OOP24-25 / Γεώργιος Ζαΐμης – 1115201900058

Το παρόν αποτελεί επεξήγηση και αιτιολόγηση σχεδιαστικών επιλογών που πάρθηκαν κατά τη διάρκεια της εργασίας. Θα γίνει αυτή σε format class based προκειμένου με μια μικρή αναφορά και στο παραγόμενο αρχείο.

Παραδοχές:

1)Η εργασία έχει ορισμένες ασυνέπειες στην υλοποίηση της. Αφενός γιατί δεν έγινε από πλευράς μου ουσιαστική κατανόηση του τι απαιτούσε το κομμάτι των unit_tests εξαρχής, με αποτέλεσμα να κάνω αλλαγές τελευταία στιγμή σε μια εργασία που δεν έχει δομηθεί για διαφορετικές πηγές input παρά μόνο για user input (cin). Αφετέρου γιατί επί της προσαρμογής σε αυτή τη παράμετρο, δεν υπήρξε χρόνος για συνεπής υλοποίηση. Βάση αυτού:

-Η κλάση e-shop που από τη περιγραφή της λαμβάνει τον ρόλο του interface του e-shop και της ανακατεύθυνσης του input είτε με cin είτε από το εκάστοτε αρχείο. Ωστόσο αυτό δεν ισχύει απόλυτα, μιας και που η κλάση admin διατηρεί στοιχεία interface και διαχείρισης input. Αυτό γιατί αντίστοιχα, μέχρι σχεδόν το τέλος της εργασίας δεν είχα κατά νου αυτή την παράμετρο οπότε τα στοιχεία του menu ήταν διαμοιρασμένα στο αρχείο oop24 στις συναρτήσεις της admin και στης customer (της οποίας κατάφερα να τα μεταφέρω στην e – shop και δεν εξαρτάται πλέον από fstream/stream). Εκ' του αποτελέσματος πρόκειται για πολύ πιο ορθή και ολοκληρωμένη λογική σε σχέση με τη πρώιμη ιδέα, αν και ανελλιπώς υλοποιημένη.

- Για τον ίδιο λόγο που προανέφερα, αν και στην εργασία γενικά επιλέγω για διάφορες χρήσεις να χρησιμοποιώ vectors (αντί για map π.χ.) για να διατηρήσω την υλοποίηση όσο πιο απλοική γίνεται, πόσο μάλλον αφού το πρόγραμμα δοκιμάζεται σε μικρές σχετικά ποσότητες αντικειμένων όπου και η πολυπλοκότητα O(1) είναι προτιμότερη, στο κομμάτι του categories.txt, το οποίο και είχα αγνοήσει (οι κατηγορίες υλοποιούντουσαν ως απλά string χωρίς συσχέτιση με το αρχείο), έγινε με map προκειμένου να προλάβω να πετύχω τη σύνδεση categories – subcategories.

Κλάσεις *(γενική περιγραφή)*

Η βασική δομή της εργασίας είναι η εξής: υπάρχουν 6 κλάσεις εκ των οποίων η User αποτελεί Base class για τις Admin και Customer προκειμένου να υπάρξει μια καλύτερη σύνδεση με τον πραγματικό κόσμο (πιο εύκολη κατανόηση της υλοποίησης) και για να γίνεται πιο απλοϊκά η αποθήκευση των χρηστών σε ενιαία δομή. Η κλάση product, όπως εύκολα καταλαβαίνει κανείς, αποτελεί την αναπαράσταση των προϊόντων και συνεπώς οι συναρτήσεις της έχουν ως κέντρο, την αλλαγή τη προβολή και τη μεταφορά πληροφοριών που χρειάζονται για τα αντικείμενά της, σε άλλες κλάσεις. Η κλάση UserManagment συσσωρεύει τις συναρτήσεις

που χρειάζονται προκειμένου να γίνεται σωστή φόρτωση και αποθήκευση των αρχείων. Τέλος, η e-shop αποτελεί τη κλάση υπεύθυνη για τη διαχείριση του input stream και της δομής του interface/menu.

