ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ, ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ Ι, ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2012-2013

ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ

Ποιος πρέπει να ολοκληρώσει αυτή την εργασία?

Φοιτητές έτους >2 που δεν ολοκλήρωσαν επιτυχώς την εργασία του ακαδημαϊκού έτους 2011-2012.

Φοιτητές έτους==2 που δεν ολοκλήρωσαν επιτυχώς τα εργαστήρια του ακαδημαϊκού έτους 2011-2012

Η εργασία μπορεί να γίνει είτε ατομικά είτε σε ομάδα των δύο ατόμων.

Προθεσμία

Κυριακή, 6 Ιανουαρίου 2013, 23:59

Διαδικαστικά

Η προθεσμία παράδοσης είναι τελική - εκπρόθεσμες εργασίες δε γίνονται δεκτές.

Για να παραδώσετε την εργασία σας:
Δημιουργήστε ένα φάκελο με όνομα (με λατινικούς χαρακτήρες)

επώνυμο_όνομα_ΑΕΜ_email

ή

επώνυμο1_όνομα1_ΑΕΜ1_email1_επώνυμο2_όνομα2_ΑΕΜ2_email2

αν η εργασία έχει γίνει σε ομάδα.

Μέσα σε αυτό το φάκελο **αντιγράψτε το αρχείο C που γράψατε**.

Δημιουργήστε ένα αρχείο ZIP ή TGZ το οποίο περιέχει το φάκελλο που κατασκευάσατε.

Η παράδοση γίνεται με email στο **ce120lab@gmail.com**. Μόνο το ένα μέλος της ομάδας πρέπει να στείλει την άσκηση. Το email πρέπει να συντάσσεται ως εξής:

Θέμα (Subject): CE120 project
Πρέπει να γράψετε το θέμα **ακριβώς** με αυτό τον τρόπο για να περάσει από τα κατάλληλα φίλτρα.

CC: το email του δεύτερου μέλους της ομάδας

Περιεχόμενο (Text): Τα ονόματα και ΑΕΜ των μελών της ομάδας

Επισυνάψεις (attachments): Το ZIP ή TAR ή TGZ αρχείο που δημιουργήσατε

Σύντομη περιγραφή

Θα υλοποιήσετε ένα παιχνίδι κρεμάλας ανάμεσα στον υπολογιστή και ένα παίκτη όπου όμως ο υπολογιστής "κλέβει".

Περιγραφή παιχνιδιού

Στην αρχή του παιχνιδιού ο παίκτης προσδιορίζει πόσα γράμματα επιθυμεί να έχει η λέξη που θα μαντέψει, πόσους γύρους θα έχει το παιχνίδι καθώς και το όνομα του αρχείου που περιέχει το λεξικό.

Στο κλασικό παιχνίδι κρεμάλας, ο υπολογιστής διαλέγει μια λέξη και την εμφανίζει στην οθόνη με παύλες αντί για γράμματα. Σε κάθε γύρο ο παίκτης μαντεύει ένα γράμμα, κι αν αυτό υπάρχει στη λέξη, τότε αντικαθίστανται η αντίστοιχη παύλα. Ο παίκτης κερδίζει αν προλάβει να μαντέψει όλα τα γράμματα στους διαθέσιμους γύρους.

Στο παιχνίδι που θα γράψετε εσείς όμως, ο υπολογιστής κλέβει. Για παράδειγμα, ας υποθέσουμε ότι ο παίκτης έχει ήδη μαντέψει όλα τα γράμματα εκτός από ένα, έχει μείνει μόνο ένας γύρος, και μέχρι στιγμής η λέξη είναι `as-en`. Υπάρχουν μόνο δύο λέξεις στο λεξικό που ταιριάζουν: `ashen` και `aspen`. Το πρόγραμμα πρέπει να είναι έτσι γραμμένο ώστε αν ο παίκτης μαντέψει `h`, ο υπολογιστής θα ισχυριστεί ότι η λέξη που είχε διαλέξει αρχικά ήταν `aspen`, ενώ αν ο παίκτης μαντέψει `p`, ο υπολογιστής θα ισχυριστεί ότι η λέξη ήταν `ashen`. Έτσι, ο παίκτης δεν έχει ελπίδα να κερδίσει.

