12. Άσκηση 12 – Διαχείριση Παραγγελιών (OrdersHandlingApp)[1]

Η άσκηση έχει στόχο την ανάπτυξη μιας εφαρμογής που θα επιτρέπει στον χρήστη να διαχειρίζεται παραγγελίες σε ένα εργοστάσιο που παράγει φιάλες εμφιάλωσης.

Η ενότητα 1 περιγράφει το αντικείμενο το οποίο θα πρέπει να ικανοποιεί η εφαρμογή.

Η ενότητα 2 δίνει τα βασικά δεδομένα που θα πρέπει να διατηρεί η εφαρμογή για κάθε παραγγελία.

Η ενότητα 3 δίνει τις βασικές λειτουργίες που θα πρέπει να υλοποιεί η εφαρμογή. Τις λειτουργίες αυτές θα παρέχει στον χρήστη της ως επιλογές μέσα από ένα μενού.

Η ενότητα 4 περιγράφει τις καταστάσεις στις οποίες μπορεί να βρίσκεται μια παραγγελία.

Η **ενότητα 5 αναφέρεται στη διαδικασία που θα πρέπει να ακολουθήσετε για την ανάπτυξη του προγράμματος** και σας προτείνει ένα ενδεικτικό πλάνο ανάπτυξης. Σας συνιστούμε να ακολουθήσετε τα βήματα αυτά, τα οποία σας οδηγούν στην **ανάπτυξη ενός αριθμού εκδόσεων για το πρόγραμμα σας**.

Η ενότητα 6 δίνει **οδηγίες για την ανάπτυξη των διαφόρων εκδόσεων** του προγράμματος και σας κατευθύνει σε μια ενδεικτική υλοποίηση. Ανατρέξτε στην ενότητα αυτή μόνο μετά από δική σας προσπάθεια για την υλοποίηση της κάθε έκδοσης του προγράμματος. Μετά την μελέτη της ενδεικτικής υλοποίησης **αξιοποιείστε τις καλές πρακτικές και στη δική σας έκδοση.** Προχωρήστε στην επόμενη έκδοση μόνο όταν έχετε βελτιώσει τον κώδικα σας αξιοποιώντας τις καλές πρακτικές της ενδεικτικής λύσης.

Αξιοποιήστε τον πίνακα περιεχομένων του pdf για να έχετε μια αφαιρετική εικόνα της δόμησης του κειμένου η οποία σας επιτρέπει να μετακινήστε εύκολα από ενότητα σε ενότητα.

12.1 Το αντικείμενο

Ένα εργοστάσιο παρασκευάζει μικρά και μεγάλα μπουκάλια εμφιάλωσης. Τα μικρά κοστίζουν 0.008€ και τα μεγάλα 0.02€. Η εταιρεία έχει μια πολιτική χορήγησης έκπτωσης που περιγράφεται παρακάτω:
α) Για παραγγελίες μεγαλύτερες από 200€ ή 3000 μπουκάλια, γίνεται έκπτωση 8%.
β) Για παραγγελίες μεγαλύτερες από 600€, γίνεται έκπτωση 20%.

Η ζητούμενη εφαρμογή θα υποστηρίζει τη διαχείριση των παραγγελιών από τη στιγμή που δίνεται η παραγγελία μέχρι και την ολοκλήρωση της.

[1] Η άσκηση βασίζεται σε ανάλογη άσκηση που αναπτύχθηκε στα πλαίσια του μαθήματος Εισαγωγή στον Προγραμματισμό κατά την ακαδημαϊκή χρονιά 2022-23 στο τμήμα Τμήμα Μηχανικών Η/Υ & Πληροφορικής.

12.2 Η Παραγγελία (order)

Θεωρήστε μια παραγγελία να αποτελείται, εκτός από τον αριθμό των μικρών και τον αριθμό των μεγάλων μπουκαλιών, από τα παρακάτω τουλάχιστο στοιχεία:

1) Ονοματεπώνυμο πελάτη,
2) Ημερομηνία που δόθηκε η παραγγελία,
3) Ημερομηνία που πρέπει να εκτελεστεί,
4) Αρχικό κόστος παραγγελίας (κόστος χωρίς την έκπτωση),
5) Έκπτωση που αντιστοιχεί στην παραγγελία (σύμφωνα με την πολιτική εκπτώσεων),
6) Τελικό κόστος παραγγελίας (όπου έχει υπολογιστεί και η έκπτωση), και
7) Ημερομηνία που τελικά εκτελέστηκε η παραγγελία.

