ΣΕΤ ΑΣΚΗΣΕΩΝ 1

ΕΡΓΑΣΤΗΡΙΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι, ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2012-2013

Προθεσμία: 10/11/12, 22:00

Περιεχόμενα

Πριν ξεκινήσετε (ΔΙΑΒΑΣΤΕ ΑΥΤΗ ΤΗΝ ΕΝΟΤΗΤΑ!!)

Διαβάστε ΟΛΗ την εκφώνηση προσεκτικά και "σχεδιάστε" το πρόγραμμά σας στο χαρτί.

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

Μη διστάζετε να ζητήσετε βοήθεια! Μπορείτε να χρησιμοποιήσετε το forum προγραμματισμού (http://inf-server.inf.uth.gr/courses/coding/) και φυσικά email.

Μην προσπαθήσετε να γράψετε το πρόγραμμα μια κι έξω. Για κάθε στάδιο που ολοκληρώνετε, βεβαιωθείτε ότι κάνει compile και λειτουργεί σωστά και μετά να προχωράτε στο επόμενο στάδιο. Κάθε φορά που τελειώνετε ένα στάδιο, να αποθηκεύετε ένα αντίγραφο του προγράμματος όπως είναι μέχρι εκείνο το σημείο. Έτσι, αν σε κάποιο επόμενο στάδιο καταστραφεί το αρχείο σας, θα έχετε μια παλιότερη έκδοση από την οποία μπορείτε να συνεχίσετε. Τις επιμέρους "εκδόσεις" να τις ονομάζετε με διαφορετικά ονόματα (πχ. `decoderstadio1.c, decoderstadio2.c`)

Σε κάθε στάδιο σας δίνουμε μια εκτίμηση της δυσκολίας του (διαβάθμιση απο 1 έως 4).

Η εργασία αυτή μπορεί να γίνει σε ομάδες μέχρι 2 ατόμων. Δεν είναι απαραίτητο να συνεργαστείτε με το ίδιο άτομο που κάνετε τα εβδομαδιαία εργαστήρια. 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: Αποκρυπτογράφηση κειμένου

Εισαγωγή

Ένας εύκολος αλγόριθμος κρυπτογράφησης αντικαθιστά κάθε εκτυπώσιμο χαρακτήρα με αυτόν που έπεται κατά κάποιους χαρακτήρες στον πίνακα ASCII. Οι εκτυπώσιμοι χαρακτήρες ASCII ξεκινούν από το `' '` (κωδικός 32) και φτάνουν ως το `'~'` (κωδικός 126). Σε περίπτωση που περάσουμε τον τελευταίο εκτυπώσιμο χαρακτήρα, συνεχίζουμε κυκλικά από τον πρώτο.

Το κλειδί της κρυπτογράφησης είναι η απόσταση του χαρακτήρα που αντικαθίσταται από αυτόν που τον αντικαθιστά. Έτσι, για παράδειγμα, αν το κλειδί είναι 5, η λέξη `wizard` κρυπτογραφείται ως `|n fwi`. Τυχόν μη εκτυπώσιμοι χαρακτήρες (π.χ. `'\n'`) παραμένουν ως έχουν.

Η βασική παρατήρηση για την αποκρυπτογράφηση είναι ότι ο πιο συχνός χαρακτήρας σε ένα κείμενο είναι κατά κανόνα το `' '` (κενό). Άρα, αν κανείς αναλύσει τη συχνότητα εμφάνισης των χαρακτήρων στο κρυπτογραφημένο κείμενο μπορεί βάσιμα να υποθέσει ότι ο χαρακτήρας με τη μεγαλύτερη συχνότητα εμφάνισης αντιστοιχεί στο αρχικό κείμενο στο κενό. Υπολογίζοντας την απόσταση αυτού του χαρακτήρα από το `' '` προκύπτει το κλειδί. Το κλειδί χρησιμοποιείται για την αποκρυπτογράφηση όλου του κειμένου, μετακινώντας κάθε χαρακτήρα πίσω (κυκλικά) τόσες θέσεις όσο η τιμή του κλειδιού.

Το πρόγραμμά σας θα πρέπει να διαβάζει από την είσοδο (με ανακατεύθυνση από αρχείο κειμένου) το κρυπτογραφημένο κείμενο, το οποίο μπορεί να έχει μήκος έως και 1024 χαρακτήρες. Αν το κείμενο περιέχει παραπάνω χαρακτήρες, αυτοί αγνοούνται. Το πρόγραμμα αναλύει τη συχνότητα εμφάνισης των χαρακτήρων στο κρυπτογραφημένο κείμενο (μετράει δηλαδή πόσες φορές εμφανίζεται κάθε εκτυπώσιμος χαρακτήρας), υπολογίζει και εκτυπώνει το κλειδί και κατόπιν αποκρυπτογραφεί το κείμενο και το εκτυπώνει στην έξοδο (από την οποία οδηγείται σε αρχείο με ανακατεύθυνση).

