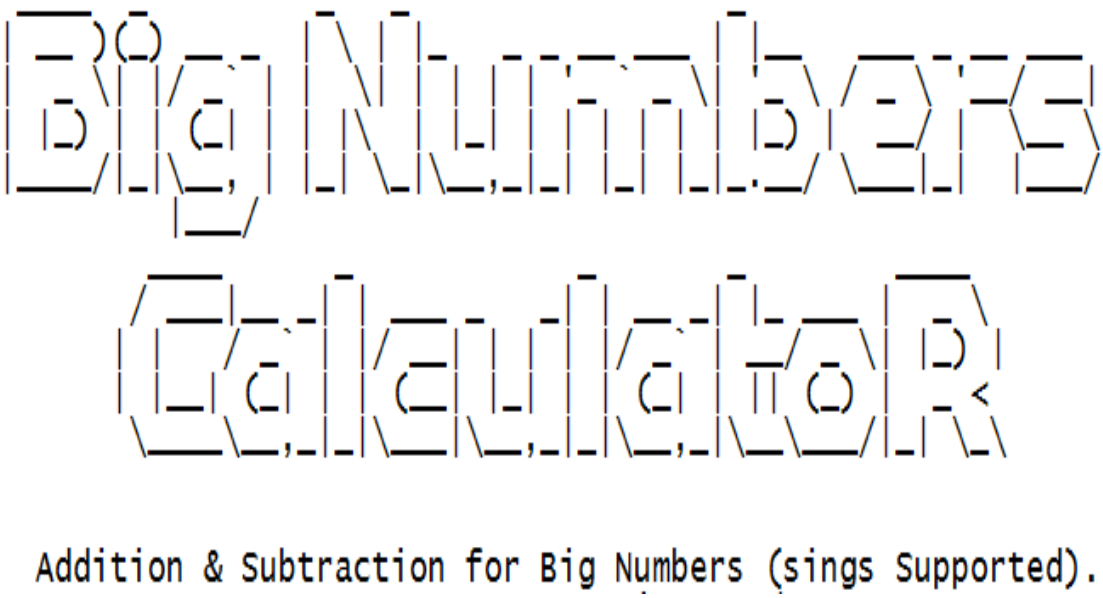
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ

1η Εργασία – Λίστες

Αριθμητική Μεγάλων Ακεραίων



ΑΛΈΞΑΝΔΡΟΣ ΠΛΈΣΣΙΑΣ
A.M.: 2025201100068
cst11068@uop.gr

Εαρινό εξάμηνο 2013-2014

Περιεχόμενα

Συναρτήσεις που χρησιμοποιήθηκαν

void TextArt();
Κείμενο καλωσορίσματος.

int Menu();
Εμφάνιση μενού και επιλογών.

void calcSimulation();
Προσομοίωση μέχρις ότου ο χρήστης πατήσει το μηδέν.

void readAndSaveNumber(int listCode);
Δυναμικό διάβασμα χαρακτήρα-χαρακτήρα και αποθήκευση σε στοίβα.

void insertDigitInBigNumber(int listCode, char digit);
Προσθήκη χαρακτήρα στη στοίβα του Αριθμού που θέλω (δηλαδή στην κεφαλή της απλά συνδεδεμένη λίστα).

void insertZero(int listCode,double position);
Προσθήκη χαρακτήρα (εδώ μηδέν) στο προτελευταίο κόμβο της στοίβας (πριν από το πρόσημο).

double sizeOfNumber (int listCode);
Υπολογίζεται το μέγεθος του αριθμού (δηλαδή κόμβων) καλείται πριν την alignmentNumbers ().

void alignmentNumbers();
Υπολογίζει πόσα μηδενικά θα πρέπει να προστεθούν στο πρώτο και δεύτερο αριθμό να είναι ίσοι.

void addFunction ();
Λειτουργία πρόσθεσης ζητά αριθμούς, βρίσκει τα πρόσημα, καλεί την addNumbers() ή την subNumbers(), δείχνει το αποτέλεσμα και στη συνέχεια απελευθερώνει τα stacks.

void addNumbers(char sign);
Προσθέτει το πρώτο και το δεύτερο αριθμό και τοποθετεί το πρόσημο που δίνεται ως παράμετρος.

void subFunction();
Λειτουργία αφαίρεσης ζητά αριθμούς, βρίσκει τα πρόσημα, καλεί την addNumbers() ή την subNumbers(), δείχνει το αποτέλεσμα και στη συνέχεια απελευθερώνει τα stacks.