Στην αρχή του παιχνιδιού, ο υπολογιστής γνωρίζει μόνο πόσα γράμματα πρέπει να έχει η λέξη, αλλά δεν έχει επιλέξει κάποια. Τη στιγμή που ο παίκτης μαντεύει το πρώτο του γράμμα, ο υπολογιστής χωρίζει όλες τις λέξεις σε **κατηγορίες** ("κλάσεις ισοδυναμίας") ανάλογα με την ύπαρξη ή όχι του γράμματος και τις θέσεις στις οποίες μπορεί να εμφανιστεί. Για παράδειγμα, ας υποθέσουμε ότι το λεξικό έχει μόνο τις παρακάτω 10 λέξεις:

`arts, dear, deer, feel, flew, pear, pend, perk, poor, rear`

Ο παίκτης στον πρώτο γύρο μαντεύει `e`. Ο υπολογιστής πρέπει να εμφανίσει τα σημεία της λέξης στα οποία υπάρχει `e`. Πώς θα το κάνει αυτό αν δεν έχει διαλέξει ακόμη τη λέξη? Θα χωρίσει τις διαθέσιμες λέξεις σε κατηγορίες ανάλογα με τις θέσεις στις οποίες υπάρχουν `e`. Οι κατηγορίες είναι:

`----` (λέξεις χωρίς `e`: `arts, poor`)

`-e--` (λέξεις όπου το `e` εμφανίζεται μόνο στη 2η θέση: `dear, pear, pend, perk, rear`)

`-ee-` (λέξεις όπου το `e` εμφανίζεται στη 2η και στην 3η θέση: `deer, feel`)

`--e-` (λέξεις όπου το `e` εμφανίζεται στην 3η θέση: `flew`)

Κάθε κατηγορία χαρακτηρίζεται από έναν **αντιπρόσωπο**. Για παράδειγμα ο αντιπρόσωπος της πρώτης κατηγορίας είναι η λέξη "`----`". Ο υπολογιστής μπορεί τώρα να διαλέξει μια *κατηγορία* λέξεων, και να εμφανίσει το `e` στις αντίστοιχες θέσεις. Υπάρχουν αρκετές στρατηγικές, αλλά θα ακολουθήσουμε την πιο απλή: κάθε φορά, ο υπολογιστής διαλέγει την πολυπληθέστερη κατηγορία. Επομένως, στο συγκεκριμένο παράδειγμα θα επιλέξει την κατηγορία με αντιπρόσωπο `-e--`. Η τελική λέξη θα είναι μία εκ των `dear, pear, pend, perk` αλλά φυσικά ακόμη δεν έχει επιλεγεί μια συγκεκριμένη από αυτές.

Ας υποθέσουμε τώρα ότι στο δεύτερο γύρο ο παίκτης μαντεύει `r`. Οι νέες κατηγορίες είναι :

`-e--` (λέξεις χωρίς `r`: `pend`)

`-e-r` (λέξεις όπου το `r` εμφανίζεται μόνο στην 4η θέση: `dear, pear`)

`-er-` (λέξεις όπου το `r` εμφανίζεται μόνο στην 3η θέση: `perk`)

`re-r` (λέξεις όπου το `r` εμφανίζεται στην 1η και στην 4η θέση: `rear`)

Αυτή τη φορά, η πιο πολυπληθής κατηγορία είναι η `-e-r`, και οι πιθανές λέξεις είναι `dear` και `pear`.

Ας υποθέσουμε ότι στον τρίτο γύρο ο παίκτης μαντεύει `f`. Το `f` δεν υπάρχει σε καμία λέξη, οπότε θα βρεθεί μόνο μια κατηγορία:

`-e-r` (λέξεις χωρίς `f`: `dear, pear`)

Το παιχνίδι θα συνεχίσει με τον ίδιο τρόπο. Αν τελειώσουν οι γύροι χωρίς να έχει βρει τη λέξη ο παίκτης, και η τελευταία κατηγορία έχει περισσότερες από μία λέξεις, τότε ο υπολογιστής διαλέγει μία από αυτές στην τύχη και ισχυρίζεται πως αυτή ήταν η λέξη που είχε μαντέψει από την αρχή.