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

Αποθηκεύστε το πρόγραμμα σε αρχείο με το όνομα `decoder.c` . Χρησιμοποιήστε σαν είσοδο (με ανακατεύθυνση) το κωδικοποιημένο αρχείο κειμένου `encoded.txt`.

Στάδιο 1: Εισαγωγή και αρχική επεξεργασία δεδομένων (éé)

Διαβάστε χαρακτήρα-προς-χαρακτήρα από το πληκτρολόγιο ένα κείμενο έως ότου συναντήσετε τον ειδικό χαρακτήρα EOF ή έχετε διαβάσει το ανώτατο όριο χαρακτήρων που επιτρέπεται να περιέχει το κείμενο (όποιο από τα δύο συμβεί πρώτο).

Το κείμενο πρέπει να αποθηκευτεί σε κατάλληλο πίνακα.

Ταυτόχρονα, για κάθε εκτυπώσιμο χαρακτήρα μετρήστε το πλήθος εμφανίσεων αυτού στο κείμενο και αποθηκεύστε αυτή την πληροφορία σε πίνακα τόσων θέσεων όσοι και οι εκτυπώσιμοι χαρακτήρες ASCII.

Προσωρινός κώδικας: Εκτυπώστε τα περιεχόμενα και των δύο πινάκων και επιβεβαιώστε ότι αυτό το στάδιο λειτουργεί σωστά.

Το πρόγραμμά σας πρέπει να είναι γραμμένο με τέτοιο τρόπο ώστε αν κάποια στιγμή αυξηθεί το μέγιστο επιτρεπτό μέγεθος κειμένου, να μπορούν να γίνουν εύκολα και γρήγορα οι κατάλληλες αλλαγές στο πρόγραμμα.

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

Στάδιο 2: Εντοπισμός κενού ' '(★)

Αφαιρέστε τον προσωρινό κώδικα που γράψατε στο στάδιο 1.

Εντοπίστε το χαρακτήρα με τη μεγαλύτερη συχνότητα (πλήθος εμφανίσεων) διατρέχοντας τον πίνακα στον οποίο είναι αποθηκευμένο το πλήθος εμφανίσεων για κάθε εκτυπώσιμο χαρακτήρα.

Προσωρινός κώδικας: Εκτυπώστε το χαρακτήρα (ή τη θέση) που βρήκατε και επιβεβαιώστε ότι είναι σωστός.

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

Στάδιο 3: Αποκωδικοποίηση κειμένου (★★★★)

Αφαιρέστε τον προσωρινό κώδικα που γράψατε στο στάδιο 2.

Χρησιμοποιήστε τη μέθοδο που περιγράφεται στην εισαγωγή για να υπολογίσετε το κλειδί, κι εκτυπώστε το στην οθόνη σε μήνυμα της μορφής :

`KEY: k`

όπου `k` το κλειδί που υπολογίσατε. Εκτυπώστε δύο χαρακτήρες αλλαγής γραμμής μετά το μήνυμα.

Έχοντας υπολογίσει το κλειδί, χρησιμοποιήστε τη μέθοδο που περιγράφεται στην εισαγωγή για να αποκωδικοποιήσετε το κείμενο. Εκτυπώστε το αποκωδικοποιημένο κείμενο στην οθόνη σε μήνυμα της μορφής:

`TEXT: t`

όπου t το αποκωδικοποιημένο κείμενο. Εκτυπώστε ένα χαρακτήρα αλλαγής γραμμής μετά το μήνυμα.

Βοήθεια:

Χρησιμοποιήστε τον τελεστή % (υπόλοιπο ακέραιας διαίρεσης – modulo) για να κάνετε "κυκλική" διάτρεξη των εκτυπώσιμων χαρακτήρων, αλλά λάβετε υπόψη ότι δεν ξεκινάμε από το 0, αλλά από το χαρακτήρα με ASCII 32. Θα πρέπει να μετατρέψετε το σύνολο χαρακτήρων με ASCII 32-126 σε σύνολο που ξεκινά από το 0 πριν μπορέσετε να κάνετε κυκλική διάτρεξη, και μετά να το ξαναμετατρέψετε στη σωστή κλίμακα. Δώστε ιδιαίτερη προσοχή ώστε να μην προκύψει ποτέ αρνητικός αριθμός κατά τη διάρκεια των πράξεών σας.

