Εργαστήριο #3: Δομές ελέγχου

Μορφοποίηση προγραμμάτων.

Όσα αναφέραμε στις εκφωνήσεις των lab1, lab2 για τη μορφοποίηση προγραμμάτων ισχύουν κι εδώ. Επιπλέον, στο lab3 ακολουθήστε τις παρακάτω υποδείξεις:

Τοποθέτηση αγκίστρων

Σε εντολές if να χρησιμοποιείτε πάντα άγκιστρα για το σώμα της εντολής (για το else), ακόμη κι αν το σώμα αποτελείται μόνο από μία εντολή.

Στοίχιση

Ο κανόνας που παρουσιάστηκε στο lab1 (οι εντολές που περικλείονται σε άγκιστρα γράφονται 4 κενά πιο δεξιά από την εντολή στην οποία υπάγονται) ισχύει για όλες τις σύνθετες εντολές (π.χ. if, switch).

Σχόλια

Μη γράφετε σχόλια με ελληνικούς χαρακτήρες γιατί δεν είναι πάντα συμβατοί με το autolab και δε μπορούμε να βαθμολογήσουμε εύκολα τις ασκήσεις σας! Γράψτε αγγλικά ή greeklish.

Οδηγίες για την υποβολή στο autolab

Υποβάλετε το αρχείο lab3.c στο autolab (χωρίς zip).

Αρχεία ελέγχου

Έχουν αναρτηθεί στο eclass ενδεικτικά αρχεία εισόδου και αντίστοιχης εξόδου. Διαβάστε το README που τα συνοδεύει για περισσότερες λεπτομέρειες.

Άσκηση

Αποθηκεύστε το πρόγραμμα που θα γράψετε σε αρχείο με όνομα **lab3.c** .

Σε αυτή την άσκηση θα δείτε ένα παράδειγμα χρήσης δυαδικών (bitwise) τελεστών για την κωδικοποίηση διαφορετικών ειδών δεδομένων σε περιορισμένο αριθμό από bits καθώς και χρήση if και switch.

Το πρόγραμμά σας θα πάρει ως είσοδο ένα δεκαεξαδικό αριθμό μεγέθους 32 bits (τύπου unsigned int) στον οποίο αποθηκεύονται κωδικοποιημένες πληροφορίες εισιτηρίου για ένα ποδοσφαιρικό αγώνα της ομάδας Brentford FC.

Η κωδικοποίηση έχει γίνει ως εξής, ξεκινώντας από το λιγότερο σημαντικό bit:

- Στα 7 λιγότερο σημαντικά bits (θέσεις 0-6) έχει αποθηκευτεί ο αριθμός της θέσης, ο οποίος έχει τιμή από 1 έως και 120.
- Στα επόμενα 5 bits (θέσεις 7-11) έχει αποθηκευτεί η σειρά της θέσης ως "απόσταση" από το γράμμα 'A'. Για παράδειγμα, αν η σειρά είναι 'G' τότε έχει αποθηκευτεί ο αριθμός 6 γιατί το 'G' απέχει 6 γράμματα από το 'A'.
- Στα επόμενα 6 bits (θέσεις 12-17) έχει αποθηκευτεί ένας ακέραιος που αντιστοιχεί στην πύλη του γηπέδου που έχει τιμή από 1 έως και 35.
- Στα υπόλοιπα bits έχει αποθηκευτεί ένας ακέραιος που είναι το άθροισμα των παραπάνω τιμών και χρησιμοποιείται για να ελεγχθεί η εγκυρότητα του εισιτηρίου*.

Μπορείτε να θεωρήσετε ότι οι τιμές είναι πάντα στο σωστό εύρος.

Παράδειγμα:

Τα στοιχεία ενός εισιτηρίου είναι θέση 90, σειρά G, πύλη 18.

Στο δυαδικό σύστημα, το 90 είναι `1011010`, η απόσταση (6) του χαρακτήρα 'G' από τον 'A' είναι `00110` και το 18 είναι `010010`. Το άθροισμα τους είναι 114 το οποίο στο δυαδικό σύστημα είναι `1110010`. Επομένως, το εισιτήριο είναι κωδικοποιημένο ως εξής:

`0000 0001 1100 1001 0010 0011 0101 1010`

Ο παραπάνω αριθμός στο δεκαεξαδικό σύστημα έχει την τιμή `0x01C9235A` .Αυτή θα είναι η είσοδος του προγράμματος.

