Εργαστήριο 5

Εαρινό Εξάμηνο 2012-2013

Στόχοι του εργαστηρίου

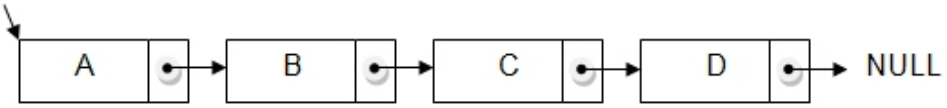
- Χρήση στοίβας
- Αναδρομή
- Δομές δεδομένων
- Δυναμική δέσμευση μνήμης

Σκοπός της 5ης εργαστηριακής άσκησης είναι να εξοικειωθείτε με την χρήση της στοίβας. Αυτό θα σας επιτρέψει να υλοποιείτε καλύτερα συναρτήσεις οι οποίες θα μπορούν να καλέσουν με την σειρά τους άλλες συναρτήσεις ή ακόμα και τον εαυτό τους οπότε και λέγονται αναδρομικές. Ανατρέξτε στα παραδείγματα που κάνατε στην τάξη πάνω στην αναδρομή.

Δομές Δεδομένων

Μία δομή δεδομένων είναι μια συλλογή δεδομένων με κάποιες ιδιότητες η οποία προσφέρει εύκολη πρόσβαση σε αυτά τα δεδομένα. Έχουμε δει ως τώρα μία στατική δομή δεδομένων, τον πίνακα. Τι κάνουμε όμως στην περίπτωση που θέλουμε να αποθηκεύουμε ένα συνεχώς μεταβαλλόμενο αριθμό από δεδομένα;
Είναι δομές δεδομένων που μπορούν να αυξομειώνουν το μέγεθός τους, δηλαδή να αναπτύσσονται και να συρρικνώνονται στο χρόνο εκτέλεσης. Σε αυτό το εργαστήριο θα δούμε μία σημαντική δομή δεδομένων: την διασυνδεδεμένη λίστα
Μία **διασυνδεδεμένη λίστα** είναι μια γραμμική συλλογή δομών, που ονομάζονται κόμβοι, και συνδέονται με δείκτες. Μία συνδεδεμένη λίστα προσπελαύνεται μέσω ενός δείκτη στον πρώτο κόμβο της λίστας (κεφαλή - head). Κατά σύμβαση, ο δείκτης σύνδεσης στον τελευταίο κόμβο της λίστας έχει μηδενική τιμή (NULL) για να σηματοδοτήσει το τέλος της λίστας.

Κεφαλή - Head



Για την περιγραφή της άσκησης θα χρησιμοποιήσουμε την εξής αναπαράσταση κόμβων:

Κόμβος λίστας:

```
typedef struct node {
      int data;
      struct node *next;
} nodeL;
```

Δυναμική Εκχώρηση Μνήμης (Dynamic Memory Allocation)

Η άσκηση 2 θα ζητάει και θα παίρνει κόμβους από το λειτουργικό σύστημα «δυναμικά», την ώρα που τρέχει (σε run-time). Για το σκοπό αυτό θα χρησιμοποιήστε την κλήση συστήματος (system call) «sbrk» (set break). Η κλήση αυτή «σπρώχνει» πιο πέρα (προς αύξουσες διευθύνσεις μνήμης) το σημείο «break», το όριο πριν από το οποίο οι διευθύνσεις μνήμης που γεννά το πρόγραμμα είναι νόμιμες, ενώ μετά από το οποίο (και μέχρι την αρχή της στοίβας) οι διευθύνσεις είναι παράνομες. Η κλήση συστήματος «sbrk» (syscall 9) παίρνει σαν παράμετρο το πλήθος των νέων bytes που επιθυμείτε στον καταχωρητή $a0. Μετά την επιστροφή της κλήσης, ο καταχωρητής $v0 περιέχει την διεύθυνση του νέου block μνήμης, του ζητηθέντος μεγέθους, που το σύστημα δίνει στο πρόγραμμά σας (έναν pointer). Σε περίπτωση που η διεύθυνση είναι μηδενική τότε έχει γεμίσει όλη η μνήμη και σε τέτοια περίπτωση το πρόγραμμά σας θα πρέπει να τερματίσει.

