3. Ουρές

3.1 Εισαγωγή στις Ουρές

Μια ουρά είναι μια "γραμμή αναμονής", όπως για παράδειγμα μια ουρά ανθρώπων, που περιμένουν στο ταμείο ενός σουπερ-μάρκετ, ή μια ουρά εργασιών σ' ένα σύστημα υπολογιστή, που περιμένουν για κάποια συσκευή εξόδου, π.χ. έναν εκτυπωτή. Σε καθένα από τα παραδείγματα αυτά, τα αντικείμενα εξυπηρετούνται με την σειρά με την οποία φτάνουν, δηλαδή το πρώτο στοιχείο της ουράς είναι το πρώτο που θα εξυπηρετηθεί. Ενώ, λοιπόν, μια στοίβα είναι μια "*τελευταίος μέσα - πρώτος έξω*" (Last In-First out, LIFO) δομή, η ουρά είναι μια "*πρώτος μέσα - πρώτος έξω*" (First In - First Out, **FIFO**) δομή.

Στην **ουρά** (**queue**) οι βασικές λειτουργίες εισαγωγής και διαγραφής πραγματοποιούνται στα δύο άκρα της. Αντίθετα με τις στοίβες, στις οποίες τα στοιχεία απωθούνται ή ωθούνται μόνο στο ένα άκρο της δομής δεδομένων, η διαγραφή ενός στοιχείου από την ουρά γίνεται στο ένα άκρο της, που ονομάζεται **εμπρός** (**front**) ή **κεφάλι** (**head**), και η εισαγωγή ενός νέου στοιχείου γίνεται στο άλλο άκρο της, που ονομάζεται **πίσω** (**rear**) ή **ουρά** (**tail**). Άλλες βασικές λειτουργίες είναι η δημιουργία μιας κενής ουράς και ο έλεγχος αν μια ουρά είναι κενή.

Γενικά, οι **βασικές λειτουργίες** που συνδέονται με την ουρά είναι:

- **Δημιουργία μιας κενής ουράς**
- **Έλεγχος αν μια ουρά είναι κενή**
- **Διαγραφή του στοιχείου που βρίσκεται στο εμπρός άκρο της ουράς**
- **Εισαγωγή νέου στοιχείου στο πίσω άκρο της ουράς**

Ένας τυπικός ορισμός για τον αφηρημένο τύπο δεδομένων Ουρά είναι ο ακόλουθος:

ΑΤΔ ΟΥΡΑ

Συλλογή στοιχείων δεδομένων:

Μια συλλογή στοιχείων δεδομένων σε γραμμική διάταξη με την ιδιότητα ότι τα στοιχεία μπορούν να διαγράφονται μόνο από το ένα άκρο της, που ονομάζεται ***εμπρός***, και να εισάγονται μόνο στο άλλο άκρο της, που ονομάζεται ***πίσω***.

Βασικές λειτουργίες:

Δημιουργία μιας κενής ουράς - CreateQ:

Λειτουργία:	Δημιουργεί μια κενή ουρά.
Επιστρέφει:	Μια κενή ουρά.

Έλεγχος αν μια ουρά είναι κενή - EmptyQ:

Δέχεται:	Μια ουρά.
Λειτουργία:	Ελέγχει αν η ουρά είναι κενή.
Επιστρέφει:	TRUE, αν η ουρά είναι κενή, FALSE διαφορετικά.

Διαγραφή του στοιχείου που βρίσκεται στο εμπρός άκρο της ουράς -RemoveQ:

Δέχεται:	Μια ουρά.
Λειτουργία:	Ανακτά και διαγράφει το στοιχείο που βρίσκεται στο εμπρός άκρο της ουράς.
Επιστρέφει:	Το στοιχείο στο εμπρός άκρο της ουράς και την τροποποιημένη ουρά.

Εισαγωγή νέου στοιχείου στο πίσω άκρο της ουράς -AddQ:

Δέχεται:	Μια ουρά και ένα στοιχείο δεδομένου.
Λειτουργία:	Εισάγει το στοιχείο στο πίσω άκρο της ουράς.
Επιστρέφει:	Την τροποποιημένη ουρά.