12.3 Η λειτουργικότητα που υλοποιεί η εφαρμογή

Η εφαρμογή θα πρέπει να υποστηρίζει τις παρακάτω λειτουργίες που κρίνονται απαραίτητες για την διαχείριση των παραγγελιών. Τις λειτουργίες αυτές θα μπορεί να επιλέξει ο χρήστης της εφαρμογής μέσα από ένα μενού επιλογών. Οι βασικές λειτουργίες περιλαμβάνουν τις παρακάτω:

0 – Τερματισμός Προγράμματος

1 - Καταχώρηση παραγγελίας

2- Εμφάνιση παραγγελιών πελάτη

Α. Πρώτη ομάδα (εκκρεμείς παραγγελίες - pendingOrders):

3 - Εμφάνιση εκκρεμών παραγγελιών

4 - Αποθήκευση εκκρεμών παραγγελιών σε αρχείο

5 - Φόρτωση εκκρεμών παραγγελιών από αρχείο.

6 - Εκτέλεση παραγγελίας (υπολογισμός αρχικού κόστους παραγγελίας)

Β. Δεύτερη ομάδα (παραγγελίες προς εξόφληση - readyOrders).

7 - Εμφάνιση των προς εξόφληση παραγγελιών

8 - Εξόφληση παραγγελίας (υπολογισμός τελικού κόστους παραγγελίας)

Γ. Τρίτη ομάδα (Εκτελεσμένες παραγγελίες- closedOrders).

9 - Εμφάνιση εκτελεσμένων παραγγελιών

10 - Αποθήκευση των εκτελεσμένων παραγγελιών σε ξεχωριστό αρχείο

11 - Φόρτωση από αρχείο εκτελεσμένων παραγγελιών

Μετά την ολοκλήρωση των παραπάνω λειτουργιών μπορείτε να προσθέσετε όποιες άλλες εσείς κρίνετε χρήσιμες σε ένα τέτοιο πρόγραμμα. Αν προχωρήσετε προς την κατεύθυνση αυτή δημιουργήστε μια νέα έκδοση για το πρόγραμμα σας.

12.4 Η εξέλιξη μιας παραγγελίας

Το Σχήμα 1 δίνει τις καταστάσεις στις οποίες μπορεί να βρίσκεται μια παραγγελία.
Μια παραγγελία όταν καταχωρείται είναι «εκκρεμής» (**pendingOrder**).
Όταν εκτελεστεί είναι «προς εξόφληση» (**readyOrder**).
Μετά την εξόφληση είναι «κλειστή» (**closedOrder**).

Η επιλογή «Εκτέλεση Παραγγελίας» εκτελείται πάνω σε μια παραγγελία που είναι pending και την μετατρέπει σε ready.

Η επιλογή «Εξόφληση Παραγγελίας» εκτελείται πάνω σε μια παραγγελία που είναι ready και την μετατρέπει σε closed.

Σχήμα 1. Καταστάσεις παραγγελίας

12.5 Διαδικασία Ανάπτυξης

Υιοθετήστε την **αυξητική ανάπτυξη**. Αξιοποιήστε την **αφαιρετικότητα στις διεργασίες**. Και τις δύο βασικές έννοιες έχουμε παρουσιάσει και αξιοποιήσει σε όλες τις εργαστηριακές ασκήσεις μέχρι σήμερα.

Περιγράφουμε στη συνέχεια ένα σύνολο από **ενδεικτικά Βήματα** τα οποία σας συνιστούμε να ακολουθήσετε για να αναπτύξετε σταδιακά με βάση την αυξητική ανάπτυξη το ζητούμενο πρόγραμμα. Κάθε ένα από τα βήματα αυτά οδηγεί στην ανάπτυξη μιας έκδοσης του προγράμματος σας.

Δημιουργήστε έναν **φάκελο με όνομα OrdersHandling** και μέσα σε αυτόν τοποθετήστε τους φακέλους που αντιστοιχούν σε κάθε μία από τις εκδόσεις που θα αναπτύξετε. Μπορείτε να δώσετε στους φακέλους αυτούς ονόματα όπως V0, V1, V2, V3, …

Προχωρήστε στην επόμενη έκδοση (επόμενο βήμα) μόνο μετά την ολοκλήρωση μιας έκδοσης και τον έλεγχο καλής λειτουργίας της. Όπου **ολοκλήρωση μιας έκδοσης** στα πλαίσια αυτής της άσκησης **σημαίνει** πως μετά την συγγραφή του κώδικα της έκδοσης από μέρους σας έχετε μελετήσει τις οδηγίες της ενδεικτικής υλοποίησης για την αντίστοιχη έκδοση και έχετε βελτιώσει τον κώδικα σας ώστε να αξιοποιεί τις καλές πρακτικές.

12.5.1 Βήμα 0

Το έργο της ανάθεσης του υπολογισμού του κόστους μιας παραγγελίας στην απλή της μορφή όπου αυτή αποτελείται μόνο από τον αριθμό των μικρών και αριθμό μεγάλων μπουκαλιών το ολοκληρώσαμε στα πλαίσια της Δράσης [Δ1] της εργαστηριακής άσκησης της 8ης Εβδομάδας.