User (Base class)

Εξαρχής πάρθηκε η αποφαση να δημιουργηθεί base class από την οποία κάθε χρήστης που αποκτά ρόλο θα κληρονομεί βασικά χαρακτηριστικά. Βάση της φύσης της εργασίας, η βασική κλάση δεν έχει λόγο να έχει μέλη και βάση αυτού είναι καλύτερο να διατηρηθεί ως Abstract class. Θέτοντας τα στοιχεία username και password protected, θεωρώ ότι έκανε τον κώδικα πιο ευέλικτο και επεκτασιμο ενω διατηρει στοιχεια πολυμορφικότητας μιας και υιοθετούνται αυτά τα στοιχεία από τις υποκλασεις Admin/Customer. Χρησιμοποιούμε κυρίαρχα αναφορές στα ορίσματα των συναρτήσεων για να αποφύγουμε την διαδικασία της αντιγραφής και όπου μας επιτρέπει η συγκυρία χρησιμοποιούμε const για διατήρηση καθαρότερου κώδικα και αποφυγή λαθών.

bool validatePassword: Χρησιμοποιείται κατά βάση για την δυνατότητα Login. Συγκρίνει το password που αντιστοιχεί σε οποιοδήποτε αντικείμενο Customer/Admin με το input που παρέχεται και επιστρέφει ανάλογη τιμή 1/0.

Void displayRole: Χρησιμοποιήθηκε για λόγους debugging.

Admin

Η κλάση admin αντιπροσωπεύει τους χρήστες με προνόμιο την αλλαγή/προσθήκη/αφαίρεση προιόντων από το "menu" του e-shop. Κληρονομεί τα στοιχεία username και password από την user. Ο constructor της καλείται όταν το Boolean isAdmin κατά τη διαδικασία του register παίρνει τιμή true και αποθηκεύεται στο users.txt . Διατηρεί στοιχεία interface/menu και διαχείρισης input stream (βλέπε παραδοχές). Οποιαδήποτε συνάρτηση της διατηρεί στοιχεία menu παίρνει ως όρισμα την αναφορά στη πηγή input.

Void searchProducts: Αποτελεί τη μόνη συνάρτηση που διατηρεί τη προσέγγιση που είχαν πολλές συναρτήσεις πριν προσαρμοστούν στις απαιτήσεις των unit_tests. Η υλοποίηση είναι απλή ως λογική: Ζητάει title, category, subcategory, ο χρήστης έχει την επιλογή να αφήσει κενό σε οποιοδήποτε από τα τρία θέλει πατώντας Enter. Ελέγχει στη συνέχεια αν οποιαδήποτε από αυτά είναι κενό μέσω του || όπου και επιστρέφει true αν ειτε βρέθηκε στο vector προιον με τέτοια αντιστοιχία είτε είναι κενό σαν κριτήριο αναζήτησης. Στη συνέχεια εκτυπώνει οτιδήποτε ταιριάζει σε όλα τα κριτήρια μέσω του ελέγχου του &&. *Η μόνη διαφορά με την συνάρτηση αναζήτησης της customer είναι ότι εδώ εκτυπώνουμε και το πόσο δημοφιλές είναι αυτό το προϊόν.*

Void AddProduct: φορτώνει τις κατηγορίες των προϊόντων από το αρχείο categories.txt σε map για να επιτευχθεί η σύνδεση category - subcategory και διαχειρίζεται το format της σύγκρισης τους προκειμένου να αντιμετωπιστούν σφάλματα παραπανίσιων space. Ζητάει όλα τα στοιχεία με την δομή που οφείλουν να έχουν και στη συνέχεια με τη λειτουργία emplace.back το προσθέτει στο vector products. Τέλος το

αποθηκεύει στο εκάστοτε αρχείο (στη παραδοτέα μορφή του κώδικα στο products_copy για να μην γίνονται αλλαγές στα βασικά αρχεία).

Void editProduct: Δίνοντας τον τίτλο του προϊόντος που θέλει ο εκάστοτε admin να αλλάξει εμφανίζονται οι επιλογές επί των οποίων μπορεί να πραγματοποιήσει αλλαγές (τίτλος, περιγραφή, κατηγορία και υποκατηγορία, τιμή, διαθέσιμο απόθεμα ή ακύρωση). Η μόνη διαφορά μεταξύ αυτών είναι ότι οι κατηγορίες εκτυπώνονται κατευθείαν από το αρχείο categories.txt προκειμένου εάν προστεθούν κι άλλες στη συνέχεια να είναι άμεση η προσαρμογή της συνάρτησης, διατηρώντας ένα κάποιο scalability.

