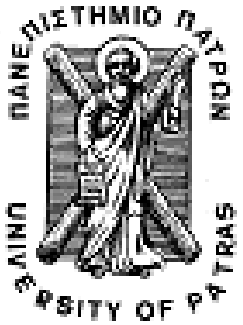
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ



2ο ΣΕΤ ΑΣΚΗΣΕΩΝ

Οι ασκήσεις αυτού του φυλλαδίου καλύπτουν τα παρακάτω θέματα:

- Συναρτήσεις (κεφάλαιο Functions)
- Πίνακες (κεφάλαιο Arrays)
- Δείκτες
- Δομές

Σημείωση: αν χρειάζεται συμπερίληψη της βιβλιοθήκης <math.h>, τότε θα πρέπει η μεταγλώττιση να πραγματοποιηθεί με την οδηγία –lm, δηλαδή η μεταγλώττιση να γίνει ως εξής:

gcc –lm ονομα_αρχείου

Το δεύτερο σύνολο ασκήσεων θα πρέπει να πραγματοποιηθεί τις εβδομάδες 28/11-27/12/16 (για κάθε υποερώτημα ο φοιτητής θα πρέπει να έχει το αντίστοιχο αρχείο κώδικα C αποθηκευμένο στο λογαριασμό του στο diogenis.ceid.upatras.gr το οποίο θα πρέπει να έχει μεταγλωττιστεί χωρίς σφάλματα, ενώ για το σύνολο των ερωτημάτων καλό είναι να ετοιμάζει και ένα αρχείο κειμένου με τις απαντήσεις του) και το επόμενο σύνολο ασκήσεων θα ανακοινωθεί 27/12/2016.

Σε πολλές από τις ασκήσεις θα χρειαστείτε συναρτήσεις από τη βιβλιοθήκη <math.h> της C. Στην περίπτωση αυτή θα πρέπει η μεταγλώττιση να πραγματοποιηθεί με την οδηγία -lm, δηλαδή η μεταγλώττιση να γίνει ως εξής: gcc -lm ...

Άσκηση 1

Να γραφεί ένα πρόγραμμα το οποίο να διαβάζει 100 ακεραίους και να τους αποθηκεύει σε ένα πίνακα. Στη συνέχεια το πρόγραμμα να εμφανίζει πόσες φορές επαναλαμβάνονται οι τιμές των στοιχείων του πίνακα μέσα σε αυτόν. Για παράδειγμα αν τα στοιχεία του πίνακα είναι {5,5,5,5,5} το πρόγραμμα να εμφανίζει 4 (αφού ο αριθμός 5 επαναλαμβάνεται τέσσερις φορές) αν είναι {1, -3, 1, 50, -3} το πρόγραμμα να εμφανίζει 2 (αφού οι αριθμοί 1 και -3 επαναλαμβάνονται από μία φορά), ενώ αν είναι {3, -1, 22, 13, -7} το πρόγραμμα να εμφανίζει 0 (αφού δεν υπάρχει στοιχείο να επαναλαμβάνεται).

Άσκηση 2

Γράψτε ένα πρόγραμμα το οποίο θα δέχεται ως είσοδο ένα πίνακα αλφαριθμητικών. Κάθε αλφαριθμητικό περιέχει ένα σύνολο από λέξεις χωρισμένα από ένα ή περισσότερα κενά. Το πρόγραμμα θα μετράει το μήκος των λέξεων αυτών και θα εμφανίζει το ιστόγραμμα των μηκών με γραφικό τρόπο ως εξής:

```
Word Length     |Number of Occurrences

1               |

2               | *******

3               | **********

4               | ****

5               | ***

6               | *****

7               | ***

8               | ***

9               | *

10              |

11              |

12              |

13              |

diogenis>
```

Επεκτάσεις:

- *Φροντίστε να μην προσμετρούνται τα σημεία στίξης (τελείες, παρενθέσεις κτλ) στο μήκος της λέξης.*

Άσκηση 3

Δημιουργήστε κώδικα που να υλοποιεί το ακόλουθο "παιχνίδι" λέξεων: αρχικά ζητείται από τον πρώτο χρήστη να εισαγάγει μια λέξη. Στη συνέχεια η οθόνη καθαρίζεται και εμφανίζονται χαρακτήρες '_' στη θέση των γραμμάτων της λέξης καθώς και ένας μετρητής που δηλώνει πόσες προσπάθειες απομένουν στο δεύτερο χρήστη, που καλείται να μαντέψει τη λέξη δίνοντας έναν χαρακτήρα κάθε φορά. Αν ο εισαγόμενος χαρακτήρας υπάρχει εμφανίζεται στην κατάλληλη θέση της λέξης, αν δεν υπάρχει ο μετρητής μειώνεται κατά 1 και αν έχει επιλεχθεί ήδη εμφανίζεται ένα ενημερωτικό μήνυμα. Το παιχνίδι τελειώνει όταν βρεθούν όλα τα γράμματα της λέξης ή όταν μηδενιστεί ο μετρητής (οπότε εμφανίζονται τα κατάλληλα μηνύματα στην οθόνη).
Προτείνεται:

1. όλοι οι εισαγόμενοι χαρακτήρες να μετατρέπονται σε μικρά
2. ο έλεγχος ύπαρξης ενός χαρακτήρα σε μια λέξη να γίνεται με την κλήση συνάρτησης η οποία θα λαμβάνει ως ορίσματα τη λέξη και τον χαρακτήρα προς αναζήτηση και θα επιστρέφει 0 αν δεν υπάρχει ή 1 αν υπάρχει.

Υπόδειξη: *Ο καθαρισμός της οθόνης γίνεται με την εντολή system("clear") (Unix) ή system("cls") (DOS).*

Άσκηση 4η

Μία από τις πιο σημαντικές χρήσεις της γλώσσας C είναι στη διαχείριση αλφαριθμητικών, και αυτό γιατί η C παρέχει πολλές ευκολίες στη διαχείρισή τους. Σκοπός της συγκεκριμένης άσκησης είναι η διερεύνηση αυτών των δυνατοτήτων, για τον αποτελεσματικό χειρισμό ενός πίνακα αλφαριθμητικών. Θεωρείστε λοιπόν ένα πίνακα N θέσεων όπου σε κάθε θέση αποθηκεύεται η τιμή μίας μεταβλητής τύπου δομής με δύο μέλη ένα αλφαριθμητικό και μία ακέραια τιμή που υποδηλώνει το μήκος του αλφαριθμητικού. Θεωρείστε ότι κάθε αλφαριθμητικό είναι το καθένα μέγιστου μήκους M-1 και αποτελείται μόνο από αριθμητικούς και αλφαβητικούς χαρακτήρες (όλοι του Αγγλικού αλφαβήτου), χωρίς κενά, με το N και το M να δηλώνονται στο πρόγραμμα σαν συμβολικές σταθερές. Έστω word_table το όνομα του πίνακα.
Τυπικά η δήλωση θα δίνεται ως εξής:

```
struct word_pair
{
char word[M];
int length;
};
struct word_pair word_table[N]
```

Ζητείται να υλοποιηθούν οι εξής συναρτήσεις:

Ζητείται να υλοποιηθούν οι εξής συναρτήσεις:
Α. Μία συνάρτηση που θα ζητάει από τον χρήστη και θα γεμίζει τον πίνακα με *N* αλφαριθμητικά, και για κάθε αλφαριθμητικό θα υπολογίζει και θα αποθηκεύει στην αντίστοιχη θέση το μήκος του. Το μήκος του αλφαριθμητικού θα πρέπει να υπολογίζεται με δική σας συνάρτηση και όχι με συνάρτηση της βιβλιοθήκης της C. Η συνάρτηση να δηλωθεί ως *void initialize(struct word_pair word_table[], int size)* (με *size* συμβολίζεται ο αριθμός των γραμμών του πίνακα). Η είσοδος των συμβολοσειρών να γίνει με αμυντικό προγραμματισμό.

Β. Μία συνάρτηση που θα δέχεται σαν είσοδο δύο αλφαριθμητικά, και θα αντικαθιστά κάθε εμφάνιση του πρώτου αλφαριθμητικού στον πίνακα με το δεύτερο αλφαριθμητικό. Η συνάρτηση να δηλωθεί ως :
void replace(struct word_pair word_table[], int size, char search_string[M], char replacement_string[M]) (με *size* συμβολίζεται ο αριθμός των γραμμών του πίνακα, *search_string* το αλφαριθμητικό που αναζητούμε, και *replacement_string* το αλφαριθμητικό με το οποίο γίνεται η αντικατάσταση) .
Γ. Μία συνάρτηση που δοθείσης ενός αλφαριθμητικού εισόδου θα εντοπίζει και θα εκτυπώνει όλες τις γραμμές του πίνακα που την περιέχουν σαν υποακολουθία. Η συνάρτηση να δηλωθεί σαν *void sub_sebsequence(struct word_pair word_table[], char search_string[M]* (με *size* συμβολίζεται ο αριθμός των γραμμών του πίνακα, με *search_string* το αλφαριθμητικό που αναζητούμε)
Δ. Χρησιμοποιώντας τον παρακάτω αρχικοποιημένο λεξικό αλφαριθμητικών *lexicon* για μετάφραση μεταξύ Γερμανικών και Αγγλικών λέξεων υλοποιήστε μία συνάρτηση που μεταφράζει κάθε αλφαριθμητικό του *word_table* στο αντίστοιχο γερμανικό αν υπάρχει στο λεξικό, διαφορετικά το αφήνει ως έχει. Η συνάρτηση να δηλωθεί ως: *void translate(struct word_table word_table[], int size)* (με *size* συμβολίζεται ο αριθμός των γραμμών του πίνακα)

```
char lexixon[][2][M]={
        {“table”, “tafel”},
        {“word”, “wort”},
        {“car”, “auto”}.
        {“tree”, “baum”},
        {“number”, “anzahl”}.
        {“drive” , “fahern”}
        {“bicycle”, “fahrrad”}
                }
```