Η πρώτη ενέργεια που έπρεπε να κάνουμε ήταν να δώσουμε τη λεκτική περιγραφή για την main του ζητούμενου προγράμματος. Ενδεικτική λεκτική περιγραφή που αξιοποιεί την αφαιρετικότητα στις διεργασίες σας δόθηκε. (I2P_LabExercise6And8.pdf Διαφάνειες 15-18)

Προχωρήστε στη δόμηση του κώδικα σας με βάση την παραπάνω λεκτική περιγραφή. Πιθανότατα το έχετε κάνει ήδη. Μόνο μετά την επιτυχή ολοκλήρωση του βήματος αυτού προχωράτε στο επόμενο που οδηγεί στην 1η έκδοση του προγράμματος OrdersHandling.

12.5.2 Βήμα 1 – 1η έκδοση (V1)

Ξεκινήστε ένα νέο project χωρίς να αξιοποιήσετε στην φάση αυτή τον κώδικα του βήματος 0.

Εφαρμόζουμε top-down προσέγγιση σύμφωνα με την οποία **θα γράψουμε πρώτα τη main().** Θα την γράψουμε και θα ελέγξουμε την λειτουργικότητα της. Αυτό μπορεί να γίνει χωρίς να γράψουμε τα σώματα των συναρτήσεων που αυτή καλεί. Προσέξτε η main() δεν θα κάνει άλλη δουλειά από το να καλεί δικές σας συναρτήσεις. Ορίστε μια συνάρτηση για κάθε μια από τις λειτουργικότητες που το πρόγραμμα σας πρέπει να υλοποιεί. Την συνάρτηση αυτή θα καλεί η main όταν ο χρήστης επιλέξει την αντίστοιχη επιλογή του μενού.

Αν έχετε αποδώσει **κατάλληλα ονόματα στις συναρτήσεις** σας τότε διαβάζοντας κάποιος τρίτος τον κώδικα σας θα καταλάβει τη κάνει η εφαρμογή σας σε απόκριση των επιλογών του χρήστη. Το πως η κάθε συνάρτηση θα κάνει την δουλειά της θα το ορίσετε σε επόμενο βήμα όταν γράψετε το σώμα της.

Ορίστε την δομή της main() με την παράλληλη ανάπτυξη του μενού επιλογής λειτουργίας, όπως ακριβώς κάνατε στην Άσκηση-Ταξινόμηση Λέξεων ([Δ2]- Εργαστήριο 6ης εβδομάδας (Αντικείμενο)).

Η έκδοση αυτή υλοποιεί μόνο την επιλογή Τερματισμού προγράμματος. Στη φάση αυτή δεν είναι απαραίτητο να περιλάβετε όλες τις επιλογές του μενού. Βάλτε 4-5 και προσθέτετε τις υπόλοιπες αργότερα καθώς θα υλοποιείτε την αντίστοιχη λειτουργικότητα.

Ανατρέξτε στις οδηγίες της ενδεικτικής υλοποίησης για να ελέγξετε αν δομήσατε σωστά την main(). Θα προχωρήσετε στην επόμενη έκδοση μόνο μετά τον έλεγχο αυτό.

Στην έκδοση αυτή μπορείτε προαιρετικά να ορίσετε, για όποια συνάρτηση είστε σε θέση να το κάνετε, τα παρακάτω:

a) τα ορίσματα που πρέπει να της περάσουμε για να κάνει τη δουλειά που της έχουμε αναθέσει, για παράδειγμα η displayOrder() πρέπει να πάρει ως όρισμα μια μεταβλητή τύπου order, που είναι αυτή που θα εμφανίσει, και,
b) την επιστρεφόμενη τιμή αν πρέπει να επιστρέψει στην main κάτι που έχει σχέση με τη διεργασία που αυτή εκτελεί, για παράδειγμα η getOrder() πρέπει να επιστρέψει την παραγγελία που πήρε από την βασική είσοδο.

12.5.3 Βήμα 2 – 2η έκδοση (V2)

Η έκδοση αυτή (2η) θα υλοποιεί τις λειτουργίες 1, 2 και 3 του μενού επιλογής. Θα πρέπει να επιλέξετε να δουλέψετε μεταξύ των δυο επιλογών:

1. πίνακα παραγγελιών, ή,
2. πίνακα δεικτών σε παραγγελίες.

Αν δεν είστε εξοικειωμένοι με τους δείκτες και τη δυναμική διαχείριση μνήμης (malloc και free) επιλέξτε τον πίνακα παραγγελιών. Και στις 2 περιπτώσεις υποθέστε ένα σταθερό μέγεθος για τον πίνακα σας, υιοθετήστε δηλαδή **στατική δέσμευση χώρου** για αποθήκευση των δεδομένων της εφαρμογής. Στην φάση αυτή δεν μας απασχολεί η βέλτιστη χρήση της μνήμης. Ο πρωταρχικός μας στόχος είναι η υλοποίηση της λειτουργικότητας της εφαρμογής.