Για το επόμενο εργαστήριο έχετε να υλοποιήσετε τις 2 παρακάτω εργαστηριακές ασκήσεις. Την ώρα του εργαστηρίου θα εξετασθείτε προφορικά πάνω στους κώδικες που θα παραδώσετε.

Άσκηση 1- Αναδρομή (5 μονάδες)

Η ρωμαϊκή αρίθμηση χρησιμοποιεί τα παρακάτω ψηφία:

- M = 1000
- D = 500
- C = 100
- L = 50

- X = 10
- V = 5
- I = 1

Σε γενικές γραμμές, τα ρωμαϊκά ψηφία είναι γραμμένα σε φθίνουσα σειρά από αριστερά προς τα δεξιά, και προστίθενται διαδοχικά, για παράδειγμα MMVI (=2006) ερμηνεύεται ως 1000 + 1000 + 5 + 1.

Ορισμένοι συνδυασμοί όμως τηρούν την αφαιρετική αρχή, η οποία ορίζει ότι όταν ένα σύμβολο μικρότερης αξίας προηγείται ένα σύμβολο μεγαλύτερης αξίας, η μικρότερη τιμή αφαιρείται από τη μεγαλύτερη τιμή, και το αποτέλεσμα προστίθεται στο σύνολο. Για παράδειγμα, MCMXLIV (=1944), τα σύμβολα C, X και I προηγούνται ένα σύμβολο μεγαλύτερης αξίας, και το αποτέλεσμα ερμηνεύεται ως εξής:
1000 + (1000 - 100) + (50 - 10) + (5 - 1).

1 = I	11 = XI	10 = X	100 = C	1 000 = M
2 = II	12 = XII	20 = XX	200 = CC	2 000 = MM
3 = III	13 = XIII	30 = XXX	300 = CCC	3 000 = MMM
4 = IV	14 = XIV	40 = XL	400 = CD	4 000 = MMMM
5 = V	15 = XV	50 = L	500 = D	
6 = VI	16 = XVI	60 = LX	600 = DC	
7 = VII	17 = XVII	70 = LXX	700 = DCC	
8 = VIII	18 = XVIII	80 = LXXX	800 = DCCC	
9 = IX	19 = XIX	90 = XC	900 = CM	

Να γραφτεί πρόγραμμα το οποίο θα διαβάζει από το πληκτρολόγιο μία συμβολοσειρά και θα καλέσει ανάλογα τις δύο παρακάτω αναδρομικές συναρτήσεις:

α) (1 μονάδα) Αναδρομική συνάρτηση int check_roman(char *s)
Η συνάρτηση αυτή επιστρέφει 1 όταν η συμβολοσειρά αποτελείται μόνο από ρωμαϊκά ψηφία, αλλιώς επιστρέφει 0. Μπορείτε να θεωρήσετε ότι δεχόμαστε μόνο τα κεφαλαία αντίστοιχα γράμματα: DCXI επιστρέφει 1, dCXi επιστρέφει 0, dcxi επιστρέφει 0.
β) (4 μονάδες) Αναδρομική συνάρτηση int roman_to_decimal(char *s)
Η συνάρτηση αυτή δέχεται μία συμβολοσειρά που αποτελείται μόνο από ρωμαϊκά ψηφία (δηλαδή η main θα την καλέσει μόνο εφόσον έχει επιστρέψει 1 η συνάρτηση check_roman). Θα υπολογίσει τον αντίστοιχο δεκαδικό αριθμό και θα τον επιστρέψει. Μπορείτε να θεωρήσετε ότι η συμβολοσειρά που θα δοθεί τηρεί τους κανονισμούς της αναπαράστασης ρωμαϊκών αριθμών και είναι έγκυρο αριθμό.

Στην συνέχεια η main θα εκτυπώσει στην οθόνη το αποτέλεσμα της roman_to_decimal εφόσον η check_roman επέστρεψε 1, αλλιώς θα εμφανίσει

κατάλληλο μήνυμα λάθους στην οθόνη και θα διαβάσει νέα συμβολοσειρά από το πληκτρολόγιο μέχρι που να δοθεί έγκυρη συμβολοσειρά.