Ακολούθως, ρωτά το χρήστη αν θέλει να παίξει κι άλλο παιχνίδι. Αν όχι, το πρόγραμμα τερματίζει. Αν ναι, ~~ρωτά το μήκος της νέας λέξης και το πλήθος των γύρων, και~~ η διαδικασία επαναλαμβάνεται. [διορθώθηκε 11/11/12]

Υλοποίηση

Εκκίνηση παιχνιδιού

Για την εκκίνηση του παιχνιδιού χρειαζόμαστε τρεις πληροφορίες: το μήκος της λέξης που θα πρέπει να μαντέψουμε, πόσους γύρους θα έχουμε για να μαντέψουμε τη λέξη, και το όνομα του αρχείου-λεξικού. Το αρχείο-λεξικό περιέχει τη συλλογή λέξεων μία εκ των οποίων θα πρέπει να μαντέψουμε.

Και οι τρεις πληροφορίες δίνονται από τη γραμμή εντολής ως ορίσματα του προγράμματος, με τη σειρά που περιγράφεται στην προηγούμενη παράγραφο.

Γράψτε μια συνάρτηση σκοπός της οποίας είναι να επεξεργαστεί τα ορίσματα του προγράμματος, να επιβεβαιώσει ότι έχουν δοθεί έγκυρες τιμές και να επιστρέψει (μέσω της λίστας παραμέτρων της συνάρτησης) το μέγεθος της λέξης και τον αριθμό των γύρων. Η συνάρτηση επιστρέφει ένα κωδικό λάθους ανάλογα με το αν η επεξεργασία των ορισμάτων ήταν επιτυχής ή όχι.

Για την ακρίβεια, η συνάρτηση αυτή :

- Ελέγχει αν έχει δοθεί ο σωστός αριθμός ορισμάτων προγράμματος. Αν έχουν δοθεί λιγότερα από όσα πρέπει, εκτυπώνει το μήνυμα `"Insufficient arguments."`, ένα χαρακτήρα αλλαγής γραμμής, οδηγίες εκτέλεσης του προγράμματος (λεπτομέρειες πιο κάτω) κι επιστρέφει 0. Αν έχουν δοθεί περισσότερα από όσα πρέπει, εκτυπώνει το μήνυμα `"Too many arguments."`, ένα χαρακτήρα αλλαγής γραμμής, οδηγίες εκτέλεσης του προγράμματος κι επιστρέφει 0.

- Ελέγχει αν το μήκος λέξης που δόθηκε είναι έγκυρο. Οι λέξεις στο λεξικό που θα σας δώσουμε έχουν μήκος από 1 έως και 29, εκτός από 26 και 27. Αν το μήκος που δόθηκε δεν είναι έγκυρο, εκτυπώνεται το μήνυμα `"There are no words of length L in the dictionary."` , όπου L το μήκος που δόθηκε, ένας χαρακτήρας αλλαγής γραμμής και το μήνυμα `"Specify a different length: "`. Ακολούθως, το πρόγραμμα διαβάζει μια νέα τιμή για το μήκος, και το παρόν βήμα επαναλαμβάνεται μέχρις ότου να δοθεί έγκυρη τιμή.

- Ελέγχει αν ο αριθμός γύρων που δόθηκε είναι έγκυρος (τουλάχιστον 1). Αν το πλήθος που δόθηκε δεν είναι έγκυρο, εκτυπώνεται το μήνυμα `"You may not have fewer than one turns."` , ένας χαρακτήρας αλλαγής γραμμής και το μήνυμα `"Specify a different number of turns: "`. Ακολούθως, το πρόγραμμα διαβάζει μια νέα τιμή για τον αριθμό γύρων, και το παρόν βήμα επαναλαμβάνεται μέχρις ότου να δοθεί έγκυρη τιμή.