Void displayOutofStock: Ελέγχει όλο το vector products προκειμένου να εκτυπώσει οποιοδήποτε προϊόν είναι εκτός αποθέματος.

Void displayPopularProducts: Ζητάει input από τον χρήστη για τον αριθμό που θέλει να δει (πχ top 3, top 5, top 10). Σε προσωρινό vector ταξινομεί όλα τα προϊόντα με βάση το unitsSold μια μεταβλητή ενός product που αυξάνεται οποιαδήποτε φορά ένα προϊόν είναι σε καλάθι που πάει για checkout.

Customer

Η κλάση customer είναι η δεύτερη κλάση που υιοθετεί στοιχεία από τη User. Δημιουργείται αντικείμενο της όταν κατά το registration το isAdmin παρει τιμη false και αποθηκεύεται στο users.txt μαζί με τους admins. Επιπρόσθετο στοιχείο της κλάσης Customer σε σχέση με τις προηγούμενες είναι ένα ιδιωτικό vector που αναπαριστά το ενεργό καλάθι του. Η επιλογή υλοποίησης του cart σε vector μας δίνει αρκετά οφέλη με πιο σημαντικό τη διατήρηση της απλότητας τόσο υπολογιστικά όσο και εννιολογικά. Μέσω των συναρτήσεων των vector (find, sort, erase, push_back) όλες οι λειτουργίες του καλαθιού μπορούν να πραγματοποιηθούν χωρίς κάτι επιπρόσθετο από μεριάς μας. Σε σχέση με άλλες ιδέες που είχα όπως το cart να αποτελεί ξεχωριστή κλάση η συγκεκριμένη υλοποίηση μας βοηθάει να αποφύγουμε την αναγκαστικη «συνδεση» της customer με μια πιθανη cart και να περιπλεξουμε περαιτερω τη λογικη του κωδικα. Σε σχέση με map για τη χρήση που του κάνουμε είναι πιο αποδοτικό το vector, μιας και δεν θα φτάσουμε συχνά σε πολύ μεγάλο αριθμό αποθηκευμένων δεδομένων.

Void SearchProducts: *Βλέπε admin. (με διαφορά ότι το input δεν βρίσκεται πλέον στη συνάρτηση αυτή αλλά στο menu από το οποίο καλείται)*.

Void Checkout: Αρχικά ελέγχει μέσω map (.empty) αν το καλάθι είναι άδειο. Αρχικοποιούμε μεταβλητές (total cost, hasValidItems (flag) και το string που αποθηκεύουμε το επιθυμητό format) που θα χρησιμοποιήσουμε στη συνέχεια. Με λούπα αναζητάμε κάθε στοιχείο του καλαθιού στο vector products και αν υπάρχει αρκετό απόθεμα συνυπολογίζεται στο συνολικό κόστος και προστίθεται στο order data για να εκτυπωθεί / αποθηκευτεί στο format που επιθυμούμε. Αν το cart έχει διαθέσιμα αντικείμενα προχωράμε στην επόμενη φάση όπου και αποθηκεύουμε τη διαφορά αποθέματος και το ιστορικό.

Void AddToCart: Αναζητάμε βάση τίτλου το προϊόν που θέλουμε να προσθέσουμε στο vector. Αν υπάρχει γίνεται ένας πρώτος έλεγχος στο διαθέσιμο απόθεμα. Αν είναι αρκετό το «δεσμεύουμε» και αποθηκεύουμε στο vector.

Void removeFromCart: Αναζητάμε βάση τίτλου το προϊόν που θέλουμε να αφαιρέσουμε απο το vector. Αν υπάρχει απελευθερώνουμε το δεσμευμένο απόθεμα και το αφαιρούμε από το vector.

Void ChangeProductQuantity: Αναζητάτε βάση τίτλου το προϊόν που θέλουμε να επεξεργαστούμε. Αν υπάρχει ανάλογα την αλλαγή που επιθυμούμε, είτε αποδεσμεύουμε τη διαφορά των ποσοτήτων είτε δεσμεύουμε επιπρόσθετη.

Void displayCart: Αντίστοιχα με το checkout, περιηγούμαστε στο vector από την αρχή μέχρι τέλους απλά εδώ το εκτυπώνουμε κιόλας στο συγκεκριμένο format.

