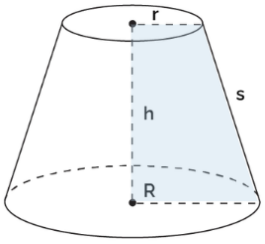
1η Σειρά Ασκήσεων - Συναρτήσεις

Άσκηση 1.1

Να γράψετε συνάρτηση με όνομα `myRand()`, που να δέχεται ως παραμέτρους, δύο ακέραιους αριθμούς. Οι παράμετροι αυτοί αντιστοιχούν στον μικρότερο και μεγαλύτερο τυχαίο ακέραιο αριθμό που θέλουμε να παράγει η συνάρτηση αυτή. Η συνάρτηση θα επιστρέφει έναν τυχαίο ακέραιο αριθμό στο επιθυμητό εύρος.
Στη συνέχεια, να γράψετε πρόγραμμα που να διαβάζει 10 τυχαίους αριθμούς στο διάστημα [10, 99], καλώντας τη συνάρτηση `myRand(int lower, int upper)` και να εμφανίζει το μεγαλύτερο από αυτούς.

Άσκηση 1.2

Ο κόλουρος κώνος παράγεται κατά την περιστροφή ορθογωνίου τραπεζίου περί την κάθετο προς τις βάσεις του πλευρά. Η επιφάνεια και ο όγκος του κόλουρου κώνου, δίνονται από τους ακόλουθους μαθηματικούς τύπους:



$$S = \pi r^2 + \pi R^2 + \pi(R + r)s \quad \text{, όπου} \quad s = \sqrt{h^2 + (R - r)^2}$$

$$V = \frac{1}{3}\pi h \left(R^2 + Rr + r^2\right)$$

Να γράψετε δύο συναρτήσεις για τον υπολογισμό της επιφάνειας και του όγκου του συγκεκριμένου σχήματος. Στη συνέχεια να υλοποιήσετε πρόγραμμα που να καλεί τις δύο συναρτήσεις και να εμφανίζει τις τιμές που επιστρέφουν.

Άσκηση 1.3

Σε μια βιομηχανική μονάδα παραγωγής, το κόστος κατασκευής ενός προϊόντος εξαρτάται από:

- Το κόστος πρώτων υλών
- Το κόστος εργασίας
- Τα γενικά έξοδα παραγωγής (π.χ. ηλεκτρικό ρεύμα, μηχανήματα, συντήρηση, κτλ.)

Η τιμή πώλησης του προϊόντος καθορίζεται από το κόστος παραγωγής του, προσθέτοντας ένα ποσοστό κέρδους. Να γράψετε ένα πρόγραμμα που θα υλοποιεί τις εξής συναρτήσεις:

1. `calcProductionCost`:
    - Δέχεται ως ορίσματα το κόστος πρώτων υλών, το κόστος εργασίας και τα γενικά έξοδα.
    - Επιστρέφει το συνολικό κόστος παραγωγής.
2. `calcSellingPrice`:

- Δέχεται ως ορίσματα το κόστος παραγωγής και το ποσοστό κέρδους.
- Καλεί τη συνάρτηση `calcProductionCost` για να υπολογίσει το κόστος παραγωγής.
- Επιστρέφει την τελική τιμή πώλησης του προϊόντος.

Το κύριο πρόγραμμα θα διαβάζει τα απαραίτητα δεδομένα από τον χρήστη, θα καλεί τις συναρτήσεις και θα εμφανίζει την τελική τιμή πώλησης του προϊόντος.

Άσκηση 1.4

Για τον υπολογισμό της ημέρας της εβδομάδας μιας δεδομένης ημερομηνίας χρησιμοποιείται ο τύπος Zeller's congruence:

$$w = \left(d + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + \left\lfloor \frac{J}{4} \right\rfloor + 5J\right) \mod 7$$

Όπου:

- d: Ημέρα του μήνα.
- m: Μήνας (Μάρτιος=3, ..., Δεκέμβριος=12, Ιανουάριος=13, Φεβρουάριος=14).
- K: Τα δύο τελευταία ψηφία του έτους (π.χ. για το 2025, είναι 25).
- J: Τα πρώτα δύο ψηφία του έτους (π.χ. για το 2025, είναι 20).
- w: Ημέρα της εβδομάδας (0 = Σάββατο, 1 = Κυριακή, ..., 6 = Παρασκευή).

Αν ο μήνας είναι Ιανουάριος (1) ή Φεβρουάριος (2), τότε θεωρείται ότι ανήκει στο προηγούμενο έτος. Δηλαδή, για Ιανουάριο γράφουμε m = 13 και για Φεβρουάριο γράφουμε m = 14. Ταυτόχρονα, μειώνουμε το έτος κατά 1.
Να υλοποιήσετε ένα πρόγραμμα που θα δέχεται από τον χρήστη μια ημερομηνία (ημέρα, μήνα, έτος) και θα υπολογίζει την ημέρα της εβδομάδας χρησιμοποιώντας τον τύπο του Zeller's Congruence.