void subNumbers(bigNumber* subtracter ,bigNumber* subtrahend,char sign);
Αφαιρεί τον αφαιρέτη από τον αφαιρετέο και τοποθετεί το πρόσημο που δίνεται ως παράμετρος.

int greaterFirstOrSecond(double position);
Ελέγξτε ψηφία για μια συγκεκριμένη θέση (και στους 2 αριθμούς) και να μας ενημερώνει σχετικά με το ποιος είναι μεγαλύτερος ή αν είναι ίσοι.

void printNumber(int listCode);
Εμφάνιση του αριθμού που δίνεται σαν παράμετρος και κρύβει τα πρώτα μηδενικά.

void freeList(int listCode);
Αδειάστε στοίβας και αρχικοποίηση δείκτη κεφαλής.

void clearBuffer();
Καθάρισμα Buffer εισόδου.

Γενική Περιγραφή

Αρχικά, ζητάμε από τον χρήση ποια λειτουργία θέλει να εφαρμόσει αφού επιλέξει μια λειτουργία του προγράμματος του ζητάμε να πληκτρολογήσει τούς 2 αριθμούς που θέλει. Αξίζει να σημειωθεί πως ο χρήστης έχει την δυνατότητα να βάλει και πρόσημο πριν τον αριθμό εάν δεν βάλει, εισάγεται αυτόματα το πρόσημο '+'. Ο αριθμός εισάγεται δυναμικά έναν-έναν χαρακτήρα (χρήση getchar) και αποθηκεύεται στον αντίστοιχο κόμβο της λίστας του αριθμού. Η λίστα που χρησιμοποιήσουμε μοιάζει πολύ με στοίβα (stack) ακολουθεί η τεκμηρίωση δηλαδή γιατί επιλέχτηκε στην ενότητα "Επιλογή Λίστας". Εννοείτε πως πάντα ελέγχουμε για όλα τα πιθανά λάθει εισαγωγής από τον χρήστη, για τον κάθε αριθμό ξεχωριστά δηλαδή εάν πληκτρολογήσει τον έναν αριθμό λάθος από τούς 2 αυτόν και μόνο ξαναδίνει επίσης στην περίπτωση λάθους καθαρίζεται ο buffer (συνάρτηση: clearBuffer()) και αδειάζει η λίστα (συνάρτηση: freeList()). Έπειτα ανάλογα με την επιλογή του χρήστη εφαρμόζουμε την λειτουργία της πρόσθεσης ή της αφαιρέσεις καλώντας τις συναρτήσεις addFunction ή subFunction των οποίων η εξήγηση υπάρχουν στις ενότητες "addFunction" και "subFunction". Τέλος, αφού γίνει η λειτουργία που πρέπει εμφανίζεται το αποτέλεσμα στην οθόνη και δίνεται στον χρήστη η δυνατότατη να τερματίσει ή να συνεχίσει το πρόγραμμα.

Επιλογή Λίστας

```
typedef struct bigNumber
{
   char digitOfNumber;
   struct bigNumber *next;
} bigNumber;
```

Η λίστα του συγκεκριμένου προγράμματος είναι μια άπλα συνδεδεμένη της οποία το struct φαίνεται παραπάνω και τα νέα της στοιχειά εισάγονται κάθε φορά στην κεφαλή της όπως ακριβώς με την είσοδο που έχει και μια στοίβα.

```
First Number: +15829
'+' -> NULL
'1' -> '+' -> NULL
'5' -> '1' -> '+' -> NULL
'8' -> '5' -> '1' -> '+' -> NULL
'2' -> '8' -> '5' -> '1' -> '+' -> NULL
'9' -> '2' -> '8' -> '5' -> '1' -> '+' -> NULL

Second Number: +89741
'+' -> NULL
'8' -> '+' -> NULL
'9' -> '8' -> '+' -> NULL
'7' -> '9' -> '8' -> '+' -> NULL
'4' -> '7' -> '9' -> '8' -> '+' -> NULL
'1' -> '4' -> '7' -> '9' -> '8' -> '+' -> NULL
```

Εικόνα 1: Εισαγωγή στοιχείων στην στοίβα και ιδιότητα αναστροφής.

Επέλεξα μια άπλα συνδεόμενη λίστα για ταχεία εισαγωγή κόμβων στην αρχή της, γιατί χρειάζεται μονάχα έναν δείκτη για να προσπελαστεί, για λιγότερη δεσμεύσει μνήμης λόγο του ενός δείκτη και για την αναστροφή πραγμάτων (Εικόνα 1 και Εικόνα 2).