Void viewOrderHistory: Φορτώνει το αποθηκευμένο ιστορικό του πελάτη και το εκτυπώνει μέσω string.

Product

Η κλάση products ξεκινάει δηλώνοντας όλα τα στοιχεία που χρειάζεται ένα αντικείμενο (Τίτλος, περιγραφή, κατηγορία, υποκατηγορία, τιμή, μονάδα μέτρησης, διαθέσιμη ποσότητα) με τα οποία καλείται και ο constructor. Υπάρχει επιπρόσθετα η orderCount που μετράει πόσες φορές ένα προϊόν βρίσκεται εντός μιας ολοκληρωμένης παραγγελίας, προκειμένου να βρεθούν τα στατιστικά που χρειάζεται ο admin.

Getters/Setters: έχουν δημιουργηθεί βάση των απαιτήσεων του προγράμματος ακριβώς όσοι χρειάστηκαν

Bool checkAndUpdateProductQuantity: Η συνάρτηση εννιοποιεί τη διαδικασία ελέγχου διαθέσιμης ποσότητας του εκάστοτε αντικειμένου και την αφαίρεση της ζητούμενης από τη συνολική, βοηθάει σε διάφορες συναρτήσεις για να διασφαλίσει την έγκυρη ενημέρωση όλων των παραμέτρων.

Void ReleaseQuantity: Αυτοματοποιεί τη διαδικασία επιστροφής και ενημέρωσης του stock, όταν ένα προϊόν αφαιρείται από ένα καλάθι.

Void increaseOrderCount: Αυξάνει τον δείκτη orderCount του εκάστοτε προϊόντος

Void displayProductDetails: Εκτυπώνει όλες τις πληροφορίες ενός προϊόντος.

Void displayProductChecking: Ενσωματώνει στην displayProductDetails έλεγχο που βάση της ιδιότητας του χρήστη δίνει πρόσβαση στα επιθυμητά χαρακτηριστικά (επιπρόσθετα για admin orderCount).

UserManagment

Η συνάρτηση αυτή είναι υπεύθυνη για την μεταβίβαση πληροφοριών από αρχεία στο πρόγραμμα και πάλι πίσω. Μάλλον πιο εύστοχο όνομα θα ήταν το FileManagment, καθώς ασχολείται με την ορθη φορτώση και μεταβίβαση κάθε πληροφορίας (από που έρχεται το input, διαθέσιμοι εγγεγραμμένοι χρήστες, διαθέσιμα προϊόντα, ιστορικό πελατών). Χρησιμοποιούμε vector για την δυναμική προσθήκη χρηστών και την αναζήτηση τους. Αντίστοιχα με το Product που έχω εξηγήσει και προηγουμένως που χρησιμοποιείται. Η αποθήκευση των users και των products σε vectors έγινε για χάρη της απλότητας και διασφάλιση χαμηλού υπολογιστικού κόστους σε μικρές ποσότητες. Επιπρόσθετα στη κλάση αρχικοποιείται αντικείμενο της istream για τον χειρισμό του input με το οποίο αρχικοποιείται και ο constructor και pointer σε συγκεκριμένο χρήστη που χρησιμοποιείται από το registration προκειμένου να διασφαλιστεί η ομαλή μετάβαση από την εγγραφή στην χρήση του χρήστη.

Trim: Χρησιμοποιείται για την αποφυγή των κενών χαρακτήρων (προκειμένου να μην έχουμε θέμα εντοπισμού σε διάφορα αντικείμενα και να μπορούμε να συγκρίνουμε μεταβλητές string πιο εύκολα). Αναζητά τον πρώτο μη κενό χαρακτήρα και φτάνει ως τον τελευταίο μη κενό χαρακτήρα / αν όλο είναι κενοί χαρακτήρες επιστρέφει κενό string, αλλιώς επιστρέφει το string χωρίς περιττούς κενούς χαρακτήρες (π.χ. “Tech” και όχι “ Tech ”).

User* UserManagment Registeruser: Η συνάρτηση αυτή είναι υπεύθυνη για την εγγραφή νέων χρηστών. Διασφαλίζει ότι δεν υπάρχουν υπολειπόμενοι χαρακτήρες στον buffer προκειμένου να μην γίνει λάθος εισαγωγή δεδομένων, αρχικοποιεί και αποθηκεύει έναν user customer/admin ανάλογα την επιλογή και επιστρέφει έναν δείκτη σε αυτόν με τον οποίο και εκτελεί στη συνέχεια το ανάλογο Main Menu.

