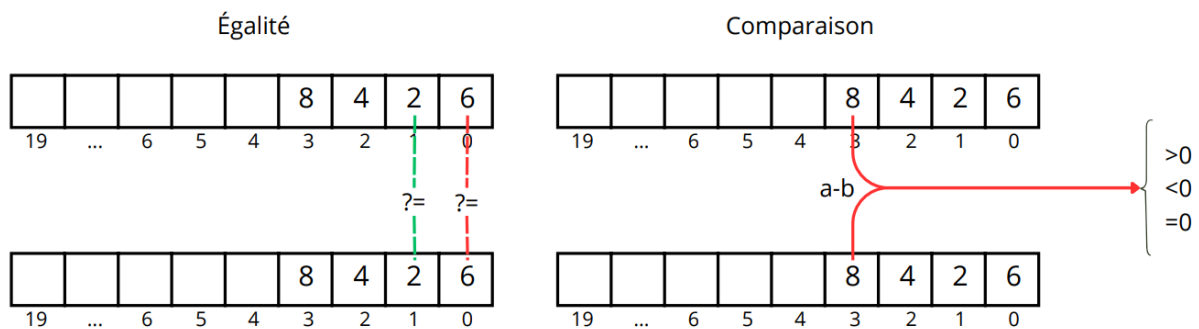


Figure 2 – Schémas des principes des fonctions égalité & comparaison

du premier nombre est supérieur à celui du second, la fonction retourne false. Si toutes les soustractions indiquent une infériorité ou une égalité du premier nombre, la fonction retourne true.

## 1.2 Fichier opérations

### 1.2.1 Addition

Le but de cette fonction est d'additionner deux entiers Long entre eux.

#### Cas signes égaux :

Ce cas est le plus simple car le résultat est une somme absolue des deux nombres. Pour le signe, celui-ci prend simplement la valeur d'un des deux.

À l'aide d'une boucle chaque valeur des tableaux est additionnée en commençant par le poids faible. Le calcul est le suivant :

$$r.chiffres[i] = (a.chiffres[i] + b.chiffres[i] + c) \% 10$$

avec :

$r \rightarrow$  résultat (EntierLong)

$c \rightarrow$  retenue (bool)

$a$  &  $b$  (EntierLong)

Le modulo permet de récupérer l'unité de la somme calculée afin de la rentrer dans le tableau.

La retenue quant à elle est vraie lorsque la somme est supérieure à 9.

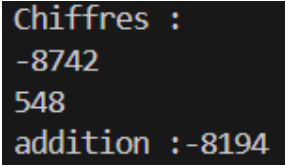
#### Cas signes différents.

L'addition devient une soustraction. Deux cas sont présents soit :  $a > b$  ou  $b \leq a$  ces deux conditions détermine à la fois le signe du résultat et l'ordre de la soustraction.

La fonction entièrement implantée, il faut désormais la tester. Pour cela, plusieurs entrées aléatoires sont choisies :

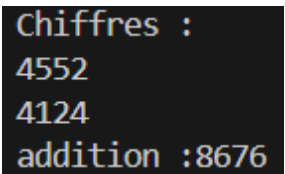
Entrées :  $a = -8742$  &  $b = 548$

Résultat attendu :  $-8194$

Résultat obtenu : 

Entrées :  $a = 4552$  &  $b = 4124$

Résultat attendu :  $8676$

Résultat obtenu : 

### 1.2.2 Soustraction

Cette fonction permet de soustraire deux entiers Long entre eux.

#### Cas signes égaux :

Dans ce cas deux sous-cas existent soit :  $a > b$  ou  $a < b$

Pour  $a > b$  :

Le résultat conserve le signe commun et une soustraction directe est effectuée. Une boucle calcule pour chaque chiffre la différence en soustrayant le chiffre correspondant de  $b$  et la retenue précédente. Si le résultat est négatif, on ajoute 10 et on active la retenue pour l'itération suivante.

Pour  $a \leq b$  :

Le résultat prend le signe inverse et la soustraction est inversée ( $b - a$ ). Le même mécanisme de retenue est appliqué, mais les opérandes sont échangés dans le calcul.

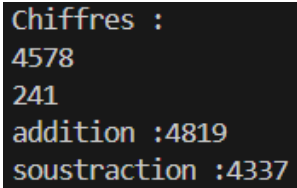
#### Cas signes différents :

La soustraction se transforme en addition :  $a - (-b) = a + b$  La fonction inverse le signe du second opérande et délègue le calcul à la fonction d'addition.