- Η συνάρτηση επιστρέφει 1 εφόσον έχουν δοθεί έγκυρες τιμές για το μήκος και τους γύρους.

Οι οδηγίες εκτέλεσης του προγράμματος, οι οποίες εκτυπώνονται σε περίπτωση που έχει δοθεί λάθος αριθμός ορισμάτων είναι:

```
./hangman LENGTH TURNS DICTIONARY
     LENGTH: requested length of mystery word. Must be >1, <30, !=26, !=27
     TURNS: requested number of turns. Must be >0
     DICTIONARY: name of dictionary file
```

Οι τρεις τελευταίες γραμμές ξεκινούν ένα tab πιο δεξιά από την πρώτη, και μετά την τελευταία γραμμή ακολουθεί χαρακτήρας αλλαγής γραμμής.

Ολοκληρώστε αυτό το στάδιο και βεβαιωθείτε ότι λειτουργεί σωστά πριν προχωρήσετε.

Δομές δεδομένων

Θα χρειαστείτε δύο δομές δεδομένων για το πρόγραμμα:

- **Μια διασυνδεδεμένη λίστα** στην οποία αποθηκεύονται οι λέξεις του λεξικού.

    Σε ένα κόμβο αυτής της λίστας αποθηκεύεται μια λέξη

- **Μια διασυνδεδεμένη λίστα** στην οποία αποθηκεύονται οι κατηγορίες (κλάσεις ισοδυναμίας) λέξεων.

    Σε ένα κόμβο αυτής της λίστας αποθηκεύονται:

    - Ο αντιπρόσωπος της κατηγορίας

    - Το πλήθος των λέξεων που ανήκουν σε αυτή την κατηγορία.

    - Ένας δυναμικά δεσμευμένος πίνακας από δείκτες προς τις λέξεις αυτής της κατηγορίας (όπως αυτές είναι αποθηκευμένες στη λίστα-λεξικό.

Το σχήμα 1 δείχνει τη μορφή των δομών του προγράμματος αμέσως μετά το δεύτερο γύρο.

Γράψτε μια συνάρτηση η οποία παίρνει ως παραμέτρους το επιθυμητό μήκος λέξης και το όνομα του αρχείου-λεξικού, κατασκευάζει τη λίστα-λεξικό κι επιστρέφει δείκτη προς την κεφαλή της. Σημειώστε πως δεν είναι απαραίτητο (αντιθέτως είναι αρκετά σπάταλο) να αποθηκεύσετε όλες τις λέξεις στη λίστα. Αρκεί να αποθηκεύσετε όσες έχουν το επιθυμητό μήκος.

Γράψτε μια συνάρτηση η οποία παίρνει ως παράμετρο τη λίστα-λεξικό (και ότι άλλο μπορεί να χρειάζεται) και αρχικοποιεί τη λίστα κατηγοριών. Σε πρώτο στάδιο, η λίστα αυτή θα περιέχει μόνο μία κατηγορία (δηλαδή μόνο ένα κόμβο). Ο αντιπρόσωπος θα είναι ένα string από παύλες, και σε αυτή την κατηγορία ανήκουν όλες οι λέξεις του λεξικού με το συγκεκριμένο μήκος. Η συνάρτηση επιστρέφει δείκτη προς την κεφαλή της λίστας που κατασκευάστηκε.

Ολοκληρώστε αυτό το στάδιο και βεβαιωθείτε ότι λειτουργεί σωστά πριν προχωρήσετε.

Βασική λειτουργία παιχνιδιού

Σε κάθε γύρο, το παιχνίδι προχωρά ως εξής:

Εκτυπώνει το μήνυμα `"Remaining guesses: X"` ακολουθούμενο από χαρακτήρα αλλαγής γραμμής, όπου X το πλήθος των γύρων που απομένουν.