Για να διευκολύνετε τη διαδικασία ανάπτυξης του προγράμματος **αρχικοποιήστε τον πίνακα των παραγγελιών σας με ορισμένες παραγγελίες**. Θα αποφύγετε με τον τρόπο αυτό να εισάγετε παραγγελίες κάθε φορά που θα ελέγχετε το πρόγραμμα σας. Φροντίστε να βάλετε ως αρχικές τιμές στον πίνακα, παραγγελίες που θα αναγκάζουν την calculateDiscount() να διατρέξει όλες τις εναλλακτικές διαδρομές (ροές) εκτέλεσης για να βεβαιωθείτε για την σωστή λειτουργία της.

Η υλοποίηση της λειτουργίας Εμφάνισης εκκρεμών παραγγελιών είναι απλή στην φάση αυτή καθώς όλες οι παραγγελίες μας είναι ακόμη εκκρεμείς (pending). Προχωρήστε στην υλοποίηση της. Θα την τροποποιήσετε αργότερα όταν θα έχετε και παραγγελίες άλλης κατάστασης, π.χ. readyOrders.

Οδηγίες για την υλοποίηση της έκδοσης αυτής θα βρείτε στο τέλος της άσκησης (ενότητα Οδηγίες Ενδεικτικής Υλοποίησης). Μελετήστε τις μόνο μετά τη δική σας προσπάθεια για τη δημιουργία της έκδοσης αυτής. Βελτίωστε τον κώδικα σας λαμβάνοντας υπόψιν τις οδηγίες της ενδεικτικής υλοποίησης.

Ο έλεγχος της σωστής εκτέλεσης των λειτουργιών 1, 2 και 3 σηματοδοτεί και την ολοκλήρωση της 2ης έκδοσης του προγράμματος σας.

12.5.4 Βήμα 3 – 3η έκδοση (V3)

Επιλέγουμε για την έκδοση αυτή τις λειτουργίες 6, 7 και 8. Αφήνουμε τις λειτουργίες που έχουν σχέση με τα αρχεία για επόμενη έκδοση.

Εκτέλεση παραγγελίας

Η λειτουργία αυτή περιλαμβάνει τον υπολογισμό του αρχικού κόστους της παραγγελίας, δηλαδή του κόστους χωρίς την έκπτωση. Η λειτουργία εκτελείται πάνω σε μία εκκρεμή (pending) παραγγελία και την μετατρέπει σε ready.

Στην φάση αυτή θα πρέπει να επιλέξετε αν θα διατηρείτε τις ready παραγγελίες στον ίδιο πίνακα με τις pending (orders) ή θα τις αποθηκεύετε σε διαφορετικό πίνακα. Στις οδηγίες περιγράφονται και οι δύο επιλογές υλοποίησης.

Η πρώτη επιλογή είναι πιο απλή από την δεύτερη. Και για τις δύο επιλογές υιοθετούμε **το μοντέλο εξυπηρέτησης first come first served**. Αυτό σημαίνει πως η διαχείριση των παραγγελιών (Εκτέλεση Παραγγελίας και Εξόφληση Παραγγελίας) γίνεται σειριακά με την σειρά που αυτές έχουν καταχωρηθεί. Επομένως η λειτουργία «Εκτέλεση παραγγελίας» επιδρά πάνω στην πιο παλιά εκκρεμή παραγγελία.

Εξόφληση παραγγελίας

Η επιλογή «Εξόφληση Παραγγελίας» εκτελείται πάνω σε μια παραγγελία που είναι ready και την μετατρέπει σε closed. Περιλαμβάνει τον υπολογισμό του τελικού κόστους της παραγγελίας με βάση την πολιτική έκπτωσης.

Αν έχετε επιλέξει για την υλοποίηση της επιλογής Εκτέλεσης παραγγελίας, ένα ξεχωριστό πίνακα για τις ready παραγγελίες θα ορίσετε τώρα έναν νέο πίνακα για τις closed παραγγελίες. Διαφορετικά κα οι closed παραγγελίες θα βρίσκονται στον αρχικό πίνακα σας.

12.5.5 Βήμα 4 – 4η έκδοση (V4)

Η έκδοση υλοποιεί τις λειτουργίες που έχουν σχέση με αποθήκευση και ανάκληση πληροφορίας από αρχεία.

Υλοποιήστε πρώτα την λειτουργία αποθήκευσης (επιλογή 4) και στη συνέχεια την επιλογή 5 που υλοποιεί την ανάκληση πληροφορίας.

Η **λειτουργία της αποθήκευσης** έχει ως στόχο να αποθηκεύσει τον πίνακα ή τους πίνακες παραγγελιών σε αρχείο ή αρχεία ώστε αυτές να μην χαθούν με τον τερματισμό του προγράμματος.