Μνήμη

Αξίζει να να σημειωθεί πως το struct αποθηκεύει το ψηφίο του αριθμού σε μορφή char το οποίο καταλαμβάνει 1 byte στην μνήμη συνήθως και ο δείκτης του επόμενου που είναι ίσος με sizeof(int) δηλαδή 4 bytes συνήθως. Άρα εξοικονομούμε πολύ χώρο σε μνήμη σε σχέση με κάποια υλοποίηση η οποία θα μπορούσε να ήταν μια διπλά συνδεδεμένη λίστα με integer για την αποθήκευση του ψηφίου. Ποιο συγκεκριμένα θα είχε μέγεθος περίπου 12 bytes/per digit σε σχέση με την υλοποίηση μου που είναι 5 bytes/per digit άρα μια διαφορά τής τάξις του 42%. περίπου per digit.

addFunction

Εδώ ζητάμε από τον χρήστη να μας δώσει τους αριθμούς που θέλει, αφού εξετάσουμε το μήκος του κάθε αριθμού συμβουλευόμαστε τούς κάτωθεν πινάκα για να δούμε τι πρόσημο θέλει τελικά έπειτα καλούμε την συνάρτηση που κάνει τον πρόσθεση ή την αφαίρεση ανάλογα με το ποιο είναι μεγαλύτερος ή μικρότερος αφοί ποιο πριν έχουμε πρόσθεση μηδενικά για να έχουν το ίδιο μήκος, Εάν έχουν εξ΄ αρχής το ίδιο μήκος εξετάζουμε ψηφίο ψηφία για να βρούμε τον μεγαλύτερο. Τέλος, δείχνουμε το αποτέλεσμα και απελευθερώνουμε τις λίστες.

size(a) > size(b)							
(+ a) + (- b)		(- a) + (+ b)		(+ a) + (+ b)		(- a) + (- b)	
SubNumbers (a – b)	Sign '+'	SubNumbers (a – b)	Sign ' - '	AddNumbers ()	Sign '+'	AddNumbers ()	Sign '-'

size(a) < size(b)							
(+ a) + (- b)		(- a) + (+ b)		(+ a) + (+ b)		(- a) + (- b)	
SubNumbers (b – a)	Sign '-'	SubNumbers (b – a)	Sign '+'	AddNumbers ()	Sign '+'	AddNumbers ()	Sign '-'

size(a) == size(b)							
number(a) > number(b)							
(+ a) + (- b)		(- a) + (+ b)		(+ a) + (+ b)		(- a) + (- b)	
SubNumbers (a – b)	Sign '+'	SubNumbers (a – b)	Sign ' - '	AddNumbers ()	Sign '+'	AddNumbers ()	Sign '-'
number(a) < number(b)							
(+ a) + (- b)		(- a) + (+ b)		(+ a) + (+ b)		(- a) + (- b)	
SubNumbers (b – a)	Sign '-'	SubNumbers (b – a)	Sign '+'	AddNumbers ()	Sign '+'	AddNumbers ()	Sign '-'
number(a) == number(b)							
(+ a) + (- b)		(- a) + (+ b)		(+ a) + (+ b)		(- a) + (- b)	
SubNumbers (b – a)	Sign '-'	SubNumbers (b – a)	Sign '-'	AddNumbers ()	Sign '+'	AddNumbers ()	Sign '-'