ΠΡΟΣΟΧΗ: Όταν διαβάζετε μία συμβολοσειρά από το πληκτρολόγιο, διαβάζεται και το '\n' όταν πατάτε enter. Μπορείτε να υλοποιήσετε μία βοηθητική συνάρτηση που θα αντικαταστήσει αυτό το '\n' με το '\0' για να είναι κανονικό NULL-terminated string, και να μην έχετε να ασχοληθείτε με το newline.
Επίσης σας συνιστούμε πρώτα να αναπτύξετε τις αναδρομικές συναρτήσεις σε ψευδοκώδικα ή σε κάποια γλώσσα τύπου C. Θα σας είναι πολύ πιο εύκολο αργότερα να μεταφέρετε τον κώδικα σε assembly.

ΠΡΟΣΟΧΗ: Λύσεις που δεν χρησιμοποιούν αναδρομή ΔΕΝ θα γίνουν δεκτές!

Άσκηση 2 – Δυναμική εκχώρηση μνήμης (5 μονάδες)

Στην δεύτερη άσκηση θα χρησιμοποιήσουμε μία δομή δεδομένων που θα αποτελεί ένα κόμβο μιας διασυνδεδεμένης λίστας (linked list). Κάθε κόμβος θα αποτελείται από δύο λέξεις (των 32 bits καθεμία): έναν ακέραιο «data» που θα περιέχει την «πληροφορία χρήστη», κι έναν δείκτη σύνδεσης (pointer) «next» που θα περιέχει τη διεύθυνση του επόμενου κόμβου στη λίστα. Στον τελευταίο κόμβο της λίστας, next=0. Τα δύο στοιχεία (λέξεις) του κόμβου μας θα βρίσκονται σε διαδοχικές θέσεις της μνήμης. Επομένως, κάθε κόμβος θα έχει μέγεθος 2 x 4 = 8 bytes. Διεύθυνση ενός κόμβου είναι η διεύθυνση του πρώτου στοιχείου του, δηλαδή του στοιχείου με «μηδενικό offset», που για μας είναι το «data».

α) (0.5 μονάδες) Να γραφεί μία συνάρτηση **nodeL* insertElement(nodeL *head, int data)** όπου head είναι δείκτη στην αρχή της διασυνδεδεμένης λίστας και data είναι ο ακέραιος που θέλουμε να εισάγουμε στην λίστα. Η λίστα πρέπει να είναι **ταξινομημένη κατά αύξουσα σειρά** με βάση την τιμή του data. Η συνάρτηση επιστρέφει την κεφαλή της λίστας. Η main θα διαβάζει επαναληπτικά από το πληκτρολόγιο θετικούς αριθμούς (> 0) και θα τα εισάγει στην λίστα καλώντας την συνάρτηση insertElement. Ο χρήστης θα εισάγει τα δεδομένα με αύξουσα σειρά από την κονσόλα. Μόλις δοθεί το 0, τότε η main σταματάει να ζητάει στοιχεία και εκτυπώνει την λίστα (ερώτημα γ).
Προσοχή:

- Η συνάρτηση πρέπει να ελέγχει την τιμή που επιστρέφει η κλήση συστήματος για την δυναμική δέσμευση μνήμης. Σε περίπτωση που αυτή είναι μηδέν, τότε εκτυπώνει κατάλληλο μήνυμα λάθους και το πρόγραμμα τερματίζει.
- Να προσέχετε τις ειδικές περιπτώσεις όπως την εισαγωγή σε κενή λίστα.

β) (4 μονάδες) Να δημιουργηθεί μια συνάρτηση **nodeL * mergeLists(nodeL *head1, nodeL *head2)** που θα κάνει συγχώνευση των στοιχείων δύο ταξινομημένων λιστών σε μία ταξινομημένη λίστα. Όπου head1 είναι ο δείκτης στην αρχή της πρώτης διασυνδεδεμένης λίστας, και head2 είναι ο δείκτης στην αρχή της δεύτερης διασυνδεδεμένης λίστας. Η συνάρτηση επιστρέφει το head της νέας ταξινομημένης

λίστας που περιέχει τα στοιχεία των δύο λιστών που περάσαμε ως ορίσματα συγχωνευμένα.