Η ουρά είναι προφανώς μια κατάλληλη δομή δεδομένων για την αποθήκευση στοιχείων τα οποία πρέπει να υποστούν επεξεργασία με τη σειρά με την οποία δημιουργήθηκαν. Για παράδειγμα, ας υποθέσουμε ότι ένα πρόγραμμα είναι φτιαγμένο να προσφέρει προβλήματα άσκησης-εξάσκησης για στοιχειώδη αριθμητική. Τα προβλήματα αυτά αφορούν την πρόσθεση τυχαίων ακεραίων αριθμών. Αν ο μαθητής απαντήσει σωστά, τότε δημιουργείται ένα άλλο πρόβλημα. Αν, όμως, δεν απαντήσει σωστά, τότε το πρόβλημα αποθηκεύεται προκειμένου να του ζητηθεί ξανά να το απαντήσει στο τέλος του μαθήματος. Όπως φαίνεται, τα προβλήματα που δεν απαντήθηκαν σωστά θα πρέπει να εμφανιστούν με την ίδια σειρά με την οποία εμφανίστηκαν αρχικά. Επομένως, μια ουρά είναι η κατάλληλη δομή δεδομένων για την αποθήκευση αυτών των προβλημάτων.

Έστω ότι έχει αναπτυχθεί ένα πακέτο ή μονάδα για τον ΑΤΔ Ουρά ώστε να ορίζει έναν τύπο δεδομένων QueueElementType, ο οποίος μπορεί να χρησιμοποιηθεί για τη δήλωση μιας ουράς WrongQ εγγραφών προβλημάτων της μορφής

```
typedef struct {
   int Addend1, Addend2;
} QueueElementType;
```

καθώς και συναρτήσεις CreateQ, AddQ και RemoveQ και μια boolean συνάρτηση EmptyQ για τις λειτουργίες της ουράς. Τότε προτάσεις σαν τις ακόλουθες μπορούν να χρησιμοποιηθούν για τη δημιουργία ενός προβλήματος και την πρόσθεσή του στην WrongQ, αν δεν απαντηθεί σωστά:

```
Problem.Addend1=RandomInt(0,NumberLimit);
Problem.Addend2=RandomInt(0,NumberLimit);
Correct=Ask(Problem,1);
if (!Correct)
   AddQ(&WrongQ,Problem);
```

Η RandomInt είναι μια συνάρτηση που δημιουργεί τυχαίους ακεραίους από ένα καθορισμένο εύρος ακεραίων αριθμών. Η συνάρτηση Ask εμφανίζει ένα πρόβλημα πρόσθεσης στο μαθητή, διαβάζει μια απάντηση και επιστρέφει την τιμή TRUE ήFALSE για την boolean παράμετρο Correct για να δηλώσει αν η απάντηση ήταν σωστή. Η παράμετρος 1 δηλώνει το πόσες φορές έχει ζητηθεί το πρόβλημα. Στην συνέχεια, μπορούν να χρησιμοποιηθούν οι ακόλουθες προτάσεις για την επανάληψη των προβλημάτων που δεν απαντήθηκαν σωστά:

```
while (!EmptyQ(WrongQ))
  {
    RemoveQ(&WrongQ,&Problem);
    Correct=Ask(Problem,2);
    if (!Correct)
      Wrong++;
  }
```

Οι ουρές χρησιμοποιούνται πολύ συχνά για να παραστήσουν γραμμές αναμονής που προκύπτουν στην λειτουργία των υπολογιστικών συστημάτων. Αυτές οι ουρές σχηματίζονται κάθε φορά που περισσότερες από μια διαδικασίες απαιτούν ένα συγκεκριμένο πόρο, όπως έναν εκτυπωτή, έναν οδηγό δίσκου ή την κεντρική μονάδα επεξεργασίας. Καθώς οι διαδικασίες ζητούν ένα συγκεκριμένο πόρο, τοποθετούνται σε μια ουρά και περιμένουν να εξυπηρετηθούν από αυτόν τον πόρο. Για παράδειγμα, πολλοί υπολογιστές μπορεί να μοιράζονται τον ίδιο εκτυπωτή, οπότε χρησιμοποιείται μια *ουρά εκτύπωσης* για να ρυθμίσει τις αιτήσεις εξόδου κατά έναν τρόπο εξυπηρέτησης "*πρώτος- έρχεται - πρώτος-εξυπηρετείται*" (*first-come-first-served*). Αν κάποια διαδικασία ζητάει τον εκτυπωτή και αυτός είναι ελεύθερος, τότε διατίθεται αμέσως γι' αυτήν την διαδικασία. Όσο αυτή η έξοδος εκτυπώνεται, άλλες διαδικασίες μπορεί να ζητήσουν τον εκτυπωτή, οπότε τοποθετούνται σε μια ουρά και περιμένουν τη σειρά τους. Όταν η έξοδος της τρέχουσας διαδικασίας τελειώσει, ο εκτυπωτής

ελευθερώνεται από αυτήν τη διαδικασία και διατίθεται για την διαδικασία που βρίσκεται πρώτη στην ουρά.