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

Στάδιο 4: Τελικός έλεγχος ορθότητας

Το πρόγραμμά σας πρέπει να λειτουργεί σωστά και να εκτυπώνει όλα τα μηνύματα και αποτελέσματα με τον τρόπο που σας περιγράφουμε. Για να μπορέσετε να ελέγξετε την ορθότητα θα σας δώσουμε ενδεικτικά αρχεία εισόδου και εξόδου. Υπάρχει ένας εύκολος τρόπος να συγκρίνετε τα δικά σας αποτελέσματα με τα δικά μας:

Ας υποθέσουμε ότι το εκτελέσιμο πρόγραμμά σας λέγεται `decoder`, το ενδεικτικό αρχείο εισόδου λέγεται `encoded.txt` και το αντίστοιχο αρχείο εξόδου που σας έχουμε δώσει λέγεται `stdout1.txt`

Η εντολή:

```
./decoder < encoded.txt > myout1.txt
```

εκτελεί το πρόγραμμά σας με την ενδεικτική είσοδο `input1.txt` και αποθηκεύει τα αποτελέσματα στο αρχείο εξόδου `myout1.txt`

Η εντολή

```
diff   -b   myout1.txt   stdout1.txt
```

συγκρίνει το δικό σας αρχείο εξόδου με το δικό μας. Αν υπάρχουν διαφορές, τις εμφανίζει (γραμμή-γραμμή). Αν δεν υπάρχουν διαφορές, δεν κάνει τίποτα.

Για περισσότερες πληροφορίες δείτε το σχετικό παράρτημα στο τέλος του φυλλαδίου.

Πρέπει το πρόγραμμα που θα μας παραδώσετε να παράγει έξοδο που δεν έχει διαφορές από τη δική μας.

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

Αρχείο προς παράδοση: `decoder.c`

Άσκηση 2: Λεκτική ανάλυση κειμένου

Εισαγωγή

Σας ζητείται να φτιάξετε πρόγραμμα το οποίο θα δέχεται από την είσοδο κείμενο έως και 1024 χαρακτήρων. Εάν το κείμενο αποτελείται από παραπάνω χαρακτήρες, αυτοί αγνοούνται. Το πρόγραμμα θα πρέπει να αναγνωρίζει τις διαφορετικές λέξεις του κειμένου και να τις εκτυπώνει. Υποθέστε ότι το κείμενο μπορεί να περιέχει έως 512 λέξεις και ότι κάθε λέξη έχει το πολύ 28 γράμματα.

Αποθηκεύστε το πρόγραμμα σε αρχείο με το όνομα `scanner.c` . Χρησιμοποιήστε σαν είσοδο (με ανακατεύθυνση) το αρχείο κειμένου `words.txt` .

Στάδιο 1: Εισαγωγή και αρχική επεξεργασία δεδομένων (★)

Διαβάστε χαρακτήρα-προς-χαρακτήρα από το πληκτρολόγιο ένα κείμενο έως ότου συναντήσετε τον ειδικό χαρακτήρα EOF ή έχετε διαβάσει το ανώτατο όριο χαρακτήρων που επιτρέπεται να περιέχει το κείμενο (όποιο από τα δύο συμβεί πρώτο). Μετρήστε τους χαρακτήρες που διαβάσατε. Η πληροφορία αυτή θα σας φανεί χρήσιμη. Το κείμενο πρέπει να αποθηκευτεί σε κατάλληλο πίνακα.

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

Στάδιο 2: Ανίχνευση λέξεων (★★★)

Διατρέξτε τον πίνακα με το κείμενο για να ανιχνεύσετε τις λέξεις. Μια λέξη αποτελείται από συνεχόμενους χαρακτήρες ‘a’-‘z’ και ‘A’-‘Z’. Αν παρεμβληθεί οτιδήποτε άλλο, ξεκινά μια νέα λέξη.

Ψάξτε για την αρχή μιας λέξης. Όσο βρίσκετε χαρακτήρες που δεν ανήκουν σε λέξη (δεν είναι δηλαδή ‘a’-‘z’ και ‘A’-‘Z’) μπορείτε να τους αντικαθιστάτε με ‘\0’. Με αυτό τον τρόπο δημιουργείτε (εύκολα) strings μέσα στον πίνακα που περιέχει το κείμενο. Κάθε string περιέχει μια λέξη.

