Άσκηση 2η: Πρόγραμμα Heap

Στις 3 πρώτες γραμμές κώδικα γίνεται η συμπερίληψη των κατάλληλων header files που θα χρειαστούμε, το Random και το functional για την λειτουργία της γεννήτριας τυχαίων αριθμών και το iostream για το standard Output και Input στο πρόγραμμα. Έπειτα δηλώνουμε ότι χρησιμοποιούμε το namespace std για μεγαλύτερη ευκολία και ταχύτητα στη συγγραφή του κώδικα.

Στις επόμενες 100 περίπου γραμμές κώδικα υλοποιούμε την κλάση MaxHeap, για δημιουργία και επεξεργασία του σωρού μεγίστων. Αρχικά, στο δημόσιο μέρος της, έχουμε των Constructor της, που αρχικοποιεί το μέγιστο δυνατό μέγεθος του σωρού, των δυναμικά δεσμευμένο πίνακα που αντιπροσωπεύει τον σωρό (με μέγεθος κατά 1 μεγαλύτερο από το μέγιστο μέγεθος του σωρού, αφού ορίζουμε πως η αρίθμησή του ξεκινάει από το 1 και όχι το 0) και το μέγεθος που έχει στη παρούσα φάση, που είναι 0. Έπειτα έχουμε φτιάξει τον Destructor του σωρού, που διαγράφει τον δυναμικά δεσμευμένο πίνακα που αναφέραμε πιο πάνω και απελευθερώνει την αντίστοιχη μνήμη, που δεσμεύσαμε δυναμικά.

Ύστερα, έχουμε φτιάξει την μέθοδο Insert, η οποία δέχεται ως όρισμα έναν float αριθμό και τον εισάγει μέσα στον σωρό μεγίστων. Εμφανίζει μήνυμα λάθους σε περίπτωση που δεν υπάρχει διαθέσιμος χώρος στο σωρό. Γενικά, ξεκινάμε από την θέση του σωρού που μόλις δημιουργήσαμε και προχωράμε σταδιακά προς τη ρίζα, πραγματοποιώντας ελέγχους για το εάν βρισκόμαστε σε αυτή καθώς και για το αν παραβιάζεται η βασική ιδιότητα του δένδρου μεγίστων, ότι η τιμή κάθε κόμβου είναι μεγαλύτερη ή ίση των τιμών των παιδιών του. Εάν δεν συμβαίνει κάτι από τα δύο, τότε εισάγουμε τον float x στη θέση i. Διαφορετικά, με τον βρόχο while, μετακινούμε το στοιχείο της θέσης i/2 στην θέση i και προσχωρούμε το στοιχείο της θέσης i στον πατέρα, δηλαδή της θέση i/2.

Μετέπειτα, υλοποιούμε τη συνάρτηση διαγραφής ενός στοιχείου από τον σωρό μεγίστων, την DeleteMax. Πρώτα, πραγματοποιούμε έλεγχο για το αν ο σωρός είναι άδειος, οπότε και δεν μπορούμε να διαγράψουμε άλλο του στοιχείο και εμφανίζουμε κατάλληλο μήνυμα. Εάν δεν συμβεί αυτό, το μέγιστο στοιχείο του σωρού (το οποίο βρίσκεται στη θέση heap[1]), αποθηκεύεται στη μεταβλητή x, ενώ το στοιχείο στην τελευταία θέση του σωρού (δηλαδή στην heap[CurrentSize]) αποθηκεύεται στην μεταβλητή y και το μέγεθος του σωρού ελαττώνεται κατά 1. Με τον βρόχο while, αρχίζουμε μια αναζήτηση της κατάλληλης θέσης για να επανεισάγουμε το y. Αυτό, ξεκινώντας από την ρίζα και προχωρώντας προς τα κάτω.

Στη συνέχεια έχουμε την μέθοδο Initialize, η οποία μετατρέπει, ουσιαστικά έναν απλό, δυναμικά δεσμευμένο πίνακα, σε πίνακα – σωρό μεγίστων, αξιοποιώντας την έτοιμη μέθοδο του βιβλίου των Δομών Δεδομένων, Αλγορίθμων και Εφαρμογών στη C++, του Dr. Sahni, στη σελίδα 428, όπως μας προτρέπει και η εκφώνηση της άσκησης. Με τον αλγόριθμο αυτό, προσαρμόζουμε τα ιδιωτικά μέλη του σωρού μεγίστων, έτσι ώστε ο πίνακας a να αναλαμβάνει το ρόλο του heap. Στο βρόχο for αρχίζουμε στον τελευταίο κόμβο του δυαδικού δένδρου που αντιστοιχεί στον πίνακα heap (το οποίο είναι πλέον ισοδύναμο του πίνακα a), ο οποίος κόμβος έχει παιδί και εργαζόμαστε στο μονοπάτι προς τη ρίζα. Σε κάθε τοποθέτηση της μεταβλητής i, ο εσωτερικός βρόχος while μας εξασφαλίζει ότι το υποδένδρο με ρίζα στη θέση i είναι σωρός μεγίστων.

Η τελευταία σημαντική μέθοδος της κλάσης MaxHeap είναι η Contraction ή συρρίκνωση, η οποία υλοποιεί την ζητούμενη διαδικασία όπου τα δύο μεγαλύτερα στοιχεία εξάγονται, προστίθενται μεταξύ τους και το αποτέλεσμα της πρόσθεσης αυτής εισάγεται εκ νέου στο σωρό μεγίστων. Η διαδικασία επαναλαμβάνεται Ν φορές και αξιοποιεί τις μεθόδους Insert και DeleteMax που περιγράψαμε προηγουμένως. Τελικά καταλήγουμε σε μοναδικό στοιχείο της σωρού.