Ελέγξτε τον κώδικά σας με ένα κυρίως πρόγραμμα το οποίο θα καλεί τις ανωτέρω συναρτήσεις με την σειρά με την οποία ορίστηκαν.

Υποδείξεις:

Προτείνεται η χρήση των συναρτήσεων

.

- *int isalnum(int ch)* της βιβλιοθήκης <ctype.h> που επιστρέφει μη μηδενική τιμή (αληθή) αν ο χαρακτήρας *ch* είναι αριθμητικό ψηφίο ή γράμμα (πεζό ή κεφαλαίο) του αγγλικού αλφαβήτου, διαφορετικά επιστρέφει 0
- *char *strcpy(char *s, const char *t),* συνάρτηση της βιβλιοθήκης <*string.h*> που αντιγράφει το αλφαριθμητικό *t* στο αλφαριθμητικό *s*.

.

Άσκηση 5
Να υλοποιήσετε τον αλγόριθμο ταξινόμησης BubbleSort. Ο Bubblesort είναι ένας απλός αλγόριθμος ταξινόμησης (http://en.wikipedia.org/wiki/Bubble_sort) που λειτουργεί σαρώνοντας όλη την ακολουθία στοιχείων που πρέπει να διαταχθεί, συγκρίνοντας κάθε ζεύγος γειτονικών στοιχείων και εναλλάσοντάς τα αν είναι σε λάθος σειρά. Το πέρασμα

από την ακολουθία στοιχείων επαναλαμβάνεται έως ότου δεν χρειάζεται να υπάρχουν εναλλαγές στιοιχειων, γεγονός που υποδεικνύει ότι η ακολουθία είναι ταξινομημένη.
Τα στοιχεία τα οποία πρέπει να ταξινομηθούν είναι πραγματικοί αριθμοί σε έναν πίνακα μεγέθους 30. Για δική σας διευκόλυνση να φτιάξετε μία συνάρτηση η οποία θα τυπώνει τον πίνακα και θα τη χρησιμοποιείτε όταν συμβαίνει μία εναλλαγή μεταξύ στοιχείων του πίνακα. Τα στοιχεία του πίνακα θα τα δίνετε εσείς από το πληκτρολόγιο. Μία δεύτερη συνάρτηση μπορεί να είναι αυτή η οποία θα πραγματοποιεί την εναλλαγή μεταξύ δύο στοιχείων του πίνακα.

Ασκηση 6
Να γραφεί πρόγραμμα που θα ζητάει από το χρήστη να του δώσει τις διαστάσεις για να κατασκευάσει ένα δισδιάστατο πίνακα ακεραίων. Μόλις ο χρήστης δώσει τις δύο διαστάσεις, το πρόγραμμα θα δεσμεύει δυναμικά μνήμη για τον πίνακα με χρήση της malloc και θα διαβάζει από τον χρήστη μια ακέραια τιμή για κάθε στοιχείο του πίνακα. Στη συνέχεια, να κατασκευαστεί συνάρτηση, η οποία θα καλείται από το πρόγραμμα και να εκτελεί τα παρακάτω:
1. θα υπολογίζει το άθροισμα κάθε στήλης του πίνακα και θα το αποθηκεύει σ' ένα νέο πίνακα.
2. να επιστρέφει το νέο πίνακα στο κύριο πρόγραμμα.
Το κύριο πρόγραμμα θα πρέπει να εκτυπώνει τα στοιχεία του νέου πίνακα που δημιουργήθηκε απ' τη συνάρτηση.

