DOCUMENTAȚIE TEHNICĂ

~ proiect final ~

Proiectul a necesitat implementarea unui magazin rock online minimalist care să simuleze gestionarea angajaților și produselor, precum și procesarea de comenzi și generarea de rapoarte. Folder-ul creat pentru acest proiect conține 1931 de linii de cod și 28 de fișiere, împărțite astfel: 14 fișiere .cpp și 14 fișiere .h.

Meniul principal conține 4 submeniuri și opțiunea de „Exit”, care permite ieșirea din program. Primul submeniu se ocupă de gestiunea angajaților și are ca opțiuni adăugarea, modificarea, eliminarea și afișarea de angajați, precum și opțiunea de reîntoarcere în meniul principal. Pentru angajați au fost implementate 4 clase, o clasă de bază Angajat și trei clase derivate: Manager, Operator și Asistent. Clasa Angajat conține atributele comune ale claselor, așa cum sunt menționate în cerința proiectului, iar metodele implementate sunt constructorul fără parametrii (declarat default), constructorul cu parametrii, o funcție virtuală de afișare, o funcție de afișare a tuturor ID-urilor, o funcție virtuală de calculare a salariului și o funcție de calculare a gradului de vechime în companie, funcții get pentru accesarea atributelor și o funcție set necesară pentru a modifica numele unui angajat. Clasa Manager nu are niciun atribut nou față de clasa de bază Angajat; ca metode au fost implementate constructorul fără parametrii (declarat default), constructorul cu parametrii care apelează constructorul cu parametrii al clasei de bază, funcția de afișare, precum și funcția de calculare a salariului așa cum este menționat în cerința proiectului. Pentru clasa Operator au fost adăugate atributele: numOrders (număr întreg, care reține numărul total de comenzi preluate de un anumit operator), vector<Comanda> orders (vector care conține comenzile pe care un operator le procesează la un anumit moment de timp, având orders.size() <= 3), precum și max_valueOrd (număr real care reține comanda cea mai valoroasă procesată de un operator până la un anumit moment de timp). Metodele implementate în această clasă sunt: constructorul fără parametrii, constructorul cu parametrii (care apelează constructorul cu parametrii implementat în clasa de bază), redefinirea funcțiilor de afișare și calculare de salariu, funcții get pentru a avea acces la atributele noi, declarate în clasa Operator, funcții de adăugare și eliminare de comenzi din vectorul de comenzi, precum și două funcții de reset, care vor fi folosite în cadrul submeniului destinat procesării de comenzi pentru a goli operatorii de comenzi la o nouă citire din fișier. Clasa Asistent nu are niciun atribut nou față de clasa de bază Angajat. A fost implementată similar cu clasa Manager și conține următoarele metode: constructor fără parametrii, constructor cu parametrii, precum și redefinirea funcțiilor de afișare și calculare a salariului.

Pentru vectorul de angajați declarat în funcția main, vector<Angajat*> employees, au fost realizate două fișiere helperFuncEmployee.h, respectiv helperFuncEmployee.cpp, care permit manipularea datelor existente în vector. Au fost implementate funcții de validare (tip bool) pentru verificarea CNP-ului, a numelui, a unicității ID-ului, a datei de angajare și a tipului de angajat introdus, conform cerinței proiectului, precum și funcția de adăugare care folosește funcțiile menționate anterior pentru citirea datelor de la tastatură și adăugarea unui obiect nou în vector, funcția de modificare și funcția de eliminare din vector. De asemenea, a fost

implementată o funcție countEmployees(…), care permite validarea numărului minim de angajați de fiecare tip necesar pentru funcționarea magazinului. Fiind de tip bool, ea permite sau nu accesarea meniului de procesare de comenzi și afișează un mesaj corespunzător.

Pentru gestiunea produselor au fost implementate patru clase: clasa Produs este clasa de bază pentru clasele derivate Disc și Vestimentație, iar clasa Vintage este derivată la rândul ei din clasa Disc. Clasa de bază conține atributele menționate în cerința proiectului (tip, cod, denumire, stoc și preț de bază), iar metodele implementate sunt: constructor fără parametrii (declarat default), constructor cu parametrii, funcții virtuale de afișare și calculare a taxelor de împachetare și livrare, funcții get pentru acces la atributele clasei, precum și o funcție set de setare a stocului, necesară în cadrul opțiunii de modificare a produsului. Clasa derivată Vestimentație conține două atribute noi, culoare și marcă, iar metodele implementate sunt: constructor fără parametrii și constructor cu parametrii, redefinirea funcțiilor de afișare și calculare a taxelor suplimentare. Clasa Disc este implementată similar, având atributele noi conform cu cerința proiectului: casă de discuri, dată apariție în magazine, trupă și album. Metodele implementate au fost: constructorul fără și cu parametrii, redefinirea funcțiilor de afișare și adăugare de taxe. Clasa Vintage, derivată din clasa Disc, moștenește toate atributele clasei Disc, la care se adaugă atributele mint și coef_rarity. Metodele implementate sunt: constructorii cu și fără parametrii, funcțiile de afișare și adăugare de taxe de împachetare și livrare. Pentru manipularea vectorului de produse, vector<Produs*> products, din funcția main, au fost create fișierele helperFuncProduct.h și helperFuncProduct.cpp, care conțin funcții de validare și afișare de atribute, folosite în cadrul funcțiilor de adăugare, modificare și eliminare de produse. Există, de asemenea, o funcție countProducts(…) care, la fel ca în cazul angajaților, verifică numărul de produse din fiecare tip și permite sau nu accesul la submeniul de procesare comenzi.

