Παρατηρήσεις κι επεξηγήσεις για το lab9

Για **μεταγλώττιση**, **στοίχιση/κενά**, **ονόματα** και **έξοδο** ισχύουν τα ίδια με προηγούμενες ασκήσεις. Προσοχή να είναι περιγραφικά και τα ονόματα των συναρτήσεων. Ονόματα όπως function1 είναι απαράδεκτα.

Είσοδος

Στην άσκηση της Πέμπτης ενδείκνυται η χρήση sprintf για τη δημιουργία κατάλληλου format string για τη scanf. Στην άσκηση της Παρασκευής προσέξτε τη δεύτερη παράμετρο της fgets που είναι το μέγεθος του πίνακα στον οποίο θα αποθηκευτεί το string, άρα πρέπει να είναι όσο και το πλήθος των στηλών

Ορισμός και κλήσεις συναρτήσεων:

Οι συναρτήσεις παίρνουν ως παραμέτρους τα μεγέθη των πινάκων. Παρόλο που στις ασκήσεις αυτής της εβδομάδας δεν ήταν απαραίτητο κάτι τέτοιο, θέλαμε να το δείτε γιατί σε αρκετές εφαρμογές ο πίνακας που περνάμε σε μια συνάρτηση έχει έγκυρα δεδομένα μόνο στα πρώτα x κελιά του. Σε αυτή την περίπτωση περνάμε αυτό το x ως παράμετρο ώστε μέσα στη συνάρτηση να μη γίνει προσπέλαση του πίνακα πέρα του x-οστού στοιχείου.

Η πρώτη συνάρτηση της Πέμπτης θα μπορούσε να είναι

```
void read_data(char words[], int size);
```

και την καλούμε;

```
read_data(words, SIZE);
```

Στο σώμα της συνάρτησης πρέπει να χρησιμοποιούμε την παράμετρο size όπου διατρέχουμε τον πίνακα words, και όχι το SIZE.

Τα ίδια ισχύουν για την άσκηση της Παρασκευής. Επιπλέον, σημειώστε πως όταν η τυπική παράμετρος μιας συνάρτησης είναι πίνακας με περισσότερες από 1 διάσταση, τότε πρέπει οι διαστάσεις από τη δεύτερη και μετά να καταγράφονται στο prototype, για παράδειγμα:

```
void init(char words[][COLS], int rows, int cols);
```

Ορθότητα

Θα αναλύσουμε την άσκηση της Πέμπτης. Η άσκηση της Παρασκευής είναι παρόμοια όσον αφορά την τοποθέτηση των δεικτών και τη διάτρεξη του πίνακα δεικτών.

Ας υποθέσουμε ότι ο πίνακας από λέξεις είναι ο `char words[SIZE]` και ο πίνακας δεικτών ο `char *ptrs[SIZE/2]` Ενας πίνακας μπορεί να αρχικοποιηθεί την ώρα που δηλώνεται με χρήση λίστας τιμών, διαφορετικά αρχικοποιείται με χρήση επανάληψης. Σε αυτή την άσκηση, εφόσον η αρχική δήλωση των πινάκων γίνεται στη main και η αρχικοποίηση σε άλλες συναρτήσεις, θα πρέπει αναγκαστικά να χρησιμοποιήσουμε επανάληψη.

Για την αρχικοποίηση του words θέτουμε κάθε κελί του σε '\0'. Για την αρχικοποίηση του ptrs θέτουμε κάθε κελί του σε NULL. Παρόλο που το NULL είναι ίσο με μηδέν, είναι καλύτερα να χρησιμοποιούμε NULL και όχι 0 για να είναι ξεκάθαρο ότι αναφερόμαστε σε δείκτες.

Για το γέμισμα του words με τις επιμέρους λέξεις, πρέπει να προσέξουμε τα εξής:

1. Δε γνωρίζουμε πόσες λέξεις θα αποθηκευτούν τελικά στον πίνακα (πιθανώς και καμία), επομένως χρησιμοποιούμε while. Το πιο βολικό είναι μια while(1) μέσα στην οποία θα κάνουμε ότι ελέγχους χρειάζεται και θα βγαίνουμε με break όποτε χρειαστεί.
2. Διαβάζουμε μια λέξη και την αποθηκεύουμε σε προσωρινό πίνακα ώστε πρώτα να ελέγξουμε αν μπορεί η λέξη να εισαχθεί στον words.
   a. Αρχικά ελέγχουμε αν η λέξη που διαβάσαμε είναι η "##quit##". Αυτό γίνεται με strcmp. Αν η