Αφού βρείτε την αρχή μιας λέξης, πρέπει να βρείτε το τέλος της. Έχοντας κρατήσει την αρχή της σε

κατάλληλη μεταβλητή, συνεχίσετε να διατρέχετε τον πίνακα-κείμενο έως ότου βρείτε χαρακτήρα που δεν είναι γράμμα. Αυτό θα είναι το τέλος της λέξης σας.

Έχοντας εντοπίσει τη λέξη, αντιγράψτε τη σε ένα διδιάστατο πίνακα χαρακτήρων (ευρετήριο) ο οποίος έχει τόσες γραμμές όσο είναι το μέγιστο δυνατό πλήθος λέξεων στο κείμενο και τόσες στείλες όσο είναι το μέγιστο δυνατό μέγεθος μιας λέξης.

Κάθε νέα λέξη που βρίσκετε πρέπει να τοποθετείται στην επόμενη γραμμή του διδιάστατου πίνακα.

Εκτυπώστε τα περιεχόμενα του πίνακα-ευρετήριο, μια-μία λέξη με χαρακτήρα αλλαγής γραμμής ανάμεσα σε διαδοχικές λέξεις.

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

Στάδιο 3: Τελικός έλεγχος ορθότητας

Το πρόγραμμά σας πρέπει να λειτουργεί σωστά και να εκτυπώνει όλα τα μηνύματα και αποτελέσματα με τον τρόπο που σας περιγράφουμε. Για να μπορέσετε να ελέγξετε την ορθότητα θα σας δώσουμε ενδεικτικά αρχεία εισόδου και εξόδου. Υπάρχει ένας εύκολος τρόπος να συγκρίνετε τα δικά σας αποτελέσματα με τα δικά μας:

Ας υποθέσουμε ότι το εκτελέσιμο πρόγραμμά σας λέγεται `scanner`, το ενδεικτικό αρχείο εισόδου λέγεται `input1.txt` και το αντίστοιχο αρχείο εξόδου που σας έχουμε δώσει λέγεται `stdout1.txt`

Η εντολή:

```
./scanner   <    input1.txt    >    myout1.txt
```

εκτελεί το πρόγραμμά σας με την ενδεικτική είσοδο `input1.txt` και αποθηκεύει τα αποτελέσματα στο αρχείο εξόδου `myout1.txt`

Η εντολή

```
diff   -b   myout1.txt   stdout1.txt
```

συγκρίνει το δικό σας αρχείο εξόδου με το δικό μας. Αν υπάρχουν διαφορές, τις εμφανίζει (γραμμή-γραμμή). Αν δεν υπάρχουν διαφορές, δεν κάνει τίποτα.

Για περισσότερες πληροφορίες δείτε το σχετικό παράρτημα στο τέλος του φυλλαδίου.

Πρέπει το πρόγραμμα που θα μας παραδώσετε να παράγει έξοδο που δεν έχει διαφορές από τη δική μας.

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

Αρχείο προς παράδοση: `scanner.c`

Πώς να παραδώσετε τη δουλειά σας

(Ακολουθείστε τις οδηγίες ακριβώς αλλιώς μπορεί να μη δούμε τα αρχεία σας)

Κατασκευάστε ένα φάκελο με όνομα `epwnumo1_AM1_epwnumo2_AM2` και αντιγράψτε μέσα σε αυτόν το `decoder.c` και το `scanner.c`

Πηγαίνετε στο φάκελο μέσα στον οποίο βρίσκεται το `epwnumo1_AM1_epwnumo2_AM2` που κατασκευάσατε και γράψτε την παρακάτω εντολή:

```
tar czf epwnumo1_AM1_epwnumo2_AM2.tgz epwnumo1_AM1_epwnumo2_AM2
```

Στείλτε email:

- στη διεύθυνση **ce120lab@gmail.com**
- αντίγραφο (**CC**) στον άλλο μέλος της ομάδας σας
- θέμα (subject) **CE120 hw1**
- και επικολλημένο αρχείο το `epwnumo1_AM1_epwnumo2_AM2.tgz`

Το πρόγραμμά σας θα βαθμολογηθεί ως προς:

- ορθότητα υπολογισμών
- σωστή επιλογή τύπων
- σωστή επιλογή και χρήση δομών ελέγχου κι επανάληψης
- σωστή χρήση πινάκων
- αναγνωσιμότητα (ονόματα, στοίχιση, σχολιασμός, μορφή κώδικα)
- φορμαρισμένη είσοδος/έξοδος και συμμόρφωση με τις προδιαγραφές.

