ΣΕΤ ΑΣΚΗΣΕΩΝ 3

ΕΡΓΑΣΤΗΡΙΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι, ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2013-2014

Προθεσμία: 7/1/2014, 22:00

Περιεχόμενα

Πριν ξεκινήσετε (ΔΙΑΒΑΣΤΕ ΑΥΤΗ ΤΗΝ ΕΝΟΤΗΤΑ!!)

Διαβάστε ΟΛΗ την εκφώνηση προσεκτικά και "σχεδιάστε" το πρόγραμμά σας στο χαρτί. Είναι ιδιαίτερα σημαντικό να διαβάσετε ολόκληρη την εκφώνηση πριν ξεκινήσετε να γράφετε κώδικα γιατί κάποιες πληροφορίες δε δίνονται με γραμμική σειρά.

Για κάθε στάδιο, αποφασίστε τι μεταβλητές θα χρειαστείτε, τι ονόματα θα τους δώσετε, αν χρειάζονται σταθερές κι αν ναι για ποιες ποσότητες, τι συναρτήσεις θα ορίσετε και τι δομές θα χρησιμοποιήσετε για κάθε λειτουργία.

Μη διστάζετε να ζητήσετε βοήθεια! Μπορείτε να χρησιμοποιήσετε κατά προτίμηση το forum προγραμματισμού (http://courses.inf.uth.gr/coding/?page_id=143) και μόνο αν είναι απαραίτητο email (π.χ. αν πραγματικά επιβάλλεται να στείλετε κάποιο κομμάτι κώδικα μαζί με το μήνυμά σας).

Μην προσπαθήσετε να γράψετε το πρόγραμμα μονομιάς. Δηλώστε τις συναρτήσεις σας, αφήστε τα σώματά τους είτε κενά είτε να επιστρέφουν κάτι συμβολικό για να κάνει compile το πρόγραμμα, και μετά γράψτε τη main. Αφού βεβαιωθείτε ότι το πρόγραμμα τρέχει μέχρι αυτό το σημείο, αρχίστε να προσθέτετε κώδικα σε μία-μία συνάρτηση. Προχωράτε στην επόμενη μόνο εφόσον βεβαιωθείτε ότι δουλεύουν όλα σωστά μέχρι το σημείο που έχετε υλοποιήσει.

ΠΡΟΣΟΧΗ: Σας δίνουμε συγκεκριμένες οδηγίες για κάποιες συναρτήσεις, αλλά θα πρέπει να σκεφτείτε και μόνοι σας τι άλλες λειτουργίες του προγράμματος είναι καλύτερα να γίνουν σε ξεχωριστή συνάρτηση ώστε να είναι πιο ευανάγνωστο και συντηρήσιμο το πρόγραμμά σας.

Η εργασία αυτή μπορεί να γίνει σε ομάδες μέχρι 2 ατόμων. Δεν είναι απαραίτητο να συνεργαστείτε με το ίδιο άτομο που κάνατε τα εβδομαδιαία εργαστήρια ή το πρώτο σετ ασκήσεων. Μπορείτε να συζητάτε τις ασκήσεις με συμφοιτητές σας αλλά δεν επιτρέπεται η ανταλλαγή κώδικα με οποιοδήποτε τρόπο.

Ξεκινήστε νωρίς! Ο προγραμματισμός είναι πάντα ΠΟΛΥ πιο χρονοβόρος από ό,τι περιμένετε.

Εκπρόθεσμες ασκήσεις δε γίνονται δεκτές.

'Ασκηση 1: Ταίριασμα ονομάτων με wildcards

Εισαγωγή

Ένα wildcard είναι ένας ειδικός χαρακτήρας (* ή ?) ο οποίος χρησιμοποιείται για να αναπαραστήσει έναν ή περισσότερους χαρακτήρες κατά την αναζήτηση αρχείων και αλλού. Ο αστερίσκος αναπαριστά μηδέν ή περισσότερους χαρακτήρες, ενώ το ερωτηματικό ακριβώς έναν χαρακτήρα. Για παράδειγμα, αν σε ένα φάκελο βρίσκονται αρχεία με ονόματα

```
page1.html page2.html page3.html image1.jpg image2.jpg agent page
```

τότε:

η έκφραση `page?.html` αναφέρεται στα `page1.html page2.html page3.html`

η έκφραση `*age*` αναφέρεται σε όλα τα αρχεία

η έκφραση `*.html` αναφέρεται σε όλα τα αρχεία με κατάληξη .hmtl

η έκφραση `*.*` αναφέρεται σε όλα τα αρχεία εκτός των `agent, page` (τα οποία δεν περιέχουν τελεία)

η έκφραση `?age*` αναφέρεται στα `page1.html page2.html page3.html page`

Σε αυτό το σετ ασκήσεων θα γράψετε ένα πρόγραμμα το οποίο διαβάζει μια σειρά από ονόματα αρχείων και μετά μια σειρά από εκφράσεις με ακριβώς ένα wildcard στην κάθε μία έκφραση. Για κάθε έκφραση, το πρόγραμμα ψάχνει να βρει ποια από τα ονόματα αρχείων ταιριάζουν.