Εκτυπώνει το μήνυμα `"Used letters: "` και μετά ένα ένα τα γράμματα (μικρά) που έχουν ήδη χρησιμοποιηθεί από το χρήστη, ταξινομημένα σε αύξουσα σειρά, με ένα κενό ανάμεσα σε διαδοχικά γράμματα, και ένα χαρακτήρα αλλαγής γραμμής στο τέλος.

Εκτυπώνει το μήνυμα `"Unused letters: "` και μετά ένα ένα τα γράμματα (μικρά) που δεν έχουν ακόμη χρησιμοποιηθεί από το χρήστη, ταξινομημένα σε αύξουσα σειρά, με ένα κενό ανάμεσα σε διαδοχικά γράμματα, και ένα χαρακτήρα αλλαγής γραμμής στο τέλος.

Εκτυπώνει το μήνυμα `"Word: "` και μετά τη λέξη της κρεμάλας με χαρακτήρα underscore ('_') στις θέσεις που δεν έχει βρεθεί γράμμα, και ένα χαρακτήρα αλλαγής γραμμής στο τέλος. Σε όποια θέση έχει ήδη βρεθεί το γράμμα, αυτό εμφανίζεται ως μικρό (όχι κεφαλαίο).

Εκτυπώνει το μήνυμα `"Guess a letter: "`

Διαβάζει από το πληκτρολόγιο ένα χαρακτήρα. Αν αυτός δεν είναι γράμμα, τότε το παραπάνω μήνυμα επαναλαμβάνεται. Το πρόγραμμα πρέπει να μπορεί να δεχτεί είτε μικρά είτε κεφαλαία. Αν το γράμμα έχει ήδη επιλεγεί στο παρελθόν, τότε εκτυπώνεται το μήνυμα `"You have already guessed this letter"` και επαναλαμβάνονται όλα τα παραπάνω βήματα (από το Remaining guesses και μετά). Το πλήθος γύρων που απομένουν δεν πρέπει να μειωθεί σε αυτή την περίπτωση.

Αφού δοθεί έγκυρο γράμμα, το πρόγραμμα **καλεί μια συνάρτηση** σκοπός της οποίας είναι να κατασκευάσει τις νέες κατηγορίες (κλάσεις ισοδυναμίας) όπως αυτές διαμορφώνονται από την επιλογή γράμματος, να βρει την πολυπληθέστερη κατηγορία, και να επιστρέψει δείκτη προς αυτή (ουσιαστικά δείκτη σε κόμβο της λίστας κατηγοριών). Αν υπάρχουν πολλαπλές κατηγορίες με μέγιστο μέγεθος, μπορείτε να επιλέξετε όποια επιθυμείτε.

Για να μπορούμε να κάνουμε επαλήθευση, μετά την κλήση αυτής της συνάρτησης πρέπει να εκτυπώνεται το μήνυμα `"Category size: X"` όπου X το μέγεθος της κατηγορίας (πλήθος λέξεων που περιέχει).

Ολοκληρώστε αυτό το στάδιο και βεβαιωθείτε ότι λειτουργεί σωστά πριν προχωρήσετε.

Έλεγχος νικητή

Στο τέλος κάθε γύρου, πρέπει να γίνεται έλεγχος για το αν υπάρχει νικητής. Το παιχνίδι τελειώνει με ήττα για τον παίκτη αν τελειώσουν οι διαθέσιμοι γύροι χωρίς να έχουν βρεθεί όλα τα γράμματα και με νίκη αν έχουν βρεθεί όλα τα γράμματα της λέξης πριν τελειώσουν οι γύροι.

Αν έχει κερδίσει ο παίκτης, **το πρόγραμμα εκτυπώνει** το μήνυμα `"YOU WON! The word is X"` όπου X η λέξη που βρέθηκε, κι ένα χαρακτήρα αλλαγής γραμμής.

