Εργαστήριο #2: Τελεστές

Μορφοποίηση προγραμμάτων.

Όσα αναφέραμε στην εκφώνηση του lab1 για τη μορφοποίηση προγραμμάτων ισχύουν κι εδώ. Επιπλέον, στο lab2 ακολουθήστε τις παρακάτω υποδείξεις.

Ονόματα σταθερών

Τα ονόματα των σταθερών, είτε έχουν οριστεί με const είτε με #define, γράφονται με όλα τα γράμματα κεφαλαία και πρέπει να είναι περιγραφικά, όπως και των μεταβλητών.

Κενά σε εκφράσεις

Βάζουμε κενό πριν και μετά το = σε αναθέσεις, για να διαβάζονται πιο εύκολα οι εκφράσεις. Σε σύνθετες εκφράσεις είναι καλό να υπάρχει κενό πριν και μετά από κάθε τελεστή, για παράδειγμα:

```
final_velocity = init_velocity + acceleration * time; // NAI

final_velocity=init_velocity+acceleration*time;  // OXI
```

Σχόλια

Πέρα από το αρχικό σχόλιο προγράμματος, προσθέτουμε ενδιάμεσα σχόλια όταν χρειάζεται να εξηγηθεί κάτι που δεν είναι προφανές, π.χ. κάποιος πολύπλοκος υπολογισμός. Τα σχόλια μπαίνουν πάνω πάνω από τον σχετικό κώδικα ή, αν είναι *σύντομα* (π.χ. 3-4 λέξεις), στα δεξιά της εντολής, μερικά κενά μετά το ; .

Συμβουλές για την ανάπτυξη κώδικα

Διαβάστε όλη την εκφώνηση κάθε άσκησης για να καταλάβετε τη γενική ιδέα.

Σκεφτείτε ποιες ποσότητες θα αποθηκεύσετε σε μεταβλητές και δώστε προσοχή στους τύπους τους. Θα χρειαστείτε μεταβλητές για τα δεδομένα που δίνονται από το πληκτρολόγιο και για τα τελικά αποτελέσματα. Αν χρειάζεται, ορίστε επιπλέον μεταβλητές για ενδιάμεσα αποτελέσματα πράξεων (π.χ. μετατροπές μονάδων). Χρησιμοποιήστε περιγραφικά ονόματα (δείτε τις υποδείξεις στην εκφώνηση του lab1).

Σκεφτείτε ποιες ποσότητες θα ορίσετε ως σταθερές (ποσότητες που δε μεταβάλλονται κατά τη διάρκεια του προγράμματος). Επιλέξτε περιγραφικά ονόματα (με κεφαλαία) και δώστε προσοχή στους τύπους τους.

Σχεδιάστε στο χαρτί τη διαδικασία υπολογισμού και επιβεβαιώσετε τις πράξεις σας με ένα κομπιουτεράκι.

Όταν χρειάζεται να γίνουν υπολογισμοί στο πρόγραμμα με σταθερές τιμές, αυτοί πρέπει να εκτελούνται μέσα στον κώδικα ώστε το πρόγραμμα να είναι πιο ευέλικτο και σωστό. Δηλαδή, δεν πρέπει να υπολογίζετε τα αποτελέσματα χρησιμοποιώντας εξωτερικά εργαλεία (π.χ. κομπιουτεράκι) και να τα εισάγετε ως σταθερές τιμές στο πρόγραμμα.

Μεταφέρετε όλα τα παραπάνω σε κώδικα. Γράψτε σταδιακά το πρόγραμμά σας, κάνοντας συχνά save, μεταγλώττιση κι εκτέλεση ώστε να διορθώνετε τυχόν λάθη καθώς προχωράτε.

Αρχεία ελέγχου

Σας δίνουμε 3 σετ αρχείων ελέγχου για την πρώτη άσκηση. Τα αρχεία εισόδου λέγονται a_in1, a_in2, a_in3 και τα αντίστοιχα αρχεία αναμενόμενης εξόδου a_out1, a_out2, a_out3.

Σας δίνουμε 2 σετ αρχείων ελέγχου για τη δεύτερη άσκηση. Τα αρχεία εισόδου λέγονται b_in1, b_in2 και τα αντίστοιχα αρχεία αναμενόμενης εξόδου b_out1, b_out2.