**Ενδιαφέρουσες πληροφορίες*: Ο δυαδικός κωδικός του εισιτήριου περιέχει πληροφορίες για τον αριθμό θέσης, σειρά και πύλη. Το άθροισμα αυτών περιλαμβάνεται επίσης στον κωδικό του εισιτηρίου και λέγεται checksum. Το checksum είναι ένας αριθμός-έλεγχος που προκύπτει από τις υπόλοιπες τιμές ενός κωδικού και προστίθεται στον κωδικό για να επαληθεύουμε ότι τα υπόλοιπα δεδομένα είναι σωστά και δεν έχουν αλλοιωθεί. Δεν είναι πάντα άθροισμα, αλλά πάντα προκύπτει μετά από κάποια μαθηματική επεξεργασία των υπόλοιπων δεδομένων. Ίσως δεν το ξέρατε αλλά το τελευταίο ψηφίο του ΑΦΜ σας είναι ένα checksum!

Το πρόγραμμά σας πρέπει να λειτουργεί ως εξής:

1. Εκτυπώνει το μήνυμα "`Ticket code:\n`" και διαβάζει τον κωδικό ενός εισιτηρίου ως δεκαεξαδική τιμή την οποία αποθηκεύει σε μία μεταβλητή τύπου unsigned int.
2. Χρησιμοποιεί μάσκες (<u>υποχρεωτικά</u>) και κατάλληλους δυαδικούς (bitwise) τελεστές για να αποσπάσει και να αποθηκεύσει σε ξεχωριστές μεταβλητές τις επιμέρους πληροφορίες που είναι κωδικοποιημένες στον κωδικό του εισιτηρίου. Συστήνουμε να ξεκινήσετε από το λιγότερο σημαντικό bit.
3. Εκτυπώνει μήνυμα στη μορφή "`G R S\n`". Στο μήνυμα, `G` είναι η πύλη, `R` η σειρά ως χαρακτήρας, όπως προκύπτει με βάση την απόσταση από το 'A', και `S` η θέση.
4. Ελέγχει αν το άθροισμα της θέσης, απόστασης και πύλης είναι ίσο με το αντίστοιχο άθροισμα (checksum) που περιέχει ο κωδικός. Αν όχι, εκτυπώνει το μήνυμα "`Invalid ticket.\n`" και τερματίζει άμεσα με χρήση της εντολής `return 1;`
   Διαφορετικά, εκτυπώνει το μήνυμα "`Valid ticket.\n`" και συνεχίζει κανονικά.
5. Εκτυπώνει το μήνυμα "`##\n`".
6. Εκτυπώνει μήνυμα στη μορφή "`C seat.\n`" όπου `C` το είδος της θέσης. Οι θέσεις 1-40 είναι "`Red `", οι θέσεις 41-80 είναι "`White`", και οι θέσεις 81-120 είναι "`Yellow`". Για οποιαδήποτε άλλη τιμή το είδος της θέσης είναι "`Standing`".
7. Εκτυπώνει το μήνυμα "`##\n`".
8. Εκτυπώνει το μήνυμα "`Game time:\n`" και διαβάζει την ώρα και λεπτό έναρξης του αγώνα στη μορφή `Ω:Λ` όπου `Ω` η ώρα και `Λ` το λεπτό. Δε χρειάζεται να ελέγξετε αν η ώρα και το λεπτό είναι έγκυρες τιμές .
9. Εκτυπώνει το μήνυμα "`Now:\n`" και διαβάζει την τωρινή ώρα και λεπτό, και πάλι στη μορφή `Ω:Λ`.
10. Εάν η στιγμή έναρξης του αγώνα είναι προγενέστερη της τωρινής, εκτυπώνει το μήνυμα "`Too late.\n`" και τερματίζει άμεσα με χρήση εντολής `return 2;` . Αν είναι ακριβώς ίδια, εκτυπώνει το μήνυμα "`On time.\n`" , διαφορετικά, εκτυπώνει το μήνυμα "`Too early.\n`". Προσπαθήστε να χρησιμοποιήσετε if με σύνθετες συνθήκες για την υλοποίηση αυτού του βήματος.
11. Εκτυπώνει το μήνυμα "`##\n`".
12. Επειδή η συντομογραφία της ομάδας είναι BRE FC, αποφασίστηκε να δοθεί μια έκπτωση σε όσους βρίσκονται σε μια από τις σειρές B, R, E, F, C. Χρησιμοποιήστε εντολή switch (<u>υποχρεωτικά</u>) ώστε το πρόγραμμα να εκτυπώνει:
    - το μήνυμα "`30% off.\n`" αν η σειρά είναι 'B'
    - το μήνυμα "`25% off.\n`" αν η σειρά είναι 'R'
    - το μήνυμα "`20% off.\n`" αν η σειρά είναι 'E'
    - το μήνυμα "`10% off.\n`" αν η σειρά είναι είτε 'F' είτε 'C'.
    - το μήνυμα "`No sale.\n`" σε κάθε άλλη περίπτωση.