Παράδειγμα συγχώνευσης λιστών:
$1^{η}$ λίστα : 1 -> 3 -> 4 -> 6 -> 8
$2^{η}$ λίστα: 2-> 3 -> 5 -> 10
Τελική λίστα: 1 -> 2 ->3 -> 3 -> 4 -> 5->6 ->8 ->10

γ) (0.5 μονάδες) Να γραφεί μία συνάρτηση **void printList (nodeL *head)** όπου head είναι δείκτης στην αρχή της διασυνδεδεμένης λίστας, η οποία θα εκτυπώνει το περιεχόμενο της λίστας στην οθόνη ως εξής:
List: (1 2 2 3 6 12 15)

Το πρόγραμμα θα δημιουργεί αρχικά δύο ταξινομημένες λίστες, χρησιμοποιώντας την συνάρτηση **insertElement** για εισαγωγή στοιχείων στις λίστες.
Μετά πραγματοποιείται η συγχώνευση των δύο λιστών με χρήση της συνάρτησης **mergeLists**. Τέλος η main θα καλέσει την συνάρτηση **printList** μόλις τελειώσει η διαδικασία συγχώνευσης των λιστών για να εκτυπώσει τη συγχωνευμένη λίστα.

Σημείωση

Μία διευκρίνηση για την άσκηση αναδρομής του Lab5. Αναδρομή είναι ο ορισμός μίας συνάρτησης F χρησιμοποιώντας πάλι την ίδια συνάρτηση με διαφορετικές όμως παραμέτρους. Για παράδειγμα ο ορισμός του παραγοντικού ως F(n) = n*F(n-1), n>1 και F(0)=0 είναι μια αναδρομική σχέση, ενώ ο ορισμός F(n) = 1*2*…*(n-1)*n, δεν είναι αναδρομική σχέση. Συνεπώς, στις ασκήσεις του Lab5 οι σωστές αναδρομικές λύσεις θα πρέπει να καλούν την ίδια συνάρτηση με διαφορετικές όμως τιμές στις παραμέτρους $a0, $a1 κοκ. για κάθε κλήση της F κατά την διάρκεια εκτέλεσης του προγράμματος. Λύσεις που δεν βασίζονται σε αυτήν την αρχή και που απλά κάνουν χρήση της εντολής κλήσης *jal F* σαν ένα απλό *goto* δεν θα γίνονται δεκτές.

Είναι σημαντικό η υλοποίηση των ασκήσεων να γίνει με κλήση συναρτήσεων ακριβώς όπως περιγράφει η εκφώνηση.
Κάθε συνάρτηση πρέπει να εξασφαλίζει ότι οι λεγόμενοι Saved Temporary καταχωρητές διατηρούν το περιεχόμενο που είχαν πριν την κλήση της.
Μην κάνετε άσκοπα χρήση της στοίβας. Η χωρίς όρια δέσμευση περιττής μνήμης αποτελεί προγραμματιστικό λάθος.

Θα πρέπει να στέλνετε με email τις λύσεις των εργαστηριακών ασκήσεων σας στους διδάσκοντες στο ce134lab@gmail.com.

Το email σας θα πρέπει να περιέχει ως attachment **<u>ένα zip file</u>** με τον κώδικα σας.

Κάθε διαφορετική άσκηση στην εκφώνηση θα βρίσκεται και σε διαφορετικό asm file. **<u>Το όνομα των asm files θα ΠΡΕΠΕΙ να αρχίζει με το ΑΕΜ σας.</u>**
Για παράδειγμα, το lab2.zip θα περιέχει 3 asm files, ένα για κάθε μία από τις ασκήσεις του lab2, με ονόματα 999_lab2a.asm, 999_lab2b.asm, 999_lab2c.asm για τον φοιτητή με ΑΕΜ 999.

Το email σας θα έχει Subject: CE134, lab N, Section X (N ο αριθμός του lab, N=2 ..., και X=1 έως 7).
Το email σας θα έχει body: το όνομα σας και το ΑΕΜ σας.

Θα πρέπει να στέλνετε το email σας πριν βγείτε από την εξέταση του εργαστηρίου.