Στο τέλος της κλάσης έχουμε τη μέθοδο που εκτυπώνει το τελευταίο στοιχείο του σωρού μεγίστων, όπως αυτό έχει διαμορφωθεί μετά τη συρρίκνωσή του. Στο ιδιωτικό κομμάτι της κλάσης έχουμε τα μέλη του σωρού που τον χαρακτηρίζουν, δηλαδή του τρέχων μέγεθός του, το μέγιστο μέγεθός του και το float pointer heap που τον ορίζει.

Στις επόμενες 100 περίπου γραμμές έχουμε την αντίστοιχη κλάση για τον σωρό ελαχίστων, την MinHeap. Η κλάση αυτή είναι στο μεγαλύτερό της κομμάτι ίδια με την MaxHeap, αφού παρουσιάζει μικρές διαφορές στον κώδικά της, οι οποίες έχουν γίνει για να προσαρμοστούν σωστά οι μέθοδοι της στην βασική ιδιότητα των δένδρων ελαχίστων, η οποία είναι πως η τιμή κάθε κόμβου του πρέπει να είναι μικρότερη ή ίση των τιμών των παιδιών του. Τέτοιες διαφορές συναντάμε στη γραμμή 125, στη γραμμή 146 αλλά και στη γραμμή 174, όπου η φορά της 2ης ανισότητας αντιστρέφεται, ώστε να συμφωνεί με την αντίστροφη λογική των ιδιοτήτων που χαρακτηρίζουν τα δένδρα μεγίστων και ελαχίστων αντίστοιχα, όπως τις αναφέραμε πιο πάνω. Κατά τα άλλα θα μπορούσαμε να πούμε ότι οι δύο αυτές κλάσεις είναι σχεδόν πανομοιότυπες.

Στις τελευταίες 30 γραμμές του προγράμματος υλοποιείται η main και γίνονται οι βασικές διαδικασίες του προγράμματος, μέσα σε μια δομή try-catch για εντοπισμό τυχών runtime σφαλμάτων, ως εξής: Πρώτα γίνεται η ενδεδειγμένη υλοποίηση της γεννήτριας τυχαίων αριθμών και έπειτα ζητείται από τον χρήστη το μέγεθος των πινάκων που έπειτα θα γίνουν σωροί, μέσα σε έλεγχο εγκυρότητας τιμών, ώστε να βεβαιωθούμε ότι ο αριθμός που θα δοθεί θα είναι θετικός. Έπειτα, βάση αυτών, γίνεται η δυναμική δέσμευση του χώρου που θα χρειαστούν οι πίνακες που έπειτα θα γίνουν σωροί αλλά και η δημιουργία τους. Τέλος γίνεται δυναμική αποδέσμευση μνήμης. Συνεχίζουμε αποδίδοντας τιμές float στον πίνακα heap\_min με τη βοήθεια της γεννήτριας τυχαίων αριθμών και έπειτα των αντιγράφουμε στον πίνακα heap\_max, όπως θέλει η εκφώνηση. Ορίζουμε αντικείμενα τύπου MinHeap και MaxHeap, τα κάνουμε instantiate με τα αντίστοιχα constructors, και καλούμε για κάθε ένα από αυτά τις μεθόδους Initialize και Contraction, της δικιάς τους κλάσης. Καλούμε και τις αντίστοιχες μεθόδους Output με τις οποίες εκτυπώνουμε τα τελικά στοιχεία των σωρών ελαχίστων και μεγίστων και αποθηκεύουμε τις τιμές αυτές στις μεταβλητές temp1 και temp2, αντίστοιχα, τις οποίες χρησιμοποιούμε για να συγκρίνουμε τα αποτελέσματα και να καταλήξουμε στο τελικό μας συμπέρασμα. Τέλος, γίνεται αποδέσμευση του χώρου που καταλάβαμε δυναμικά, μόνο με τον πίνακα heap\_min, το οποίο επιδρά ταυτόχρονα και στον πίνακα heap\_max, αφού και αυτός δείχνει στις θέσεις του πίνακα heap\_min (λόγω της αντιγραφής του heap\_min στον heap\_max), θέσεις οι οποίες θα έχουν ήδη αποδεσμευθεί. Στο catch block δεχόμαστε τα μηνύματα από τυχών exceptions που μπορεί να προκλήθηκαν κατά το «τρέξιμο» του προγράμματος και εκτυπώνουμε τα μηνύματα που έχουμε δημιουργήσει για το καθένα τους, μέσω της μεταβλητής msg (pointer), με το cerr. Αξίζει επίσης να σημειωθεί ότι όταν αρχικοποιούμε ένα αντικείμενο αντί για τον τύπο του χρησιμοποιούμε το keyword «auto», οπότε και ο τύπος του αντικειμένου καθορίζεται αποκλειστικά από το δεξί μέρος της εντολής δήλωσης και αρχικοποίησής του (το initializer του) και έτσι δεν χρειάζεται να ανησυχούμε για αλλαγή τύπου δεδομένων ή αναπάντεχα λάθη κατά την πληκτρολόγηση, κάνοντας το πρόγραμμά μας πιο εύκολο στη συντήρηση και τη διόρθωση.

Παναγιώτης Λειβαδάρος Π21085 και Νικόλαος Παπανικολάου Π21130