Δομές Δεδομένων

Γεωργία Χρονοπούλου
Ελένη Στρίκλαντ

Υλοποίηση StringQueue & StringStack:

Ζητούμενο του πρώτου μέρους ήταν η υλοποίηση δύο κλάσεων για τις διεπαφές StringQueue και StringStack. Χρησιμοποίησα τον κώδικα που μας δόθηκε ως βοηθητικός της εργασίας και τον μετέτρεψα ως generic ώστε να μπορεί να διαχειριστεί όχι μόνο Sting αντικείμενα, αλλά οποιοδήποτε τύπου του δινόταν. Στις υλοποιήσεις θα παρατηρήσετε οτι οι μέθοδοι peek() και size() είναι κοινές και στις δύο δομές δεδομένων. Επιπλέον, η μέθοδος print είναι κοινή, όμως για πιο κατανοητό κώδικα, στις υλοποιήσεις ονομάζομαι printStack , για την στοίβα και printQueue για την ουρά. ΄Οπως ζητείται και στην εκφώνηση η υλοποίηση των δομών εχει πραγματοποιηθεί με λίστα μονής σύνδεσης. Πιο συγκεκριμένα στην ουρά, για να εκτελούνται οι μέθοδοι put() και get() σε χρόνο O(1), χρησιμοποιήθηκε FIFO εισάγοντας ένα στοιχείο από πίσω και εξάγοντας ένα στοιχείο από μπροστά. Σε αντίθεση στην στοίβα χρησιμοποιήθηκε LIFO που εισάγει pop() και εξάγει push() στοιχεία από μπροστά. Για λειτουργία της στοίβας δεν μας χρειάζονται 2 δείκτες αλλα μόνο ο ένας, ο οποίος δείχνει πάντα στην κορυφή της στοίβας. Με αυτό τον τρόπο η στοίβα δεν επηρεάζεται από n στοιχεία και γίνεται εκτέλεση των μεθόδων της σε όμοιο χρόνο, O(1), όπως και της ουράς.

Υλοποίηση Thiseas:

Ζητούμενο του δεύτερου μέρους ήταν η δημιουργία ενός προγράμματος που θα διασχίζει έναν λαβύρινθο με σκοπό να βρεθεί αν υπάρχει έστω και μία έξοδος. Το πρόγραμμα ξεκινάει διαβάζοντας από το command line το αρχείο ως argument(Εκτέλεση: java Thiseas pathtofile/filename.txt). Μέσω ενός BufferedReader διαβάζετε το αρχείο γραμμή προς γραμμή τοποθετώντας τα δεδομένα του αρχείου στις ορισμένες για αυτό τον σκοπό μεταβλητές. Για την υλοποίηση του ζητούμενου χρησιμοποιήθηκαν 2 στοίβες έτσι όπως ορίστηκαν στο Μέρος Α. Οι στοίβες αυτές αποθηκεύουν το μονοπάτι που ακολουθείτε από το πρόγραμμα για να βρεθεί η έξοδος. Η stack1 τον αριθμό της γραμμής (x), ενώ η stack2 τον 1 αριθμό της στήλης (y). Μέσω των μεθόδων CheckDown, CheckUp, CheckLeft, CheckRight γίνεται έλεγχος για την κατεύθυνση που μπορεί να ακολουθηθεί (αυτό εξαρτάται από το αν βρω 0 ή 1) ώστε να γίνει push των currentX και currentY στις στοίβες stack1 και stack2 αντίστοιχα. Στις μεθόδους που προαναφέρθηκαν γίνεται και αλλαγή στο array που έχουμε αποθηκεύσει τον λαβύρινθο. Η αλλαγή αυτή μετατρέπει το ¨0’ σε ”ok” ώστε να αναγνωρίζεται οτι έχουμε ξαναπεράσει από αυτό το σημείο ώστε να μην βρεθούμε σε αδιέξοδο. Οι μέθοδοι αυτοί γυρίζουν πάντα μια boolean μεταβλητή flag2 η οποία χρησιμοποιείται για τον έλεγχο της ύπαρξης ή όχι εξόδου από τον λαβύρινθο. Εάν η μεταβλητη αυτή είναι true γίνεται έλεγχος για το αν έχει βρεθεί η έξοδος αλλιώς χρησιμοποιείται η μέθοδος pop() των στοιβών ώστε να αφαιρεθει το πρώτο στοιχείο αυτών. Σε περίπτωση που και οι δύο στοίβες είναι άδειες σημαίνει ότι ο λαβύρινθος δεν έχει έξοδο. Τότε μια άλλη boolean μεταβλητή σταματάει τον έλεγχο για εξόδους και τυπώνει αντίστοιχο μήνυμα. Για καλύτερη κατανόηση της υλοποίησης έχουν προστεθεί αρκετα σχόλια στον κώδικα. Παραδίδω και μερικά από τα αρχεία εισόδου που χρησιμοποιήθηκαν για τον έλεγχο ορθότητας(1ExitMaze, 2ExitMaze, 3ExitMaze)