Οδηγίες για την υποβολή στο autolab

1)

Πρέπει να ακολουθήσετε τη διαδικασία σχηματισμού ομάδας και για το lab2.

2)

Το lab2 έχει δύο ασκήσεις τις οποίες πρέπει να υποβάλετε. Η πρώτη πρέπει να είναι αποθηκευμένη σε αρχείο με όνομα **lab2a.c** και η δεύτερη σε αρχείο με όνομα **lab2b.c**. Για να μπορέσετε να τα υποβάλετε θα πρέπει να τα "πακετάρετε" σε ένα αρχείο zip. Αυτό μπορείτε να το κάνετε με δύο τρόπους:

Από τερματικό:

Μεταβείτε με cd στον κατάλογο που περιέχει τα αρχεία lab2a.c και lab2b.c και γράψτε την εντολή:

```
zip  -r  lab2.zip  lab2a.c  lab2b.c
```

Εναλλακτικά, από γραφικό περιβάλλον:

Μεταβείτε μέσω Dolphin στον κατάλογο που περιέχει τα αρχεία lab2a.c και lab2b.c, επιλέξτε με το ποντίκι τα αρχεία lab2a.c και lab2b.c, κάντε δεξί κλικ κι επιλέξτε από το μενού Compress και μετά Here (as ZIP).

Σε κάθε περίπτωση, θα δημιουργηθεί το αρχείο **lab2.zip** το οποίο υποβάλετε στο autolab.

ΠΡΟΣΟΧΗ: Για την τελική σας υποβολή, το zip πρέπει να περιέχει και τα δύο αρχεία C και μόνο αυτά.

Άσκηση 1

Αποθηκεύστε το πρόγραμμα για την άσκηση 1 σε αρχείο με όνομα **lab2a.c** .

Μία κατασκευαστική εταιρία θα αναλάβει την υλοποίηση ενός αυτοκινητόδρομου με δύο σταθμούς διοδίων, ένα σε κάθε άκρο. Το κόστος κατασκευής είναι 3.12 εκατομμύρια ευρώ / χιλιόμετρο. Κάθε όχημα πληρώνει το 73% του συνολικού κόστους των δύο διοδίων, καθώς δεν περνούν όλα τα οχήματα και από τα δύο διόδια.

Γράψτε ένα πρόγραμμα που παίρνει ως είσοδο το μήκος του αυτοκινητόδρομου, τον μέσο αριθμό των οχημάτων που διέρχονται σε ημερήσια βάση και το κόστος διέλευσης από κάθε σταθμό διοδίων και υπολογίζει σε πόσα έτη η εταιρεία θα αποσβέσει το έργο. Θεωρήστε ότι το έτος έχει 365 ημέρες.

Το πρόγραμμα λειτουργεί ως εξής:

1. Εκτυπώνει το μήνυμα "`Highway length:\n`" και διαβάζει με μία scanf το μήκος του αυτοκινητόδρομου (πραγματικός αριθμός) και τη μονάδα μέτρησης του μήκους (χαρακτήρας).
2. Εκτυπώνει το μήνυμα "`Vehicles per day:\n`" και διαβάζει το μέσο αριθμό αυτοκινήτων που διέρχονται από τον αυτοκινητόδρομο σε μία ημέρα (πραγματικός αριθμός).
3. Εκτυπώνει το μήνυμα "`Tolls:\n`" και διαβ΄ζει τις τιμές των δύο σταθμών διοδίων (πραγματικοί αριθμοί).
4. Αποθηκεύει σε μεταβλητή το μήκος σε χιλιόμετρα αφού χρησιμοποιήσει τον τελεστή `?:` ώστε να ελέγξει αν η μονάδα μέτρησης μήκους είναι 'm' ή όχι. Αν είναι 'm', τότε το μήκος έχει δοθεί σε μέτρα και πρέπει πρώτα να μετατραπεί σε χιλιόμετρα και το αποτέλεσμα να αποθηκευτεί στη μεταβλητή. Διαφορετικά, το μήκος έχει ήδη δοθεί σε χιλιόμετρα και αποθηκεύεται ως έχει στη μεταβλητή.
5. Υπολογίζει τα έτη έως την απόσβεση του έργου.
6. Εκτυπώνει το μήνυμα: "`Investment of X euros pays off after Y years.\n`" όπου `X` είναι το συνολικό κόστος κατασκευής του αυτοκινητόδρομου και `Y` τα έτη απόσβεσης. Το κόστος εκτυπώνεται με δυο δεκαδικά ψηφία ενώ τα έτη με ένα.