Simularea comenzilor a fost cu siguranță cea mai solicitantă parte a proiectului; au fost implementate pentru gestiunea comenzilor două clase: Comanda și Request. Clasa Request conține ca atribute un șir de caractere și un număr întreg, care reprezintă un cod și o cantitate asociate unui produs din vectorul de produse. Pentru această clasă au fost implementate constructorii cu și fără parametri, funcții get, precum și funcțiile friend necesare redefinirii operatorilor: operator<< și operator>>. Clasa Comanda conține, în afara atributelor menționate în cerința proiectului: id unic, data preluării comenzii, valoare minimă, durată de împachetare, atributele: numRequest (număr întreg, necesar pentru a ști câte linii se vor citi din cadrul fișierului), vector <Request> requests (vector de obiecte de tip Request care conțin un cod și o valoare întreagă), valoarea comenzii cu taxele adăugate (număr întreg necesar în generarea raportului care conține top 3 cele mai valoroase comenzi), precum și o valoare reală, timeSimulator. Metodele implementate sunt: constructor cu și fără parametri, funcție de afișare comenzi, funcții get și set, precum și o funcție de redefinire a operatorului >>. Submeniul de procesare a comenzilor permite citirea din fișier, adăugarea unei noi comenzi, precum și afișarea tuturor comenzilor. La fiecare citire din fișier, vectorul <Comanda> orders este golit, vectorul de comenzi asociat fiecărei comenzi este golit, numărul total de comenzi (declarat static int) este resetat, numărul total de comenzi procesate de un operator este resetat, precum și salariul din luna curentă. Citirea din fișier reprezintă popularea vectorului de comenzi

declarat în main și simularea încărcării operatorilor. Funcționalitatea de adăugare a unei noi comenzi este de fapt scrierea în fișier a unei noi comenzi care poate fi interpretată, și deci validată, doar în cadrul unei noi citiri din fișier. Pentru manipularea comenzilor au fost folosite fișierele helperFuncOrder.h și helperFuncOrder.cpp, care conțin următoarele: funcție de validare a unui request, care verifică existența codului și disponibilitatea stocului în vectorul de produse, funcția decrementStock(…), care modifică numărul de produse de un anumit tip din stoc, dacă acel produs face parte dintr-o comandă validă, funcțiile minOrders(…) și findFreeOperator(…), care găsesc angajatul cu cele mai puține comenzi procesate până la un anumit moment de timp, funcția removeExpiredOrders(…) verifică în toți vectorii de comenzi asociați operatorilor și curăță vectorul de comenzile terminate (atributul timeSimulator al unei comenzi este inactiv până în momentul în care comanda este asignată unui operator; din acel moment, la fiecare trecere prin bucla de procesare de comenzi, pentru toate comenzile asignate până la acea iterație valoarea sa crește cu o valoare setată și se verifică dacă acest atribut a depășit timpul de împachetare și livrare; dacă da, comanda este eliminată și astfel funcționează simularea încărcării și eliberării de comenzi). Funcția processFileOrder(…) calculează valoarea comenzii și timpul de împachetare și verifică restricțiile impuse asupra comenzilor, apelând toate funcțiile menționate anterior. O explicație mai detaliată a mecanismului de funcționare este următoarea: se verifică dacă există vreun operator liber, iar acesta este cel cu numărul minim de comenzi procesate; dacă da, se verifică ca lista de așteptare să fie goală, astfel încât să nu fie procesată o comandă înaintea uneia care deja era în așteptare și se ia o decizie cu privire la următoarea comandă de procesat; dacă nu, aceasta va fi adăugată direct în vectorul de așteptare.

Rapoartele generate în ultimul submeniu al programului sunt în format .csv, nu a fost necesară implemetarea unor clase, ci doar a funcției generateReports(…) din cadrul fișierului helperFuncReport.cpp, care folosește metode specifice bibliotecii <algorithm> pentru sortare (sort), găsire element maxim (max_element).

Elementele tehnice noi identificate în proiect au fost golirea buffer-ului, folosind comanda cin.ignore(numeric_limits<streamsize>::max(), '\n'), precum și lucrul cu fișiere in format .csv, însă în rest consider că materia predată în cadrul cursului a acoperit toate informațiile necesare rezolvării acestui proiect. Proiectul folosșste elemente de modern C++, precum: expresii lambda, folosite în majoritatea cazurilor împreună cu funcțiile bibliotecii <algorithm>, folosirea cuvântului cheie auto (ex: în functia removeExpiredOrders(…)), inițializarea uniformă, pentru inițializarea vectorului de număr maxim de zile ale fiecărei luni din an folosit la validarea datei, nullptr, folosit în cadrul funcției findFreeOperator(…), precum și smart pointers, mai precis unique pointers în functia main, la popularea vectorilor de angajați și produse. Au fost folosite de asemenea bibliotecile: string, STL(vector), chrono(pentru data/ora).