Έτσι όταν ανοίξουμε πάλι το πρόγραμμα μπορούμε με την **λειτουργία της φόρτωσης** του ή των αρχείων να επαναφέρουμε το πρόγραμμα μας στην κατάσταση στην οποία ήταν πριν την αποθήκευση και το κλείσιμο της εφαρμογής.

Για την αποθήκευση της παραγγελιών σε αρχείο και για την φόρτωση από αρχείο (επιλογές 5,6,9,10) μπορείτε να χρησιμοποιήσετε τις παρακάτω συναρτήσεις της βασικής βιβλιοθήκης της C:

- `fopen()`
- `fprintf()`
- `fscanf()`
- `fclose()`

Μια αναφορά στις συναρτήσεις αυτές και τον τρόπο χρήσης τους μπορείτε να βρείτε στην ενότητα 6.

12.6 Οδηγίες Ενδεικτικής Υλοποίησης

Ανατρέξτε στην ενότητα αυτή μόνο μετά από τη δική σας προσπάθεια για την εκτέλεση του κάθε βήματος. Μελετήστε το υλικό της ενότητας ακόμη και αν έχετε ολοκληρώσει επιτυχώς το αντίστοιχο βήμα. Στην χειρότερη περίπτωση θα δείτε μία εναλλακτική υλοποίηση, που πάντα έχει κάτι να σας προσφέρει.

12.6.1 Βήμα 1 – 1η έκδοση

Η διαδικασία ανάπτυξης της κύριας ροής του προγράμματος (που είναι και η κύρια ροή της main) περιγράφεται αναλυτικά στο αρχείο KTBook-Exercise4.pdf, και πιο συγκεκριμένα στο Βήμα 1.

12.6.2 Βήμα 2 – 2η έκδοση

Ορίστε τη δομή (struct) που αναπαριστά την παραγγελία και στην συνέχεια τον πίνακα παραγγελιών.

Καταχώρηση Παραγγελίας

Υλοποιήστε την συνάρτηση που θα διαβάζει από την κύρια είσοδο μία παραγγελία. Πιθανότατα θα την ονομάσετε **getOrder**. Είναι αρμοδιότητα μιας άλλης συνάρτησης να βάλει την παραγγελία στον πίνακα παραγγελιών. Δώστε της κατάλληλο όνομα. Αυτή θα τοποθετεί την παραγγελία στην επόμενη ελεύθερη θέση του πίνακα.

Για να απλοποιήσουμε την υλοποίηση θα θεωρήσουμε πως οι παραγγελίες καταχωρούνται στον πίνακα η μία μετά την άλλη. Μια μεταβλητή με όνομα **lastOrder** μπορεί να περιέχει την τάξη στον πίνακα της τελευταίας παραγγελίας που έχει καταχωρηθεί. Φροντίστε να την κρατάτε ενήμερη κατά τη διάρκεια λειτουργίας του προγράμματος σας.

Εκκρεμείς Παραγγελίες

Υλοποιήστε την συνάρτηση που θα εμφανίζει στην κύρια έξοδο όλες τις εκκρεμείς παραγγελίες. Στην φάση αυτή όλες οι παραγγελίες που είναι καταχωρημένες στον πίνακα παραγγελιών είναι εκκρεμείς. Άρα η επιλογή αυτή εμφανίζει στην οθόνη όλες τις παραγγελίες του πίνακα. Το Σχήμα 2 εμφανίζει ένα στιγμιότυπο από μια ενδεικτική υλοποίηση η οποία περιέχει ορισμένα μόνο στοιχεία της παραγγελίας (ονοματεπώνυμο πελάτη, ποσότητα μικρών φιαλών, ποσότητα μεγάλων φιαλών, αρχικό και τελικό κόστος). Οι παραγγελίες είναι εκκρεμείς οπότε το αρχικό κόστος καθώς και το τελικό δεν έχουν υπολογιστεί ακόμη.

Παραγγελίες Πελάτη

Υλοποιήστε τη συνάρτηση που θα εμφανίζει στην κύρια έξοδο τις παραγγελίες ενός συγκεκριμένου πελάτη. Το όνομα του πελάτη θα δίνει ο χρήστης από την κύρια είσοδο. Αν δεν υπάρχουν παραγγελίες για τον πελάτη να εμφανίζεται σχετικό μήνυμα.