User* UserManagment Login: εξακριβώνει τον συνδυασμό username και password και εφόσον υπάρχει στους εγγεγραμένους χρήστες επιστρέφει pointer σε αυτόν με τον οποίο τρέχει και το ανάλογο main menu.

Void SaveUsersToFile: Παιρνει ως όρισμα το όνομα του αρχείου. Αν το αρχείο βρεθεί αποθηκεύει τα αντικείμενα τύπου user σε admin ή customer βάση του ορίσματος isAdmin που έχει δοθεί.

Void loadUsersFromFile: Παίρνει ως όρισμα το όνομα του αρχείου και φορτώνει τους χρήστες χωρίς περιττά κενά (, και κενά) γραμμή προς γραμμή, περνώντας τους στο vector users.

Void addProduct: περνάει ένα product στο vector products.

Void SaveProductsTofile: Αν το product είναι ορθά αρχικοποιημένο, το αποθηκεύει στο δοσμένο αρχείο με το κατάλληλο format.

Void LoadCategoriesFromFile: Φορτώνει τις κατηγορίες και τις υποκατηγορίες σε map για να διατηρηθεί η σύνδεση μεταξύ των δύο και χρησιμοποιείται η trim για να αποφευχθούν περιττά κενά που υπήρξαν κατά το debug και οδηγούσαν τα unit tests σε fail.

Void LoadProductsFromFile: Αντίστοιχη λογική με το Users, φορτώνει τα προϊόντα χωρίς περιττούς χαρακτήρες.

Void saveorderHistory: Αποθηκεύει το string που δίνεται ως order data στο _history.txt αρχείο που ανταποκρίνεται στο username που δίνεται.

E- shop

Αποτελεί τη κλάση υπεύθυνη για το interface/menu. Έχει ως στοιχεία τη πηγή του input και ένα αντικείμενο της κλάσης usermanagment με το οποίο διαχειρίζεται χρήστες/προϊόντα κλπ. Το menu χωρίζεται σε τρια στάδια (το κοινό για όλους τους χρήστες) 1)login , 2) το menu των δυνατοτήτων ενός admin 3) το menu των customer.

Void Run: Η run φορτώνει εκ των προτέρων τα απαραίτητα αρχεία για τη λειτουργια του προγράμματος. Καλεί τη main menu και στον τερματισμό αυτής αποθηκεύει τυχόν αλλαγές που έγιναν κατά τη διάρκεια αυτής στα αρχεία (σαν τελευταία δικλείδα ασφαλείας να μην χαθούν δεδομένα).

Void displayMainMenu: Παρέχονται δύο δυνατότητες login ή register, η κάθε μια καλεί τις αντίστοιχες συναρτήσεις της userManagment. Βάση του pointer που επιστρέφεται από αυτές ανοίγει το αντίστοιχο menu (customer/ admin). Υπάρχει «έτοιμη» και η επιλογή απευθείας κλεισίματος (exit) του προγράμματος η οποία χρησιμοποιήθηκε για debug.

Void AdminMenu: Αφού διασφαλιστεί ότι ο pointer σε admin έχει διατηρηθεί με το άνοιγμα της συνάρτησης, παρέχονται οι δυνατότητες «Προσθήκη , αφαιρεση, επεξεργασία, αναζήτηση προιοντος» καθώς και η εμφανιση out of stock προιοντων και των πιο δημοφιλων. Η καθεμια απο αυτες καλει τις αντιστοιχες συναρτησεις. Τελος υπαρχει η δυνατοτητα κλεισιματος.

Void CustomerMenu: Αντίστοιχα με το admin, δίνονται οι δυνατότητες που περιγράφονται στα samples της εργασίας, με τη διαφορά ότι το input που χρειάζεται κάθε function πλέον ζητείται εντός της CustomerMenu.

Oop24

Το κύριο εκτελέσιμο της εργασίας. Ελέγχει αν τα ορίσματα με τα οποία κλήθηκε είναι σωστά (αριθμητικά τουλάχιστον) και στη συνέχεια καλεί την eshop.