Παράρτημα Α: Ανακατεύθυνση στο Linux

Μέχρι στιγμής, έχετε δεί πως να γράφετε ένα πρόγραμμα που διαβάζει από το πληκτρολόγιο και γράφει στην οθόνη. Αυτό που συμβαίνει στην πραγματικότητα είναι ότι το πρόγραμμα διαβάζει από το λεγόμενο standard input (stdin, προκαθορισμένη είσοδος) και γράφει στο λεγόμενο standard output (stdout, προκαθορισμένη έξοδος). Τυπικά, το standard input είναι συνδεδεμένο με το πληκτρολόγιο και το standard output με την οθόνη.

Στην πρώτη άσκηση του σημερινού εργαστηρίου θέλουμε να διαβάσουμε από το πληκτρολόγιο μέχρι 1024 χαρακτήρες. Αυτό είναι αρκετά μεγάλο νούμερο - θα ήταν πιο πρακτικό αν είχαμε αυτούς τους χαρακτήρες σε ένα αρχείο και διαβάζαμε από το αρχείο. Το Linux μας επιτρέπει να επιτύχουμε κάτι τέτοιο με την "ανακατεύθυνση".

Όταν το πρόγραμμα θέλει να διαβάσει είσοδο, μπορούμε να την ανακατευθύνουμε ώστε να γίνεται από αρχείο, αντί από το πληκτρολόγιο. Ομοίως, όταν το πρόγραμμα παράγει έξοδο, μπορούμε να την ανακατευθύνουμε από την οθόνη σε ένα αρχείο.

Για να ανακατευθύνουμε την είσοδο χρησιμοποιούμε το ειδικό σύμβολο <

Παράδειγμα: `./myprogram < input`

Το πρόγραμμα `myprogram` παίρνει την είσοδό του από το αρχείο input αντί για το πληκτρολόγιο

Για να ανακατευθύνουμε την έξοδο χρησιμοποιούμε το ειδικό σύμβολο >

Παράδειγμα: `./myprogram > output`

Το πρόγραμμα `myprogram` γράφει την έξοδό του στο αρχείο output αντί για την οθόνη

Τέλος, μπορούμε να ανακατευθύνουμε και την είσοδο και την έξοδο ταυτόχρονα:

Παράδειγμα: `./myprogram < input > output`

Το πρόγραμμα `myprogram` παίρνει την είσοδό του από το αρχείο input και γράφει την έξοδό του στο αρχείο output.

Γράψτε σε ένα αρχείο με όνομα `data.c` το παρακάτω πρόγραμμα:

```
#include<stdio.h>

int main(int argc, char *argv[]) {
      int i;
      for (i=0; i<120; i++)  {
            printf("%d\n", i);
      }
      return 0;
}
```

Κάντε compile κι εκτελέστε το. Θα δείτε να τυπώνει στην οθόνη 120 αριθμούς.

Τώρα, εκτελέστε το με ανακατεύθυνση εξόδου προς ένα αρχείο με όνομα `numbers.txt`:

`./data > numbers.txt`

Δε θα εμφανιστεί τίποτα στην οθόνη, αλλά αν ανοίξετε το `numbers.txt` θα δείτε μέσα τους 120 ακεραίους.

Παράρτημα Β: Σύγκριση αρχείων

Η εντολή `diff` του Linux χρησιμοποιείται για να ελέγχουμε αν δυο αρχεία είναι ίδια. Δοκιμάστε τη! Φτιάξτε ένα αντίγραφο του αρχείου `data.c` από το προηγούμενο παράρτημα και ονομάστε το `data2.c`. Μετά, στη γραμμή εντολής γράψτε:

```
diff  -b  data.c   data2.c
```

Δε θα βγεί τίποτα γιατί τα αρχεία είναι ίδια. Τώρα, κάντε κάποιες αλλαγές στο `data2.c` και ξαναδοκιμάστε. Το `diff` θα παρουσιάσει όλες τις διαφορές ανάμεσα στα δύο προγράμματα (ανά γραμμή)

Μπορείτε να χρησιμοποιείτε ανακατεύθυνση για να εκτυπώνετε την έξοδο ενός προγράμματός σας σε αρχείο, να κάνετε το ίδιο και για την έξοδο του εκάστοτε εκτελέσιμου που σας δίνουμε και μετά να χρησιμοποιήσετε diff για να δείτε αν τα αποτελέσματα είναι ίδια. Με τον ίδιο τρόπο θα ελέγχουμε κι εμείς τα αποτελέσματα των προγραμμάτων σας όποτε είναι δυνατό.