Ο έλεγχος της καλής λειτουργίας των παραπάνω συναρτήσεων σηματοδοτεί και την ολοκλήρωση της 2ης έκδοσης του προγράμματος σας.

```
0-telos programmatos
1-add order
2-display client's orders
3-display pending orders
4-save pending orders
5-load pending orders
6-execute order

Dose epilogi:3

Pending Orders
OrderNo 1 : Nikos          200          300          0.000000       0.000000
OrderNo 2 : Kostas        1000         3000          0.000000       0.000000
OrderNo 3 : panos         2200         3100          0.000000       0.000000
```

Σχήμα 2. Εμφάνιση εκκρεμών παραγγελιών

12.6.3 Βήμα 3 – 3η έκδοση

Περιγράφονται στη συνέχεια 2 επιλογές υλοποίησης της εφαρμογής :

1. Χρήση ενός πίνακα για όλες τις παραγγελίες
2. Χρήση 3 διαφορετικών πινάκων για τις παραγγελίες

Η πρώτη επιλογή είναι πιο απλή από την δεύτερη. Και για τις δύο επιλογές υιοθετούμε το **μοντέλο εξυπηρέτησης first come first served**. Αυτό σημαίνει πως η διαχείριση των παραγγελιών (Εκτέλεση Παραγγελίας και Εξόφληση Παραγγελίας) γίνεται σειριακά με την σειρά που αυτές καταχωρήθηκαν.

Προαιρετικά αξιοποιήστε την υποστήριξη που παρέχει η βασική βιβλιοθήκη για τη διαχείριση ημερομηνίας και ώρας (w3resource).

12.6.3.1 Χρήση ενός πίνακα για όλες τις παραγγελίες

Με βάση την επιλογή αυτή έχουμε ένα πίνακα στον οποίο καταχωρούνται οι παραγγελίες και παραμένουν εκεί και μετά την ολοκλήρωση τους.

Το σχήμα 3 δίνει μια εικόνα του πίνακα orders κατά τη διάρκεια εκτέλεσης της εφαρμογής. Υπάρχουν 3 ακέραιες μεταβλητές:

Α. Η **lastOrder** έχει τιμή 5 που είναι η τάξη του στοιχείου του πίνακα της τελευταίας παραγγελίας που καταχωρήθηκε.

Β. Η **lastHandled** έχει τιμή 3 που είναι η τάξη του στοιχείου του πίνακα της τελευταίας παραγγελίας που εκτελέστηκε.

Γ. Η **lastClosed** έχει τιμή 1 που είναι η τάξη του στοιχείου του πίνακα της τελευταίας παραγγελίας που ολοκληρώθηκε.

Σχήμα 3. Εικόνα του πίνακα παραγγελιών κατά την διάρκεια εκτέλεσης της εφαρμογής.

Με βάση τις τρείς αυτές μεταβλητές μπορείτε να υλοποιήσετε τις λειτουργίες καταχώρηση (επιλογή 1), εκτέλεση (επιλογή 6) και εξόφληση (επιλογή 8) και τις επιλογές εμφάνισης παραγγελιών.

12.6.3.2 Χρήση τριών διαφορετικών πινάκων για τις παραγγελίες

Με βάση την επιλογή αυτή έχουμε τρείς πίνακες, ένα για κάθε κατάσταση παραγγελιών (εκκρεμείς παραγγελίες, παραγγελίες εκτελεσμένες, δηλαδή, έτοιμες προς εξόφληση, παραγγελίες ολοκληρωμένες/κλειστές). Το σχήμα 4 δείχνει τους 3 αυτούς πίνακες.

Η επιλογή «Εκτέλεση Παραγγελίας» (executeOrder) ενεργεί πάνω στον πίνακα εκκρεμών παραγγελιών (pendingOrders[]) και εκτελεί μία παραγγελία με βάση την πολιτική first come first served.

Η επιλογή «Εξόφληση Παραγγελίας» (closeOrder) ενεργεί πάνω στον πίνακα παραγγελιών προς εξόφληση (readyOrders[]) και εξοφλεί μία παραγγελία με βάση την πολιτική first come first served.

pendingOrders	readyOrders	closedOrders
Παραγγελία No1	Παραγγελία No1	Παραγγελία No1
Παραγγελία No2	Παραγγελία No2	Παραγγελία No2
Παραγγελία No3	Παραγγελία No3	
Παραγγελία No4	Παραγγελία No4	
Παραγγελία No5		
Παραγγελία No6		

Σχήμα 4. Εικόνα των τριών πινάκων με τις παραγγελίες όπως εξελίσσονται.

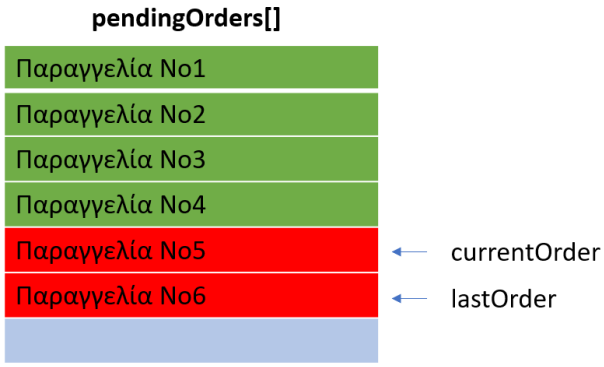
Για τους 2 πρώτους πίνακες θέλετε 2 ακέραιες μεταβλητές που θα προσδιορίζουν:
A) την τάξη στον πίνακα της προς εξυπηρέτηση παραγγελίας (currentOrder).
B) την τάξη της τελευταία εισαχθείσας στον πίνακα παραγγελίας (lastOrder).

