7η Προγραμματιστική άσκηση για το μάθημα
343 Εισαγωγή στον Προγραμματισμό
Μιχάλης Μπέκος / Κωνσταντίνα Τζουβάρα

Χειμερινό εξάμηνο 2025-2026
Πανεπιστήμιο Ιωαννίνων
15/11/2025

7η Προγραμματιστική Εργασία

Στόχοι: Εξάσκηση στη χρήση μονοδιάστατων πινάκων και εξοικείωση με τις μεθόδους ταξινόμησης.

Άσκηση 1:

Στη διάλεξη είδατε την υλοποίηση της `Στοίβας (Stack)` με χρήση διανύσματος (fixed-size array). Την υλοποίηση μπορείτε να τη βρείτε στο συνοδευτικό υλικό των διαλέξεων στο αρχείο Stack.cpp. Η στοίβα ακολουθεί το πρότυπο **Last In – First Out (LIFO)** και υποστηρίζει τις λειτουργίες **`push()`, `pop()`, `top()`, `empty()`** και **`size()`.**

Στην εργασία αυτή θα υλοποιήσετε την **`ουρά (Queue)`**, η οποία ακολουθεί το πρότυπο **First In – First Out (FIFO)**, και υποστηρίζει τις λειτουργίες:

- `enqueue(e)`**:** εισαγωγή στοιχείου στο τέλος της ουράς.
- `dequeue()`**:** διαγραφή του πρώτου στοιχείου της ουράς.
- `front()`**:** επιστρέφει το πρώτο στοιχείο της ουράς χωρίς διαγραφή.
- `size()`**:** επιστρέφει το πλήθος των στοιχείων της ουράς.
- `empty()`**:** επιστρέφει true αν και μόνο αν η ουρά είναι άδεια.

Η υλοποίηση της ουράς θα πρέπει να γίνει χρησιμοποιώντας **δύο** αντικείμενα τύπου `Stack`. Η ιδέα στην προσέγγιση αυτή είναι η εξής:

- Η ένθεση (enqueue) γίνεται στην πρώτη από τις δύο στοίβες.
- Η διαγραφή (dequeue) και η ανάγνωση (front) γίνονται από τη δεύτερη στοίβα.
- Προηγουμένως όμως γίνεται έλεγχος αν η δεύτερη στοίβα είναι κενή. Στη θετική περίπτωση, όλα τα στοιχεία της πρώτης στοίβας μεταφέρονται σε δεύτερη, ώστε να αναστραφεί η σειρά τους.
- Έτσι, τα πιο «παλιά» στοιχεία μεταφέρονται στο πάνω μέρος της δεύτερης στοίβας.

Για την υλοποίηση της κλάσης `Queue` χρησιμοποιήστε τα παρακάτω (ιδιωτικά) πεδία:

- **`Stack s1;`** // πρώτη στοίβα για τις ενθέσεις
- **`Stack s2;`** // δεύτερη στοίβα για τις διαγραφές

Στο ιδιωτικό τμήμα της κλάσης `Queue` υλοποιήστε επίσης την παρακάτω βοηθητική μέθοδο, η οποία θα καλείται από τις `dequeue()` και `front()`:

void shiftStacks()	Αν η `s2` είναι κενή, μεταφέρει όλα τα στοιχεία της `s1` στη `s2` καλώντας επαναληπτικά `pop()` από την `s1` και `push()` στην `s2` μέχρι η `s1` να γίνει κενή.

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

Queue()	Κατασκευάζει μια κενή ουρά.
bool empty()	Επιστρέφει `true` αν και μόνο αν και οι δύο στοίβες είναι κενές.
int size()	Επιστρέφει το συνολικό πλήθος στοιχείων στην ουρά (άθροισμα των μεγεθών των δύο στοιβών).
void enqueue(int elem)	Καλεί τη μέθοδο `push(elem)` της στοίβας `s1`.
void dequeue()	• Αν και οι δύο στοίβες είναι κενές, δεν κάνει τίποτα. • Διαφορετικά, καλεί την μέθοδο shiftStacks() και καλεί την pop() στην s2.
int front()	• Αν η ουρά είναι κενή, επιστρέφει -1. • Διαφορετικά, καλεί την μέθοδο `shiftStacks()` και επιστρέφει το αποτέλεσμα της κλήσης της `top()` στην `s2`.