Ακολουθούν λεπτομερείς οδηγίες για το πώς πρέπει να λειτουργεί το πρόγραμμά σας. ΜΗΝ προσπαθήσετε να γράψετε όλο το πρόγραμμα σε ένα βήμα γιατί θα κάνετε λάθη και θα σας πάρει πολύ περισσότερο χρόνο.

Αποθηκεύστε το τελικό πρόγραμμα σε αρχείο με όνομα `hw3.c` .

Δομές δεδομένων

Ορίστε ένα struct με όνομα `nameInfoT` και δύο πεδία, `name` και `replacement`. Και τα δύο είναι δείκτες σε χαρακτήρα. Το `name` θα δείχνει στην αρχή μιας συμβολοσειράς που περιέχει το όνομα ενός αρχείου. Το `replacement` θα δείχνει στην αρχή μιας συμβολοσειράς που περιέχει το κομμάτι του ονόματος του αρχείου που αντιστοιχεί σε ένα wildcard.

Συνάρτηση *main*

Η main εκτελεί τις εξής βασικές λειτουργίες:

- Καλεί μια συνάρτηση `readNames` η οποία διαβάζει τα ονόματα των αρχείων και τα αποθηκεύει σε κατάλληλο, δυναμικά δεσμευμένο πίνακα από `nameInfoT`.
- Διαβάζει επαναληπτικά μια σειρά από συμβολοσειρές, κάθε μία από τις οποίες πρέπει να περιέχει ακριβώς ένα wildcard.
  - Αν η έκφραση που διαβάστηκε δεν περιέχει κανένα wildcard ή περιέχει περισσότερα από ένα, τότε εκτυπώνει το μήνυμα `Skipping X` όπου X η έκφραση, ένα χαρακτήρα αλλαγής γραμμής και προχωρά να διαβάσει την επόμενη έκφραση.
  - Αν η έκφραση περιέχει ακριβώς ένα wildcard, τότε την εκτυπώνει στην οθόνη ακολουθούμενη από χαρακτήρα αλλαγής γραμμής.
    - Καλεί μια συνάρτηση `findAllReplacements` η οποία περιγράφεται παρακάτω στην εκφώνηση.
    - Καλεί μια συνάρτηση `printAllReplacement` η οποία περιγράφεται παρακάτω στην εκφώνηση..
    - Καλεί μια συνάρτηση `clearAllReplacements` η οποία περιγράφεται παρακάτω στην εκφώνηση.
- Η επαναληπτική ανάγνωση εκφράσεων σταματά όταν διαβαστεί η λέξη `END`.
- Καλεί μια συνάρτηση η οποία απελευθερώνει όλη τη δυναμικά δεσμευμένη μνήμη του προγράμματος.

Συνάρτηση `readNames`

Η συνάρτηση αυτή πρέπει να εκτελεί τις παρακάτω λειτουργίες

- Κατασκευάζει δυναμικά ένα πίνακα από `nameInfoT` μεγέθους 1.
- Διαβάζει από το πληκτρολόγιο μια σειρά από συμβολοσειρές (ας πούμε ότι αντιστοιχούν σε ονόματα αρχείων). Κάθε όνομα έχει το πολύ 32 χαρακτήρες.
- Κάθε όνομα αποθηκεύεται σε μια θέση του πίνακα, στο πεδίο `name`, κι έτσι ώστε να έχει δεσμευθεί γι αυτό *ακριβώς* όση μνήμη χρειάζεται. Σε αυτό το στάδιο δεν υπάρχουν πληροφορίες για το πεδίο `replacement`.
- Δεν είναι γνωστό εκ των προτέρων πόσα είναι τα ονόματα. Κάθε φορά που γεμίζει ο πίνακας από `nameInfoT`, επαναδεσμεύεται διπλάσιο μέγεθος γι αυτόν.
- Η ανάγνωση ονομάτων τερματίζει όταν διαβαστεί η λέξη END (η οποία *δεν* αποθηκεύεται στον πίνακα). Σε αυτό το σημείο, το μέγεθος του πίνακα προσαρμόζεται ώστε να είναι ακριβώς τόσο όσο το πλήθος ονομάτων που είναι αποθηκευμένα σε αυτό.
- Η συνάρτηση επιστρέφει τη διεύθυνση της αρχής του πίνακα από nameInfoT. Επιπλέον, "επιστρέφει" μέσω των παραμέτρων της το μέγεθος του πίνακα.

Το πρόγραμμά σας πρέπει να είναι γραμμένο με τέτοιο τρόπο ώστε αν αλλάξει το μέγιστο δυνατό μέγεθος

ενός ονόματος, να μπορούν να γίνουν εύκολα και γρήγορα οι κατάλληλες αλλαγές στον κώδικα.

