Το παρόν φυλλάδιο απευθύνεται κυρίως σε φοιτητές που προετοιμάζονται για τη γραπτή εξέταση αλλά μπορεί να χρησιμοποιηθεί και από φοιτητές που συμμετέχουν στα εργαστήρια.

Στόχος των ασκήσεων είναι η εξάσκηση στα πιο βασικά σημεία της ύλης του Προγραμματισμού 2. Κάντε τις σταδιακά, μετά από κάθε νέα ενότητα διδασκαλίας.

ΠΡΟΣΟΧΗ: Οι ασκήσεις δεν καλύπτουν όλη την ύλη (άρα ούτε και όλα τα πιθανά ζητούμενα μιας τελικής εξέτασης) αλλά σας δείχνουν ένα καλό τρόπο μελέτης τον οποίο μπορείτε να μεταφέρετε σε όλα τα θέματα του μαθήματος: Δοκιμάζετε σχετικά μικρά παραδείγματα, συχνά βασισμένα στα προγράμματα που έχετε δει στις διαλέξεις, προσπαθώντας να κατανοήσετε γιατί και πώς λειτουργούν και τα μεταβάλλετε με ενδιαφέροντες τρόπους για επιπλέον εξάσκηση.

1. Στη διάλεξη [**Δυναμικοί πίνακες**] περιλαμβάνεται κώδικας για την υλοποίηση ενός phonebook ως δυναμικού πίνακα από εγγραφές. Κάντε τις παρακάτω προσθήκες/αλλαγές στο πρόγραμμα:

α) Αφού μελετήσετε το πρόγραμμα, σβήστε τις έτοιμες συναρτήσεις και ξαναγράψτε τις εσείς. Βεβαιωθείτε ότι το πρόγραμμα συνεχίζει να λειτουργεί σωστά.

β) Αλλάξτε τη συνάρτηση main ώστε σε επανάληψη να εκτυπώνει ένα μενού λειτουργιών (προσθήκη, αφαίρεση, εκτύπωση, έξοδος) μέσα από το οποίο ο χρήστης μπορεί να διαχειρίζεται ελεύθερα τον κατάλογο.

γ) Μεταβάλλετε τη βασική δομή ώστε το phonebook να υλοποιείται ως δυναμικός πίνακας από *δείκτες* σε εγγραφές. Σκεφτείτε πώς πρέπει να αλλάξετε τον τύπο του πεδίου entries.

δ) Ξεκινώντας από την αρχική έκδοση του προγράμματος, μεταβάλλετε το ώστε να μη χρησιμοποιεί το struct phonebook_t αλλά να περνάει μεμονωμένα τα entries και το size ως παραμέτρους. Σκεφτείτε προσεκτικά πώς πρέπει να τα περάσετε ώστε να μπορείτε να τα μεταβάλετε μέσα στην εκάστοτε συνάρτηση με τρόπο που θα τους επιτρέπει να διατηρηθούν μετά την επιστροφή στη main.

2. Στη διάλεξη [**Λίστες**] περιλαμβάνεται κώδικας για την υλοποίηση τριών τύπων λίστας:

- Απλά διασυνδεδεμένη,
- Απλά διασυνδεδεμένη, κυκλική με τερματικό κόμβο (sentinel) και
- Διπλά διασυνδεδεμένη, κυκλική με τερματικό κόμβο.

Μελετήστε τις συναρτήσεις αρχικοποίησης, εύρεσης, εισαγωγής, διαγραφής και καταστροφής που δίνονται στις διαφάνειες και μετά για κάθε ένα τύπο λίστας γράψτε ένα πρόγραμμα στο οποίο υλοποιείτε εκ νέου αυτές τις συναρτήσεις. Για κάθε υλοποίηση, σκεφτείτε γιατί και πώς λειτουργεί στις οριακές περιπτώσεις (π.χ. διαγραφή από την αρχή της λίστας, αναζήτηση στοιχείου που δεν υπάρχει).

α) Για κάθε ένα τύπο λίστας, προσθέστε στο αντίστοιχο πρόγραμμα μια συνάρτηση που διατρέχει τη λίστα από την αρχή ως το τέλος κι εκτυπώνει τα περιεχόμενα της. Ειδικά για την διπλά διασυνδεδεμένη, τροποποιήστε τη συνάρτηση εκτύπωσης ώστε αφού εκτυπώσει τα περιεχόμενα από την αρχή προς το τέλος, να τα εκτυπώσει κι από το τέλος προς την αρχή.

β) Σε κάθε ένα πρόγραμμα γράψτε μια main που κάνει τα παρακάτω (χρησιμοποιώντας τις συναρτήσεις σας):

   i. Κατασκευάζει μία λίστα με 10 στοιχεία

   ii. Αφού την κατασκευάσει, εκτυπώνει τα περιεχόμενα της

   iii. Αναζητά στοιχεία που υπάρχουν ή δεν υπάρχουν στη λίστα.

   iv. Διαγράφει ένα στοιχείο από την αρχή, ένα από τη μέση κι ένα από το τέλος της λίστας. Εκτυπώστε τα περιεχόμενα μετά από κάθε διαγραφή

γ) Σκεφτείτε τι θα αλλάζατε στις συναρτήσεις εισαγωγής, αναζήτησης και διαγραφής αν θέλατε τα στοιχεία της λίστας να διατηρούνται πάντα ταξινομημένα. Κάντε τις αλλαγές κι εξετάστε αν εξακολουθεί να λειτουργεί σωστά το πρόγραμμα που γράψατε στο (β).