Για τον τρίτο πίνακα θέλετε μόνο μια ακέραια μεταβλητή που προσδιορίζει την τάξη της τελευταία εισαχθείσας στον πίνακα παραγγελίας (lastOrder).

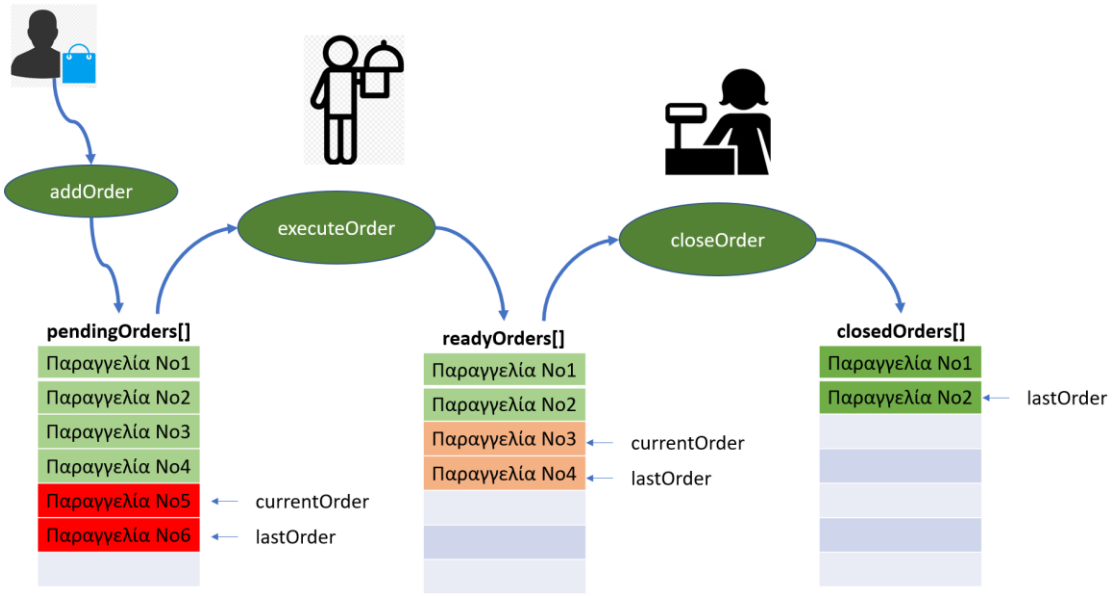
Το σχήμα 5 δίνει μια εικόνα του πίνακα των εκκρεμών παραγγελιών με τις δύο μεταβλητές. Βλέπουμε πως έχουν καταχωρηθεί 6 παραγγελίες και έχουν εκτελεστεί οι 4 πρώτες. Η currentOrder εχει την τιμή 4 και η lastOrder την τιμή 5.

Στο σχήμα 6 δίνεται μια εικόνα των πινάκων της εφαρμογής με τις μεταβλητές κατά την διάρκεια εκτέλεσης της εφαρμογής.

Την lastOrder του πίνακα pendingOrders θα χρησιμοποιήσει η συνάρτηση που υλοποιεί την επιλογή καταχώρηση παραγγελία (addOrder) για να καταχωρήσει τη νέα παραγγελία στην επόμενη κενή θέση του πίνακα. Θα πρέπει να αυξήσει την lastOrder κατά 1 πριν την καταχώρηση.



Σχήμα 5. Εικόνα του πίνακα των εκκρεμών παραγγελιών με τις δύο μεταβλητές.



Σχήμα 6. Εικόνα των πινάκων της εφαρμογής «Διαχείριση Παραγγελιών».

Την currentOrder του πίνακα pendingOrders θα χρησιμοποιήσει η συνάρτηση που υλοποιεί την επιλογή Εκτέλεση παραγγελίας (executeOrder) για να εκτελέσει την παραγγελία No5, την οποία θα μεταφέρει στον πίνακα παραγγελιών προς εξόφληση υπολογίζοντας και το κόστος της. Θα αυξήσει την currentOrder κατά 1 και θα χρησιμοποιήσει την μεταβλητή lastOrder του πίνακα readyOrders την οποία και θα αυξήσει κατά 1 για να τοποθετήσει την εκτελεσθείσα παραγγελία.