subFunction

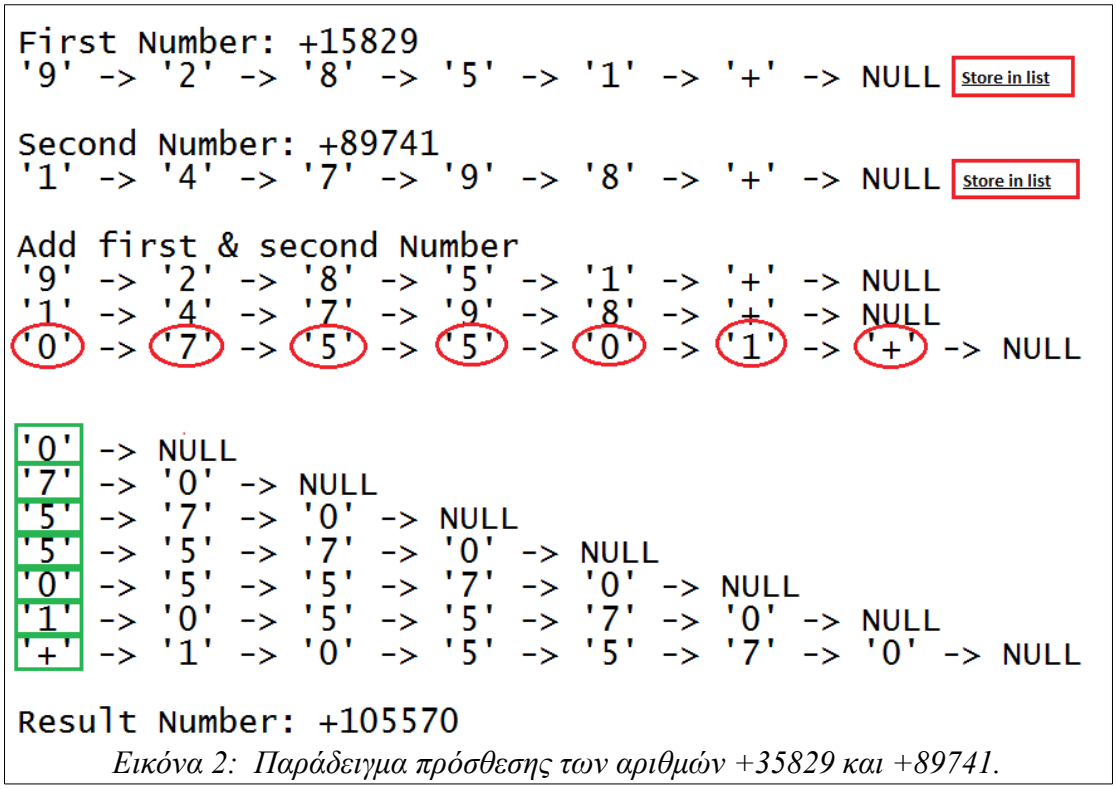
Εδώ ζητάμε από τον χρήστη να μας δώσει τους αριθμούς που θέλει, αφού εξετάσουμε το μήκος του κάθε αριθμού συμβουλευόμαστε τούς κάτωθεν πινάκα για να δούμε τι πρόσημο θέλει τελικά έπειτα καλούμε την συνάρτηση που κάνει τον πρόσθεση ή την αφαίρεση ανάλογα με το ποιο είναι μεγαλύτερος ή μικρότερος αφοί ποιο πριν έχουμε πρόσθεση μηδενικά για να έχουν το ίδιο μήκος, Εάν έχουν εξ΄ αρχής το ίδιο μήκος εξετάζουμε ψηφίο ψηφία για να βρούμε τον μεγαλύτερο. Τέλος, δείχνουμε το αποτέλεσμα και απελευθερώνουμε τις λίστες.

size(a) > size(b)							
(+ a) – (- b)		(- a) – (+ b)		(+ a) – (+ b)		(- a) – (- b)	
AddNumbers ()	Sign '+'	AddNumbers ()	Sign ' - '	SubNumbers (a – b)	Sign '+'	SubNumbers (a – b)	Sign ' - '

size(a) < size(b)							
(+ a) – (- b)		(- a) – (+ b)		(+ a) – (+ b)		(- a) – (- b)	
AddNumbers ()	Sign '+'	AddNumbers ()	Sign ' - '	SubNumbers (b – a)	Sign '-'	SubNumbers (b – a)	Sign ' + '

size(a) == size(b)							
number(a) > number(b)							
(+ a) – (- b)		(- a) – (+ b)		(+ a) – (+ b)		(- a) – (- b)	
AddNumbers ()	Sign '+'	AddNumbers ()	Sign ' - '	SubNumbers (a – b)	Sign '+'	SubNumbers (a – b)	Sign ' - '
number(a) < number(b)							
(+ a) – (- b)		(- a) – (+ b)		(+ a) – (+ b)		(- a) – (- b)	
AddNumbers ()	Sign '+'	AddNumbers ()	Sign ' - '	SubNumbers (b – a)	Sign '-'	SubNumbers (b – a)	Sign ' + '
number(a) == number(b)							
(+ a) – (- b)		(- a) – (+ b)		(+ a) – (+ b)		(- a) – (- b)	
AddNumbers ()	Sign '+'	AddNumbers ()	Sign ' - '	SubNumbers (a – b)	Sign '+'	SubNumbers (a – b)	Sign '+'

Παράδειγμα Πρόσθεση



Εικόνα 2: Παράδειγμα πρόσθεσης των αριθμών +35829 και +89741.