Τέλος, να γραφεί η συνάρτηση-οδηγός `main()` για τον έλεγχο της υλοποίησής σας, η οποία:

1. Δημιουργεί ένα μονοδιάστατο πίνακα ακεραίων Α με 10 θέσεις.
2. Αρχικοποιεί τον πίνακα Α με τις τιμές `0,1,…,9`.
3. Δημιουργεί ένα αντικείμενο τύπου `Queue`.
4. Ενθέτει στην ουρά όλα τα στοιχεία του πίνακα Α.
5. Ενθέτει επιπλέον 3 στοιχεία (π.χ. 100, 200, 300).
6. Ακολουθεί μια εντολή ανακύκλωσης, σύμφωνα με την οποία, εκτυπώνεται το πρώτο στοιχείο της ουράς (μέσω της μεθόδου `front()`) και στη συνέχεια καλείται η μέθοδος `dequeue()` για να αφαιρεθεί.

Άσκηση 2:

Στην τάξη μελετήσαμε διάφορους τρόπους ταξινόμησης ενός μονοδιάστατου πίνακα (με συγκρίσεις). Στην άσκηση αυτή θα υποθέσουμε ότι τα προς ταξινόμηση στοιχεία του πίνακα είναι ακέραιοι αριθμοί στο διάστημα `[0,N-1]`, όπου το `N` είναι μια καθολική σταθερά (έστω 100). Η ταξινόμηση ενός τέτοιου πίνακα `A` μπορεί να γίνει χωρίς να κάνουμε κάποια σύγκριση ως εξής:

- Δοθέντος του πίνακα `A`, δημιουργούμε ένα `1xN` μονοδιάστατο πίνακα `B`, τον οποίο τον αρχικοποιούμε ως εξής: Το στοιχείο `B[i]` στη θέση `i` του πίνακα του πίνακα `B` έχει την τιμή `j` αν και μόνο αν ο πίνακας `A` έχει ακριβώς `j` στοιχεία με τιμή `i` (όπου $0 \le i \le N-1$ και $0 \le j \le N$). Παράδειγμα: Αν `N=8`, και `A=[1,4,1,2,7]` τότε `B=[0,2,1,0,1,0,0,1]`
- Παρατηρήστε ότι ο υπολογισμός του πίνακα `B` μπορεί να γίνει θέτοντας αρχικά όλα του τα στοιχεία στην τιμή `0` και ακολούθως κάνοντας μια διαπέραση του πίνακα `A` ανανεώνουμε τις τιμές του πίνακα `B` ως εξής: Για `i=0,…,N-1,` αν `A[i]=k`, τότε αυξάνουμε την τιμή του `B[k]` κατά `1`.

- Έχοντας υπολογίσει τον πίνακα `B`, τότε η ταξινόμηση του πίνακα A αντιστοιχεί στην εκτύπωση των στοιχείων του πίνακα `B` ως εξής: Για `i=0,…,N-1` εκτυπώνουμε `B[i]` φορές τον ακέραιο `i`. Για το παράδειγμα που δώσαμε παραπάνω, αυτό αντιστοιχεί σε: `1,1,2,4,7`.

Να υλοποιήσετε την συνάρτηση `sortAndPrint(int[] A, int n, int max)`, η οποία δέχεται ως παράμετρο έναν πίνακα `A` μεγέθους `n`, του οποίου τα στοιχεία ανήκουν στο διάστημα `[0,max-1]`, και ακολουθώντας τον κανόνα που δόθηκε προηγουμένως εκτυπώνει στο τερματικό τον πίνακα `A` ταξινομημένο.

Ενσωματώστε τη συνάρτηση σε κατάλληλο δοκιμαστικό πρόγραμμα, το οποίο θα την καλεί και θα εμφανίζει το αποτέλεσμα της.