Άσκηση 7
Να γράψετε πρόγραμμα το οποίο να μπορεί να λύσει ένα σύνολο εξισώσεων 3 x 3 όπως:

$2x_1 + 2x_2 -7x_3 = 3$

$-2x_1+6x_2-12x_3=-5$

$x_1-2x_2+5x_3=1$

Άσκηση 8
Γράψτε πρόγραμμα που να υπολογίζει όλους τους πρώτους αριθμούς μεταξύ του 1 και του N χρησιμοποιώντας το κόσκινο του Ερατοσθένη.
Τα βήματα του αλγορίθμου είναι:

1.καταγράφουμε όλους τους αριθμούς από το 2 μέχρι το N σε μια λίστα
2.για κάθε πρώτο που συναντάμε στη λίστα, διαγράφουμε από τη λίστα όλα τα πολλαπλάσιά του
3.αν ο πρώτος αριθμός που συναντάμε στη λίστα είναι μικρότερος της τετραγωνικής ρίζας του N, τότε επαναλαμβάνουμε το βήμα 2, διαφορετικά η λίστα περιλαμβάνει μόνο πρώτους αριθμούς

Άσκηση 9
Θεωρείστε ότι δίδεται ο εξής ορισμός δομής:

```
struct node
{
int data;
```

struct node *next;
};

Με βάση τον ορισμό αυτό, το ζητούμενο είναι να σχεδιάσετε και να υλοποιήσετε μια συνδεδεμένη λίστα στοιχείων που ικανοποιεί τον κανόνα πρόσβασης First-In-First-Out, με άλλα λόγια μια δομή ουράς (queue). Στις δομές αυτές το στοιχείο το οποίο εντέθηκε πρώτο στη λίστα (με μία πράξη Insert) είναι και αυτό που θα διαγραφεί πρώτο (με μία πράξη Delete). Αυτή η λογική πράξεων μπορεί να υλοποιηθεί αν η ένθεση στοιχείων γίνεται στο τέλος της λίστας, ενώ η διαγραφή αφαιρεί στοιχεία από την αρχή της λίστας.

Γράψτε συναρτήσεις σύμφωνα με τα παρακάτω πρότυπα, καθώς και ένα πρόγραμμα επίδειξης

- **void Insert(struct node **headRef, int newData);**
- **int Delete(struct node **headRef),**
- **int GetNth(struct node *head, int index);**
- **int Count(struct node *head, int searchFor)**

Η συνάρτηση Insert() δέχεται ως όρισμα ένα δείκτη στον δείκτη που αντιστοιχεί στην κεφαλή της λίστας. Δεσμεύει μνήμη, με κλήση της malloc(), για μια δομή του τύπου struct node, αποθηκεύει σ'αυτήν την ακέραια τιμή newData (στο πεδίο data), και συνδέει κατάλληλα την νέα δομή (βάσει του πεδίου next) ώστε η λίστα να ικανοποιεί τον κανόνα πρόσβασης First-In-First-Out, συνεπώς ενθέτει το νέο στοιχείο στο τέλος της λίστας. Η συνάρτηση Delete() δέχεται ως όρισμα ένα δείκτη στον δείκτη που αντιστοιχεί στην κεφαλή της λίστας διαγράφει και επιστρέφει ένα στοιχείο της λίστας έτσι ώστε να ικανοποιείται ο κανόνας First In First Out, συνεπώς διαγράφει και επιστρέφει το πρώτο στοιχείο της λίστας. Η GetNth() δέχεται ως όρισμα τον δείκτη που αντιστοιχεί στην κεφαλή της λίστας, και την θέση (ως ακέραια τιμή που ξεκινά από το 0) ενός στοιχείου. Εφόσον η λίστα περιέχει στοιχείο στην καθοριζόμενη θέση, η συνάρτηση επιστρέφει την τιμή του πεδίου data για το στοιχείο αυτό, διαφορετικά εκτυπώνει κατάλληλο μήνυμα και επιστρέφει την τιμή 0.

Η Count() δέχεται ως όρισμα τον δείκτη που αντιστοιχεί στην κεφαλή της λίστας, και μια ακέραια τιμή. Εξετάζει ένα-προς-ένα όλα τα στοιχεία της λίστας και επιστρέφει σε πόσα από αυτά το πεδίο data ισούται με την αναζητούμενη τιμή (επιστρέφει 0 εάν η τιμή δεν εμφανίζεται στην λίστα).

Γράψτε την main() η οποία εμφανίζει μενού επιλογών στο χρήστη:

1. Insert
2. Delete
2. Get-Nth
3. Count
4. Έξοδος

Ανάλογα με την επιλογή του χρήστη ζητούνται τα απαραίτητα στοιχεία (π.χ. για την πράξη Insert ζητείται από τον χρήστη η ακέραια τιμή newdata) και καλείται η κατάλληλη συνάρτηση.