Οδηγίες Υλοποίησης:

1. Υλοποιήστε μια συνάρτηση `zeller(int d, int m, int y)` που θα δέχεται:
   - `d`: Την ημέρα του μήνα.
   - `m`: Τον μήνα (με τροποποίηση: 3=Μάρτιος, 4=Απρίλιος, ..., 13=Ιανουάριος, 14=Φεβρουάριος).
   - `y`: Το έτος.
   - Η συνάρτηση θα επιστρέφει έναν ακέραιο αριθμό `w`, όπου: 0 = Σάββατο, 1 = Κυριακή, 2 = Δευτέρα, ..., 6 = Παρασκευή.
2. Υλοποιήστε μια συνάρτηση `printDayOfWeek(int w)` που θα δέχεται την τιμή `w` και θα εμφανίζει την αντίστοιχη ημέρα της εβδομάδας.
3. Στη συνάρτηση `main()`, το πρόγραμμα θα:
   - ζητά από τον χρήστη να εισάγει ημέρα, μήνα και έτος. Ο μήνας θα εισάγεται χωρίς τροποποίηση (1=Ιανουάριος, 2=Φεβρουάριος, ..., 12=Δεκέμβριος)

- καλεί τη συνάρτηση `zeller()` για να υπολογίσει την ημέρα της εβδομάδας.
- εμφανίζει το αποτέλεσμα με την `printDayOfWeek()`.

Άσκηση 1.5

Να γραφούν τρεις συναρτήσεις για τις παρακάτω λειτουργίες:

- συνάρτηση που να δέχεται ως όρισμα έναν ακέραιο αριθμό και να τον εμφανίζει αντίστροφα.
- συνάρτηση που να μετρά το πλήθος των ψηφίων ενός ακεραίου.
- συνάρτηση που να μετρά το πλήθος των bit της δυαδικής αναπαράστασης ενός ακεραίου (π.χ. ο αριθμός 9 στο δυαδικό σύστημα γράφεται 1001, άρα πλήθος bit = 4)

Να γραφεί πρόγραμμα που να διαβάζει έναν θετικό ακέραιο αριθμό, να καλεί τις παραπάνω συναρτήσεις και να εμφανίζει τα αποτελέσματα.

Άσκηση 1.6

Στα μαθηματικά τέλειος αριθμός (perfect number) θεωρείται ένας θετικός ακέραιος αριθμός του οποίου το άθροισμα των διαιρετών του (εκτός του ίδιου του αριθμού), είναι ίσο με τον αριθμό αυτό. Ο μικρότερος τέλειος αριθμός είναι ο 6 (1+2+3=6). Άλλοι τέλειοι αριθμοί είναι: 28=1+2+4+7+14, 496=1+2+4+8+16+31+62+124+248.

Να γραφεί συνάρτηση που να δέχεται ως όρισμα έναν αριθμό, να ελέγχει αν αυτό ο αριθμός είναι τέλειος και να επιστρέφει τον αριθμό 1 αν είναι τέλειος ή το 0 αν δεν είναι.

Επίσης, να γραφεί πρόγραμμα που να ζητά από το χρήστη 2 ακέραιους αριθμός και με κατάλληλη χρήση της παραπάνω συνάρτησης, να εντοπίζει και να εμφανίζει τους τέλειους αριθμούς που υπάρχουν ανάμεσα σε αυτούς που έδωσε ο χρήστης. (Σημείωση: οι τέλειοι αριθμοί είναι ελάχιστοι, π.χ. οι 5 πρώτοι είναι: 6, 28, 496, 8128, 33550336)

Άσκηση 1.7

Η μηνιαία πληρωμή (Payment) ενός δανείου μπορεί να υπολογιστεί από τον παρακάτω τύπο:

$$Payment = \frac{Rate*(1+Rate)^{N}}{((1+Rate)^{N}-1)}*Loan$$

όπου:

- `Rate` είναι το μηνιαίο επιτόκιο, δηλαδή το ετήσιο επιτόκιο διαιρεμένο με το 12 (π.χ. 12% ετήσιο επιτόκιο αντιστοιχεί σε 1% μηνιαίο επιτόκιο),
- `N` είναι το πλήθος των πληρωμών και
- `Loan` είναι το ποσό του δανείου.