Για απλοποίησης της υλοποίησης επιλέξαμε να αφήνουμε τις παραγγελίες τους πίνακες. Μια επιλογή βελτίωσης προς την κατεύθυνση αυτή είναι η υλοποίηση των πινάκων ως **κυκλικών buffers**. Στην περίπτωση αυτή η μεταβλητή lastOrder όταν φτάσει στο τέλος του πίνακα λαμβάνει ως τιμή την τάξη του πρώτου στοιχείου με την προϋπόθεση βέβαια πως η currentOrder δεν δείχνει εκεί. Αν δείχνει, σημαίνει πως ο buffer έχει γεμίσει και δεν υπάρχει δυνατότητα καταχώρησης παραγγελίας έως ότου μία παραγγελία αλλάξει κατάσταση.

12.6.4 Βήμα 4 – 4η έκδοση

12.6.4.1 Η συνάρτηση fopen ()

```
FILE *fopen( const char * filePath, const char * mode );
```

Η fopen () χρησιμοποιείται για να ανοίξει ένα αρχείο και να δημιουργήσει ένα κανάλι επικοινωνίας του προγράμματος με το αρχείο. Αν το αρχείο δεν υπάρχει μπορεί να το δημιουργήσει αν την καλέσετε περνώντας στην παράμετρο mode την αλφαριθμητική σταθερά "w". Η σταθερά αυτή έχει ως αποτέλεσμα να ανοίγει η fopen το αρχείο για εγγραφή (writing) και αν αυτό δεν υπάρχει να το δημιουργεί. Τις τιμές της mode και την σημασία τους μπορείτε να βρείτε στον παρακάτω σύνδεσμο όπου θα βρείτε και παράδειγμα χρήσης της.
https://www.w3schools.in/c-programming/file-handling/fopen
Η συνάρτηση επιστρέφει δείκτη σε δομή FILE τον οποίο θα χρησιμοποιήσετε στη συνέχεια για την εγγραφή στο αρχείο ή την ανάγνωση από αυτό. Στην φάση αυτή μην ασχοληθείτε με τον τύπο FILE. Απλά ορίστε μια μεταβλητή αυτού του τύπου για να της αποδώσετε την τιμή που επιστρέφει η fopen(). Την μεταβλητή αυτή θα περάσετε ως όρισμα στην συνέχεια στις συναρτήσεις που σας δίνουν τη δυνατότητα να γράψετε (fprintf ()) και να διαβάσετε (fscanf ()) από το συγκεκριμένο αρχείο.

12.6.4.2 Η συνάρτηση fprintf ()

```
int fprintf(FILE *stream, const char *format, ...)
```

Η fprintf () μας δίνει την δυνατότητα να γράψουμε (αποθηκεύσουμε) δεδομένα στο αρχείο που προσδιορίζεται από το πρώτο όρισμα της συνάρτησης.
Η συνάρτηση είναι παρόμοια με την printf με τη διαφορά ότι τα ορίσματα της δεν προωθούνται στην βασική έξοδο (οθόνη) αλλά στο αρχείο που προσδιορίζεται από το πρώτο όρισμα, το οποίο είναι η τιμή που επέστρεψε η fopen().
Οδηγίες χρήσης της συνάρτησης και παράδειγμα κλήσης θα βρείτε στον παρακάτω σύνδεσμο.
https://www.w3schools.in/c-programming/file-handling/fprintf

12.6.4.3 Η συνάρτηση fscanf ()

Η fscanf () χρησιμοποιείται για ανάγνωση δεδομένων από το αρχείο που προσδιορίζεται από το πρώτο όρισμα της συνάρτησης.
Η συνάρτηση είναι παρόμοια με την scanf με την διαφορά ότι διαβάζει την είσοδο όχι από την βασική είσοδο (πληκτρολόγιο) αλλά από το αρχείο που προσδιορίζεται από το πρώτο όρισμα, το οποίο είναι η τιμή που επέστρεψε η fopen().

```
int fscanf(FILE *stream, const char *format, ...)
```

Οδηγίες χρήσης της συνάρτησης και παράδειγμα κλήσης θα βρείτε στον παρακάτω σύνδεσμο.
https://www.w3schools.in/c-programming/file-handling/fscanf

12.6.4.4 Η συνάρτηση fclose ()

```
int fclose( FILE * stream );
```

Η fclose () χρησιμοποιείται για να κλείσει το αρχείο και να ελευθερώσει τους πόρους που δεσμεύτηκαν για τη δημιουργία του καναλιού επικοινωνίας με το αρχείο ελευθερώνοντας τους πόρους που δεσμεύθηκαν για τη δημιουργία του.

Οδηγίες χρήσης της συνάρτησης και παράδειγμα κλήσης θα βρείτε στον παρακάτω σύνδεσμο.
https://www.w3schools.in/c-programming/file-handling/fclose

Δεν θα πρέπει να λησμονείτε να κλείνετε το αρχείο όταν τελειώνετε με την εγγραφή ή/και ανάγνωση σε/από αυτό.