5η Προγραμματιστική άσκηση για το μάθημα
343 Εισαγωγή στον Προγραμματισμό
Μιχάλης Μπέκος / Κωνσταντίνα Τζουβάρα

Χειμερινό εξάμηνο 2022-2023
Πανεπιστήμιο Ιωαννίνων
14/11/2022

5η Προγραμματιστική Εργασία

Ημερομηνία παράδοσης: 20.11.2022

Συζήτηση λύσης - επίλυση απορίων: 23.11.2022

Στόχοι: Εξοικείωση με τη χρήση μονοδιάστατων πινάκων και με τη χρήση πινάκων ως παραμέτρους συναρτήσεων και ως πεδία κλάσεων.

Άσκηση 1.

Θέλουμε να συμπληρώσουμε έναν μονοδιάστατο πίνακα 20 στοιχείων, με τυχαίες ακέραιες τιμές στο διάστημα [0, 100] και στη συνέχεια να βρούμε το πλήθος των εμφανίσεων ενός δοθέντος στοιχείου στον πίνακα καθώς και το μέσο όρο των περιττών στοιχείων του πίνακα. Πιο συγκεκριμένα:
Για κάθε έναν υπολογισμό, φτιάξτε μια ξεχωριστή συνάρτηση:
α) Φτιάξτε τη συνάρτηση `fillTable(int A[], int n)`, η οποία θα γεμίζει με τυχαίες ακέραιες τιμές στο διάστημα [0, 100] έναν μονοδιάστατο πίνακα `A`, `n` θέσεων (με χρήση της συνάρτησης rand()).
β) Φτιάξτε τη συνάρτηση `countElement(int A[], int n, int element)` η οποία θα δέχεται ως παραμέτρους έναν μονοδιάστατο πίνακα `A` με `n` στοιχεία, καθώς και έναν ακέραιο αριθμό `element` και θα επιστρέφει το πλήθος των εμφανίσεων του `element` στον `A`.
γ) Φτιάξτε τη συνάρτηση `meanOdd(int A[], int n)` η οποία θα δέχεται ως παραμέτρους έναν μονοδιάστατο πίνακα `A` με `n` στοιχεία και θα επιστρέφει το μέσο όρο των περιττών στοιχείων του πίνακα.
Ενσωματώστε τις συναρτήσεις σε κατάλληλο δοκιμαστικό πρόγραμμα, το οποίο θα τις καλεί και θα εμφανίζει τα αποτελέσματά τους.

Άσκηση 2.

Στη διάλεξη είδατε την υλοποίηση της Στοίβας (`Stack`) με χρήση διανύσματος στην οποία οι προσθήκες (`push`) και οι διαγραφές (`pop`) ακολουθούν το μοντέλο `Last In First Out`. Η αντίστοιχη δομή, στην οποία οι προσθήκες (`enqueue`) και οι διαγραφές (`dequeue`) ακολουθούν το μοντέλο `First In First Out`, ονομάζεται Ουρά (`Queue`). Στην εργασία αυτή, θα πρέπει να αναπτύξετε την κλάση `Queue` για την υλοποίηση με χρήση διανύσματος μιας ουράς. Καθώς η υλοποίηση σας θα είναι παρόμοια με αυτή της Στοίβας, κατεβάστε από το E-Course και μελετήστε το αρχείο Stack.cpp, το οποίο θα βρείτε στο συνοδευτικό υλικό του τρίτου μέρους των διαλέξεων.

Για την υλοποίηση αρχικά θα χρειαστεί να ορίσετε μια καθολική σταθερά QUEUE_MAX_SIZE, η οποία αντιστοιχεί στο μέγιστο αριθμό στοιχείων (έστω 1000) που μπορεί να αποθηκευθούν σε μια ουρά σύμφωνα με την υλοποίηση σας. Για την υλοποίηση της κλάσης `Queue` χρησιμοποιήστε τα παρακάτω (ιδιωτικά) πεδία:

```
int elements[QUEUE_MAX_SIZE];
int index;
```

Αντικείμενα της κλάσης `Queue` υποστηρίζουν τις παρακάτω (δημόσιες) μεθόδους:

`Queue()`	Κατασκευάζει μια κενή ουρά (αρχικοποιώντας το πεδίο index στην τιμή -1).
`Queue(int A[], int N)`	Κατασκευάζει μια νέα ουρά (αρχικοποιώντας αρχικά το πεδίο index στην τιμή -1) στην οποία ακολούθως ενθέτει (μέσω της μεθόδου `enqueue()` που ακολουθεί) ένα προς ένα όλα τα στοιχεία του πίνακα A σε αυτή. Η σειρά με την οποία γίνεται η ένθεση είναι: A[0], A[1], …, A[N-1].
`bool empty()`	Ελέγχει αν η ουρά είναι κενή.
`int size()`	Επιστρέφει το πλήθος των στοιχείων στην ουρά.
`void enqueue(int elem)`	Αν η ουρά δεν είναι γεμάτη, ενθέτει το δοθέν στοιχείο στην ουρά (στην πρώτη από αριστερά διαθέσιμη θέση του πίνακα `elements`).
`void dequeue()`	Αν η ουρά δεν είναι άδεια, διαγράφει από αυτή το στοιχείο στην θέση 0 του πίνακα `elements` και ακολούθως μειώνει την τιμή του πεδίου `index` κατά ένα. Η διαγραφή γίνεται μετατοπίζοντας τα στοιχεία στις θέσεις `1,2,…,index` του πίνακα `elements` κατά μια θέση αριστερά, δηλ, το στοιχείο στη θέση `0` αντικαθίσταται από αυτό στη θέση `1` κ.ο.κ.
`int peek()`	Αν η λίστα δεν είναι κενή επιστρέφει το στοιχείο στη θέση 0 του πίνακα `elements`. Διαφορετικά επιστρέφει -1.

Να γραφεί η συνάρτηση-οδηγός main() για τον έλεγχο της υλοποίηση σας, η οποία:

1. Δημιουργεί ένα μονοδιάστατο πίνακα ακεραίων `A` με 10 θέσεις.
2. Αρχικοποιεί τον πίνακα `A` με τις τιμές `0,1,…,9`.
3. Αρχικοποιεί ένα αντικείμενο τύπου Queue χρησιμοποιώντας τον δεύτερο κατασκευαστή (που περιγράφηκε παραπάνω) και τον πίνακα A που δημιουργήθηκε στα βήματα 1-2 ως όρισμα του.
4. Ενθέτει 3 ακόμη στοιχεία στην ουρά που δημιουργήθηκε στο βήμα 3 (μέσω της μεθόδου `enqueue()`).
5. Ακολουθεί μια εντολή ανακύκλωσης, σύμφωνα με την οποία, όσο η ουρά που δημιουργήθηκε στα βήματα 3 και 4 δεν είναι κενή, εκτυπώνει στο τερματικό το πρώτο στοιχείο της ουράς (μέσω της μεθόδου `peek()`) και ακολούθως το διαγράφει από την ουρά (μέσω της μεθόδου `dequeue()`).

Πριν την παράδοση της εργασίας σας, εμπλουτίστε τον κώδικα σας με σχόλια.