Να γράψετε μια συνάρτηση για τον υπολογισμό της μηνιαίας πληρωμής (`Payment`) και στη συνέχεια ένα πρόγραμμα που θα χρησιμοποιεί τη συνάρτηση. Η συνάρτηση θα έχει ως ορίσματα τα `Rate, N, Loan`. Το πρόγραμμα θα πρέπει να έχει τη δυνατότητα να εμφανίζει αποτελέσματα για πολλές διαφορετικές τιμές των ορισμάτων της συνάρτησης. Η εμφάνιση δεδομένων και αποτελεσμάτων θα γίνεται στη main() με την παρακάτω μορφή:

```
Ποσό δανείου:          € 10000.00
Μηνιαίο επιτόκιο:      1%
```

Πλήθος πληρωμών:	36
Μηνιαία πληρωμή:	€ 332.14
Συνολικό ποσό πληρωμής:	€ 11957.16
Συνολικός τόκος:	€ 1957.16

Άσκηση 1.8

Για τον υπολογισμό της τιμής του n! όταν η τιμή του n είναι μεγάλη, χρησιμοποιείται ο προσεγγιστικός τύπος του Stirling:

$$n! \approx e^{-n} n^n \sqrt{2\pi n}$$

Να γραφούν 2 διαφορετικές συναρτήσεις που να δέχονται ως είσοδο έναν ακέραιο αριθμός και να υπολογίζουν το παραγοντικό του με:

- τον προσεγγιστικό τύπο του Stirling
- χρήση του τύπου: `n! = 1 x 2 x 3 x ... x n`

Και οι δύο συναρτήσεις θα επιστρέφουν τιμή τύπου double.
Να υλοποιήσετε ένα πρόγραμμα που να δέχεται έναν ακέραιο αριθμό και να υπολογίζει το παραγοντικό του με κατάλληλη κλίση των παραπάνω συναρτήσεων. Να βρίσκει επίσης και να εκτυπώνει τη διαφορά (ως ποσοστό %) που παρουσιάζουν οι 2 μέθοδοι υπολογισμού.

Άσκηση 1.9

Χρησιμοποιήστε τη συνάρτηση της προηγούμενης άσκησης που υπολογίζει το παραγοντικό ενός αριθμού, για να υπολογίσετε το παρακάτω άθροισμα:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

με ακρίβεια 10^{-5} (σταματάμε την επανάληψη όταν ο τρέχων όρος γίνει μικρότερος από 10^{-5}). Η τιμή του x είναι τύπου double και εισάγεται με χρήση της scanf. Το μέγιστο επιτρεπόμενο πλήθος επαναλήψεων είναι 15.

Άσκηση 1.10

Να γράψετε συνάρτηση που να δέχεται έναν ακέραιο αριθμό `k ≥ 2`, και να ελέγχει αν αυτός ο αριθμός είναι πρώτος (prime) ή όχι, επιστρέφοντας κατάλληλη τιμή.
Στη συνέχεια να γραφεί ένα πρόγραμμα που να καλεί τη συνάρτηση και να εμφανίζει στην οθόνη όλους τους πρώτους αριθμούς που είναι μικρότεροι μιας γνωστής μέγιστης τιμής `max` που ορίζεται μέσω μιας δήλωσης `#define`.

Άσκηση 1.11

Οι H.A.Brothers και J.A. Knox ανακάλυψαν ότι καθώς η τιμή του `x` γίνεται μεγαλύτερη, η τιμή της παρακάτω έκφρασης:

$$\left(\frac{2x+1}{2x-1}\right)^x$$

προσεγγίζει τη μαθηματική σταθερά `e`. Να γράψετε συνάρτηση που να υλοποιεί την παραπάνω μαθηματικό τύπο. Στη συνέχεια να γραφεί ένα πρόγραμμα που να καλεί επαναληπτικά τη συνάρτηση αυτή, μέχρι η απόλυτη τιμή της διαφοράς του `x` από τη συνάρτηση `exp` (της βιβλιοθήκης `math.h`) να γίνει μικρότερη του 10^{-6}. Το πρόγραμμα να εμφανίζει την τελική τιμή του `x`.

Άσκηση 1.12

Ένας ακέραιος αριθμός, ονομάζεται αριθμός Armstrong αν ισχύει: (άθροισμα των ψηφίων του, καθένα υψωμένο στη δύναμη του πλήθους των ψηφίων) = ο ίδιος ο αριθμός.

Παραδείγματα: $153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27$

$9474 = 9^4 + 4^4 + 7^4 + 4^4$

Να γραφεί συνάρτηση που δέχεται έναν θετικό ακέραιο αριθμό, να ελέγχει αν είναι Armstrong, επιστρέφοντας κατάλληλη τιμή.

Στη συνέχεια να γραφεί ένα πρόγραμμα που να εμφανίζει όλους τους αριθμούς Armstrong που υπάρχουν στο διάστημα [10, 9999].