Παραδείγματα εκτέλεσης: Τα δεδομένα που πληκτρολογεί ο χρήστης εμφανίζονται με κόκκινο χρώμα.

Highway length: 64852m Vehicles per day: 3585 Tolls: 4.2 3.6 Investment of 202338240.00 euros pays off after 27.2 years.
Highway length: 95.327k Vehicles per day: 1200.5 Tolls: 5 1.35 Investment of 297420240.00 euros pays off after 146.4 years.

Άσκηση 2

Αποθηκεύστε το πρόγραμμα για την άσκηση 2 σε αρχείο με όνομα **lab2b.c** .

Για τους υπολογισμούς πρέπει να χρησιμοποιήστε μάσκες και δυαδικούς τελεστές (π.χ. ολίσθηση, &, |, ~, ^).

Γράψτε ένα πρόγραμμα το οποίο:

1. Εκτυπώνει το μήνυμα "`Enter number:\n`" και διαβάζει έναν ακέραιο σε δεκαεξαδική μορφή.
2. Εκτυπώνει το μήνυμα "`Enter positions:\n`" και διαβάζει δύο ακεραίους που αναπαριστούν τις θέσεις δύο bits του ακεραίου που διαβάσατε στο βήμα 1. Θυμίζουμε ότι η αρίθμηση των θέσεων ξεκινά από 0 και από το λιγότερο σημαντικό bit. Μπορείτε να υποθέσετε ότι οι αριθμοί που θα δοθούν ως θέσεις είναι μεταξύ 0 και 31 (δηλαδή δε χρειάζεται να κάνετε κάποιο έλεγχο).
3. Εκτυπώνει το μήνυμα "`Bit at position X is Y\n`" όπου `X` η πρώτη θέση που διαβάστηκε και `Y` η τιμή του bit σε αυτή τη θέση.
4. Εκτυπώνει το μήνυμα "`Bit at position X is Y\n`" όπου `X` η δεύτερη θέση που διαβάστηκε και `Y` η τιμή του bit σε αυτή τη θέση.
5. Εκτυπώνει το μήνυμα "`##\n`"
6. Θέτει σε 1 τα δύο bits του ακεραίου που βρίσκονται στις δύο θέσεις.
7. Εκτυπώνει τη νέα τιμή του ακεραίου στο δεκαεξαδικό σύστημα ώστε να καταλαμβάνει 8 θέσεις, με μηδενικά σε τυχόν κενές αρχικές θέσεις. Μετά εκτυπώνει χαρακτήρα αλλαγής γραμμής ('\n').
8. Εκτυπώνει το μήνυμα "`##\n`"
9. Θέτει σε 0 τα bits του ακεραίου που βρίσκονται στις δύο θέσεις.
10. Εκτυπώνει τη νέα τιμή του ακεραίου στο δεκαεξαδικό σύστημα ώστε να καταλαμβάνει 8 θέσεις, με μηδενικά σε τυχόν κενές αρχικές θέσεις. Στο τέλος εκτυπώνει χαρακτήρα αλλαγής γραμμής ('\n').

Παράδειγμα εκτέλεσης: Τα δεδομένα που πληκτρολογεί ο χρήστης εμφανίζονται με κόκκινο χρώμα.

```
Enter number:
ba5e5
Enter positions:
10 18
Bit at position 10 is 1
Bit at position 18 is 0
##
000fa5e5
##
000ba1e5
```

Βοήθεια: Αν η αρχική είσοδος είναι ba5e5 και στο βήμα 2 δοθούν οι ακέραιοι 10 και 18, τότε η δυαδική αναπαράσταση του αριθμού είναι

0000 0000 0000 1011 1010 0101 1110 0101

Το bit στη θέση 10 εμφανίζεται με κίτρινο φόντο και το bit στη θέση 18 με πράσινο.