La phase de test de cette fonction peut désormais débiter :

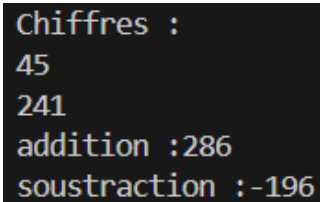
Entrées :  $a = 4578$  &  $b = 241$

Résultat attendu : 4337

Résultat obtenu : A terminal window with a black background and white text. It displays the following output:  
Chiffres :  
4578  
241  
addition :4819  
soustraction :4337

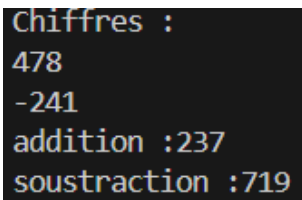
Entrées :  $a = 45$  &  $b = 241$

Résultat attendu : -196

Résultat obtenu : A terminal window with a black background and white text. It displays the following output:  
Chiffres :  
45  
241  
addition :286  
soustraction :-196

Entrées :  $a = 478$  &  $b = -241$

Résultat attendu : 719

Résultat obtenu : A terminal window with a black background and white text. It displays the following output:  
Chiffres :  
478  
-241  
addition :237  
soustraction :719

Les résultats démontrent que la fonction implantée fonctionne parfaitement.

### 1.2.3 Multiplication

Cette fonction permet de calculer le produit de deux entiers Long

L'algorithme implémente la méthode de multiplication traditionnelle par double boucle. Le résultat est initialisé à zéro.

La multiplication procède en deux boucles imbriquées. La boucle externe parcourt chaque chiffre du multiplicateur (b). Pour chaque chiffre du multiplicateur, la boucle interne parcourt tous les chiffres de (a). À chaque étape, le produit du chiffre courant de a et du chiffre courant de b est calculé, puis additionné à la position appropriée dans le résultat, en tenant compte de la retenue. La retenue est gérée par division entière par 10, tandis que le chiffre à conserver dans la position courante est obtenu par modulo 10.

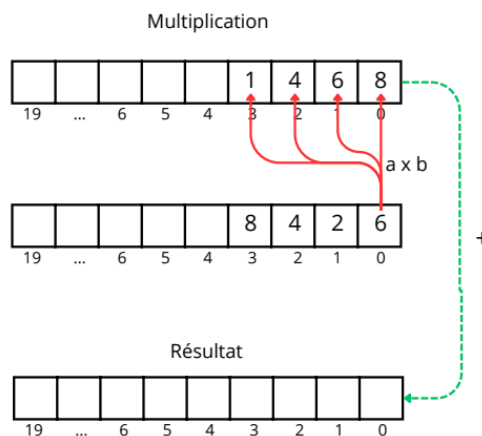


Figure 3 – Schéma du principe de la fonction multiplier

Afin de tester la fonctionnalité, plusieurs essais vont être réalisés :

Entrées :  $a = 1468$  &  $b = 8426$

Résultat attendu : 12 369 368

Résultat obtenu :

```
Chiffres :
1468
8426
addition :9894
soustraction :-6958
multiplication :12369368
```

Entrées :  $a = 87565$  &  $b = -652$

Résultat attendu : - 57 092 380

Résultat obtenu :

```
Chiffres :
87565
-652
addition :86913
soustraction :88217
multiplication :-57092380
```

Grâce à ces tests, la fonction multiplication a pu être validée.

### 1.3 Fibonacci

---

Afin de compléter les tests, la fonction de calcul de la suite de Fibonacci faite précédemment a été importée dans le projet. Cependant, la fonction a nécessité quelques modifications afin d'utiliser les fonctions d'opérations pour les entiers Long.

Le calcul de la suite est donc :

$$un = add(v[i - 1], v[i - 2])$$

*avec add  $\rightarrow$  la fonction d'addition*

Dès lors les modifications effectuées, la phase de test a pu reprendre avec notamment le calcul de la suite avec 20 000 valeurs, la vérification que  $n_{60} - n_{59} = n_{58}$  :

```
Fibonacci avec n=20000
44781108286810141877
97538287842201475624
42319396129011617501
n60-n59 = 591286729879
n58 = 591286729879
égalité : 1
```