3. Όταν θα έχετε άνεση σε **λίστες**, προσπαθήστε τις παρακάτω ασκήσεις σε διάφορους τύπους λιστών:

α) Γράψτε ένα πρόγραμμα που κατασκευάζει μια λίστα ακεραίων και μετά διαβάζει έναν ακέραιο και αφαιρεί από τη λίστα όλες τις εμφανίσεις του. Δοκιμάστε το σε διάφορους τύπους λίστας.

β) Γράψτε ένα πρόγραμμα που κατασκευάζει μια λίστα ακεραίων και μετά τη διατρέχει και για προσθέτει μετά από κάθε κόμβο ένα αντίγραφο αυτού. Για παράδειγμα, αν αρχικά η λίστα περιέχει 5-8-3-1-8, τότε μετά θα περιέχει 5-5-8-8-3-3-1-1-8-8

γ) Γράψτε ένα πρόγραμμα που αρχικά κατασκευάζει δύο λίστες ακεραίων και στη συνέχεια "κολλά" τη δεύτερη στο τέλος της πρώτης. Για παράδειγμα, αν οι δύο αρχικές λίστες περιέχουν 5-8-3-1 και 2-9-6, τότε μετά την επικόλληση η πρώτη λίστα θα περιέχει 5-8-3-1-2-9-6.

δ) Γράψτε ένα πρόγραμμα που αρχικά κατασκευάζει δύο λίστες ακεραίων που διατηρούνται ταξινομημένες, και στη συνέχεια τις διατρέχει (σκεφτείτε πώς!) και κατασκευάζει μια νέα λίστα που αποτελείται από τα στοιχεία των δύο αρχικών, πάλι ταξινομημένα. Για παράδειγμα, αν οι δύο αρχικές λίστες περιέχουν 1-3-7-9 και 2-3-5-8-10-14, τότε η νέα λίστα θα περιέχει 1-2-3-3-5-7-8-9-10-14

ε) Γράψτε ένα πρόγραμμα που αρχικά κατασκευάζει δύο λίστες ακεραίων και στη συνέχεια τις διατρέχει και κατασκευάζει μια νέα λίστα που αποτελείται μόνο από τα στοιχεία που είναι κοινά και στις δύο αρχικές λίστες

4. Ένας **πίνακας κατακερματισμού** μπορεί να υλοποιηθεί ως δυναμικός πίνακας από λίστες. Υλοποιήστε έναν απλό πίνακα κατακερματισμού, με ότι είδος λίστας θέλετε και

δοκιμάστε μερικές εισαγωγές και διαγραφές στοιχείων. Να εκτυπώνετε κάθε φορά τα περιεχόμενα του πίνακα. Χρησιμοποιείστε την απλή συνάρτηση κατακερματισμού που παρουσιάζεται στις διαφάνειες.

5. Γράψτε ένα πρόγραμμα το οποίο δημιουργεί ένα **αρχείο** και γράφει σε αυτό μια σειρά από λέξεις που του δίνει ο χρήστης. Κάθε λέξη πρέπει να μπορεί να αποθηκευτεί σε πίνακα μεγέθους SIZE. Μετά, προσθέστε κώδικα που ψάχνει στο αρχείο να βρει την N-οστή λέξη, τη διαβάζει και την εκτυπώνει στην οθόνη. Χρησιμοποιήστε δικές σας τιμές για τα SIZE, N. Κάντε δοκιμές και για N μεγαλύτερο του πλήθους λέξεων που είναι στο αρχείο.
Σκεφτείτε: Ένας τρόπος επίλυσης είναι γράφοντας SIZE bytes στο αρχείο για κάθε λέξη. Εναλλακτικά, μπορείτε να γράφετε ακριβώς τη λέξη. Πώς επηρεάζει την αναζήτηση της N-οστής λέξης κάθε μία από αυτές τις παραλλαγές;

6. Επαναλάβετε την παραπάνω άσκηση, αλλά αυτή τη φορά γράψτε στο αρχείο ακεραίους (όχι ως κείμενο αλλά στην "ωμή" μορφή τους).

7. Επαναλάβετε τις ασκήσεις 5 και 6, αλλά αυτή τη φορά η λέξη ή ο ακέραιος που βρέθηκε πρέπει να αντιγράφεται στο τέλος του αρχείου.

8. Επαναλάβετε τις ασκήσεις 5 και 6, αλλά αυτή τη φορά η λέξη ή ο ακέραιος που βρέθηκε πρέπει να αντιγράφεται στην αρχή του αρχείου (δηλαδή πριν την πρώτη λέξη).

9. Κατασκευάστε ένα struct με 3 πεδία: Μια συμβολοσειρά (όνομα) μεγέθους SIZE, ένα χαρακτήρα (φύλο), έναν ακέραιο (ηλικία). Φτιάξτε ένα πίνακα από τέτοια struct και γεμίστε τον με ενδεικτικές τιμές. Στη συνέχεια, γράψτε κάθε ένα από αυτά τα struct στο αρχείο. Γράψτε ένα δεύτερο πρόγραμμα που διαβάζει το παραπάνω αρχείο και αφαιρεί από αυτό όλες τις εγγραφές που ικανοποιούν κάποια συνθήκη (π.χ. ηλικία < 18). Στο τέλος, χρησιμοποιήστε ftruncate για να μειώσετε κατάλληλα το μέγεθος του αρχείου.

[Επιστρέψτε αργότερα για περισσότερες ασκήσεις]