strcmp επιστρέψει 0, τότε κάνουμε break. Σημειώστε εδώ πως μπορείτε είτε να χρησιμοποιήσετε απευθείας το "##quit##" ως παράμετρο στην strcmp, είτε να το ορίσετε με κάποιο #define και να χρησιμοποιήσετε το όνομα, είτε να το αποθηκεύσετε σε κάποια μεταβλητή-πίνακα και να χρησιμοποιήσετε το όνομα αυτής στην strcmp. Στην τελευταία περίπτωση, μπορείτε απλά να δηλώσετε char terminator[] = {"##quit##"}; Δεν υπάρχει λόγος να δυσκολεύετε τη ζωή σας γράφοντας char terminator[] = {'#', '#', 'q', 'u', κτλ.

b. Αν η λέξη δεν είναι "##quit##", τότε πρέπει να δούμε αν χωράει στον πίνακα words. Για να το βρούμε αυτό, πρέπει να γνωρίζουμε πόσες θέσεις του πίνακα είναι κατειλημένες (ή ισοδύναμα, πόσες είναι ελεύθερες). Αρκεί να χρησιμοποιήσουμε μια ακέραια μεταβλητή την οποία ανανεώνουμε κατάλληλα. Προσοχή στον έλεγχο να "μετρήσουμε" και τη θέση που θα χρειαστεί για το '\0'. Την ίδια μεταβλητή μπορούμε να χρησιμοποιήσουμε για να αντιγράψουμε τη νέα μας λέξη στο words εφόσον χωράει (δείτε #3). Αν η λέξη δε χωράει, κάνουμε break.

3. Αν περάσουμε τους ελέγχους, τότε πρέπει να αντιγράψουμε τη λέξη μας στο words, μετά το '\0' της προηγούμενης λέξης που είχε τοποθετηθεί. Αυτό θα γίνει με strcpy με την αντιγραφή να ξεκινάει από το κατάλληλο σημείο. Ας υποθέσουμε ότι έχουμε την ακέραια μεταβλητή lengthSoFar στην οποία κρατάμε το πλήθος κελιών του words που έχουμε γεμίσει με λέξεις. Αρχική τιμή είναι φυσικά 0. Κάθε φορά που είμαστε έτοιμοι να αντιγράψουμε τη λέξη μας στο words, γράφουμε

   `strcpy(words+lengthSoFar, str);` ή ισοδύναμα

   `strcpy(&words[lengthSoFar], str);`

   Μετά (και <u>όχι</u> πριν) ανανεώνουμε το lengthSoFar για να είμαστε έτοιμοι για την επόμενη λέξη:

   `lengthSoFar += strlen(str)+1;`

Για την τοποθέτηση των δεικτών πρέπει να προσέξουμε τα εξής:

1. Οι δείκτες που θέτουμε είναι σε διαδοχικές θέσεις του πίνακα. Ας υποθέσουμε ότι ο πίνακας words περιέχει διαδοχικά τις λέξεις That, is και fun, με '\0' στο τέλος της κάθε μίας. Τότε θα πρέπει να αποθηκεύσουμε τη διεύθυνση του T (&words[0]) στη θέση 0 του πίνακα δεικτών, τη διεύθυνση του i (&words[5]) στη θέση 1 του πίνακα δεικτών και τη διεύθυνση του f (&words[8]) στη θέση 2 του πίνακα δεικτών. Βλέπουμε λοιπόν ότι οι θέσεις του πινακα δεικτών δεν έχουν καμία σχέση με το τις θέσεις των χαρακτήρων του πίνακα words. Αρα, θα χρειαστούμε μια διαφορετική μεταβλητή για να διατρέξουμε τις θέσεις του πίνακα δεικτών, την οποία θα αυξάνουμε μόνο όποτε θέτουμε ένα δείκτη. Αν υποθέσουμε ότι το μέγεθος του πίνακα words είναι wsize, τότε :

```
int k = 0;
ptrs[k++] = &words[0]; // τοποθέτηση του πρώτου δείκτη στην αρχή του words.
for (i=0; i < wsize-1; i++) { // wsize-1 για να μη βγούμε εκτός πίνακα με το i+1
    if (words[i] == '\0' && words[i+1] != '\0') {
        ptrs[k++] = &words[i+1];
    }
}
```

Τέλος για τη διάτρεξη του πίνακα δεικτών και την εκτύπωση των λέξεων στις οποίες δείχνουν, θα πρέπει να προσέξουμε να σταματήσουμε την επανάληψη όταν συναντήσουμε δείκτη NULL.