Αν έχει χάσει ο παίκτης, **το πρόγραμμα εκτυπώνει** το μήνυμα `"YOU LOST! The word was X"` όπου X μια λέξη η οποία έχει επιλεγεί **τυχαία** από την τρέχουσα κατηγορία, κι ένα χαρακτήρα αλλαγής γραμμής

Σε κάθε περίπτωση τερματισμού του παιχνιδιού δίνεται η ευκαιρία στον παίκτη να ξαναπαίξει. **Το πρόγραμμα εκτυπώνει** ένα χαρακτήρα αλλαγής γραμμής και το μήνυμα `"Play again? (y/n): "` και **διαβάζει από το πληκτρολόγιο** ένα χαρακτήρα.

Αν αυτός είναι οτιδήποτε εκτός y/Y ή n/N τότε το πρόγραμμα εκτυπώνει το μήνυμα `"Please enter y for yes or n for no.",` ένα χαρακτήρα αλλαγής γραμμής, το μήνυμα `"Play again? (y/n): "` και ξαναδιαβάζει την απάντηση του παίκτη. Το βήμα αυτό επαναλαμβάνεται μέχρις ότου να δοθεί έγκυρη απάντηση.

Αν η απάντηση είναι y ή Y τότε ξεκινά καινούργιο παιχνίδι. Το μήκος της λέξης και το αρχικό λεξικό παραμένουν ίδια. Αν η απάντηση είναι n ή N, το πρόγραμμα τερματίζει.

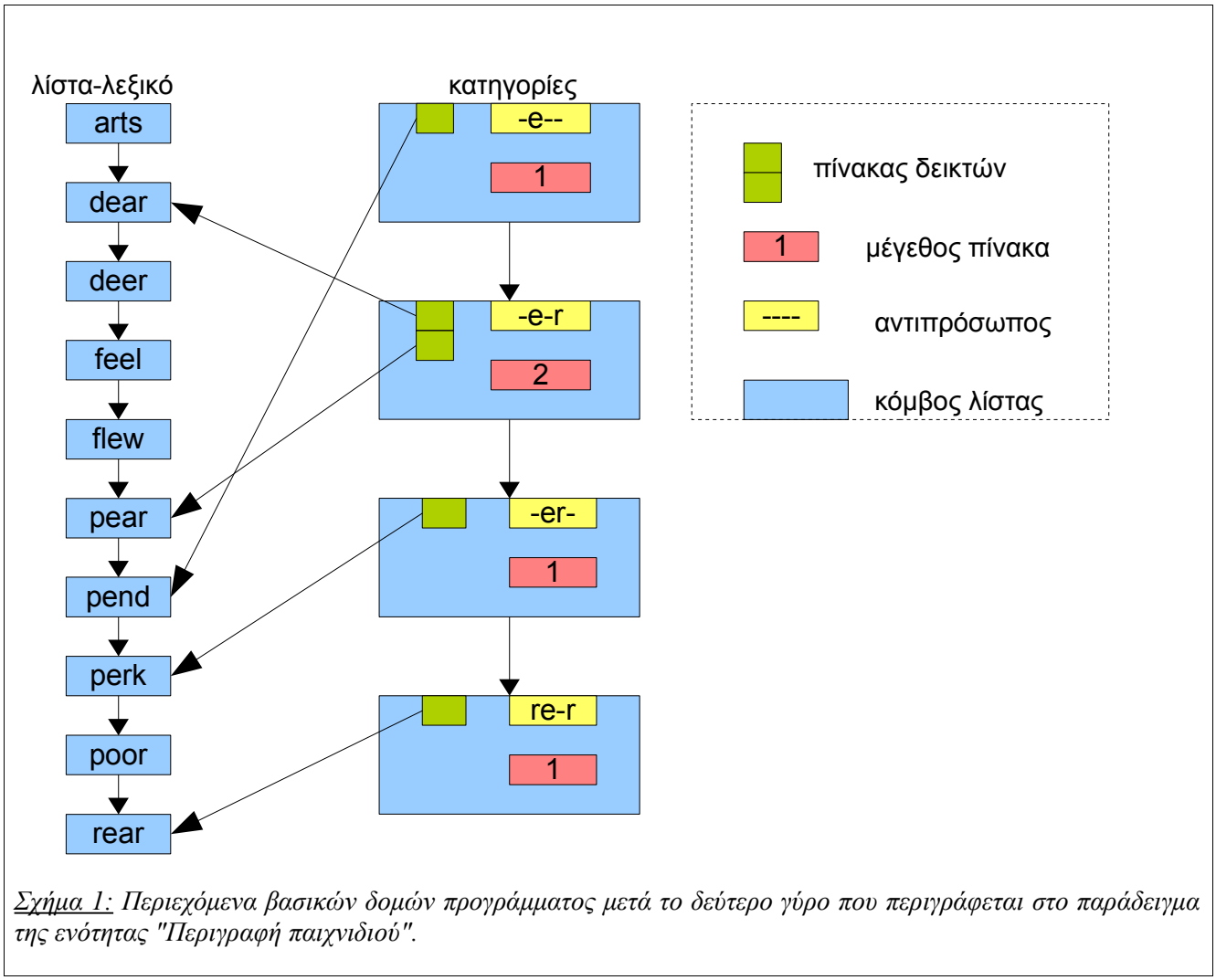
Γενικές απαιτήσεις:

- Απαγορεύεται αυστηρά η χρήση καθολικών μεταβλητών
- Χρησιμοποιήστε συναρτήσεις για καλά ορισμένες λειτουργίες του προγράμματος. Θα δοθεί ιδιαίτερη σημασία στο πώς έχετε οργανώσει τον κώδικά σας.
- Ο δυναμικά δεσμευμένος πίνακας δεικτών που βρίσκεται σε κάθε κόμβο της λίστας κατηγοριών πρέπει να έχει *ακριβώς* τόσο μέγεθος όσοι είναι οι δείκτες που αποθηκεύονται σε αυτόν.
- Πρέπει να χρησιμοποιείτε συναρτήσεις από τη string.h οπουδήποτε κάνετε διαχείριση συμβολοσειρών.

- Φροντίστε να αποδεσμεύετε πάντα σωστά τη δυναμικά δεσμευμένη μνήμη.
- Το πρόγραμμά σας πρέπει να έχει αποτελεσματικά σχόλια, καλά ονόματα μεταβλητών, σωστή στοίχιση και γενική μορφή κώδικα.
- Το πρόγραμμά σας πρέπει να κάνει compile χωρίς λάθη ή προειδοποιήσεις και να τρέχει σε linux.
- Οτιδήποτε εκτυπώνεται στην οθόνη πρέπει να είναι ακριβώς όπως περιγράφεται στην εκφώνηση.
- Οι λίστες μπορούν να είναι ότι τύπου επιθυμείτε (απλά ή διπλά διασυνδεδεμένες, κυκλικές ή μη, με τερματικό ή μη), αρκεί οι κόμβοι να περιέχουν τις πληροφορίες που απαιτεί η εκφώνηση. Επίσης, μπορείτε να βελτιώσετε ελαφρώς τη λίστα κατηγοριών: Είναι απαραίτητο να κρατάτε μια κατηγορία στη λίστα αν ξέρετε ότι έχει ήδη βρεθεί μία με περισσότερες λέξεις?

Βοήθεια

Στο αρχείο `getWord.c` σας δίνουμε την υλοποίηση μιας συνάρτησης με όνομα `getWord` η οποία κάθε φορά που καλείται διαβάζει την επόμενη λέξη από ένα αρχείο (το όνομα του οποίου δίνεται ως παράμετρος στη συνάρτηση). Η `getWord` επιστρέφει τη λέξη που διάβασε, ως δείκτη σε δυναμικά δεσμευμένη συμβολοσειρά, μεγέθους WORDLEN. Επιστρέφει NULL αν δεν υπάρχουν άλλες λέξεις στο αρχείο. Χρησιμοποιήστε τη ως έχει, χωρίς αλλαγές.



Σχήμα 1: Περιεχόμενα βασικών δομών προγράμματος μετά το δεύτερο γύρο που περιγράφεται στο παράδειγμα της ενότητας "Περιγραφή παιχνιδιού".