Συνάρτηση `findAllReplacements`

Η συνάρτηση αυτή παίρνει ως παραμέτρους τον πίνακα από `nameInfoT`, το μέγεθός του και την έκφραση και εκτελεί τις παρακάτω λειτουργίες:

- Διατρέχει τον πίνακα ονομάτων και για κάθε ένα όνομα
  - καλεί μια βοηθητική συνάρτηση η οποία παίρνει ως παραμέτρους το όνομα και την έκφραση, βρίσκει αν η έκφραση μπορεί να αναπαραστήσει το όνομα, κι αν ναι, κατασκευάζει κι επιστρέφει μια νέα συμβολοσειρά η οποία περιέχει το κομμάτι του ονόματος του αρχείου που αντιστοιχεί στο wildcard της έκφρασης. Πρέπει να σκεφτείτε πώς θα χειριστείτε την περίπτωση που δεν είναι δυνατό να αναπαρασταθεί το όνομα από την έκφραση. Επίσης, ίσως σας φανεί πιο πρακτικό να χρησιμοποιήσετε περισσότερες βοηθητικές συναρτήσεις (πχ μία για την περίπτωση που το wildcard είναι αστερίσκος και μία για την περίπτωση που είναι ερωτηματικό).
  - Αποθηκεύει τη συμβολοσειρά στο πεδίο `replacement` κάθε στοιχείου του πίνακα.

Συνάρτηση `printAllReplacements`

Η συνάρτηση αυτή παίρνει ως παραμέτρους ό,τι κρίνετε ότι απαιτείται. Διατρέχει τον πίνακα ονομάτων και για κάθε ένα όνομα για το οποίο βρέθηκε replacement εκτυπώνει ένα χαρακτήρα tab, το μήνυμα `"word: X, replacement: Y"` όπου X το όνομα και Y το replacement, κι ένα χαρακτήρα αλλαγής γραμμής.

Συνάρτηση `clearAllReplacements`

Η συνάρτηση αυτή παίρνει ως παραμέτρους ό,τι κρίνετε ότι απαιτείται. Διατρέχει τον πίνακα ονομάτων και για κάθε ένα όνομα για το οποίο βρέθηκε replacement διαγράφει το replacement απελευθερώνοντας κατάλληλα τη μνήμη που είχε δεσμευθεί γι αυτό.

Γενικές απαιτήσεις

- Απαγορεύεται αυστηρά η χρήση καθολικών μεταβλητών και goto.
- Σε περιπτώσεις αποτυχίας στη δέσμευση μνήμης, το πρόγραμμά σας πρέπει να εκτυπώνει κατάλληλο μήνυμα λάθους και είτε να επιστρέφει από τη συνάρτηση όπου έγινε το λάθος, είτε να τερματίζει άμεσα. Δικαιολογήστε την απόφασή σας σε σχόλια.
- Σε κάθε περίπτωση, το πρόγραμμα πρέπει να απελευθερώνει όλη τη δυναμικά δεσμευμένη μνήμη πριν τερματίσει.
- Τόσο ο πίνακας από struct όσο και οι συμβολοσειρές που είναι αποθηκευμένες σε αυτόν πρέπει να έχουν ακριβώς όσο μέγεθος απαιτείται (όχι περισσότερο και φυσικά όχι λιγότερο)
- Το πρόγραμμα πρέπει να έχει περιγραφικά ονόματα μεταβλητών και σχόλια όπως περιγράφεται στο σχετικό φυλλάδιο στη σελίδα του εργαστηρίου.

Πώς να παραδώσετε τη δουλειά σας

Προσθέστε σε σχόλια στην αρχή του κάθε αρχείου με κώδικα τα πλήρη ονόματα και ΑΜ των μελών της ομάδας. Παρακαλούμε να γράφετε τα σχόλια ΜΟΝΟ με λατινικούς χαρακτήρες.

Κατασκευάστε ένα φάκελο με όνομα `hw3_epwnumo1_AM1_epwnumo2_AM2` και αντιγράψτε μέσα σε αυτόν το `hw3.c`

Πηγαίνετε στο φάκελο μέσα στον οποίο βρίσκεται το `hw3_epwnumo1_AM1_epwnumo2_AM2` που κατασκευάσατε και γράψτε την παρακάτω εντολή:

```
tar czvf hw3_epwnumo1_AM1_epwnumo2_AM2.tgz hw3_epwnumo1_AM1_epwnumo2_AM2
```

Στείλτε email:

- στη διεύθυνση **`ce120lab@gmail.com`**
- αντίγραφο (**CC**) στο άλλο μέλος της ομάδας σας
- θέμα (subject) **`CE120 hw3`**
- και επικολλημένο αρχείο το `hw3_epwnumo1_AM1_epwnumo2_AM2.tgz`

Το πρόγραμμά σας θα βαθμολογηθεί ως προς:

- ορθότητα λειτουργίας
- σωστή επιλογή και χρήση δομών ελέγχου κι επανάληψης
- σωστή χρήση συναρτήσεων για συμβολοσειρές
- σωστή χρήση δεικτών και δυναμικά δεσμευμένης μνήμης
- σωστό ορισμό και χρήση συναρτήσεων, δημιουργία δικών σας συναρτήσεων
- αναγνωσιμότητα (ονόματα, στοίχιση, σχολιασμός, μορφή κώδικα)
- φορμαρισμένη είσοδο/έξοδο και συμμόρφωση με τις προδιαγραφές.