PROJEKTNI ZADATAK

Student:

1. Husejn Omerbašić

Sadržaj

[Sadržaj 1](#_Toc9589)

[1 Uvod 2](#_Toc7065)

[1.1 Tehnički zahtjevi 2](#_Toc5407)

[2 Metodologija rada 3](#_Toc2988)

[3 Plan za unaprijeđenje 4](#_Toc23004)

[4 Zaključak 4](#_Toc2097)

# Uvod

U supermarketima stalno dolazi do nekih promjena, kao što je povećanje ili smanjenje asortimana, dolazak novih radnika i slično. Da bi supermarket upješno poslovao potreban je efikasan način da se te promjene zabilježe. Kao prvi korak ka optimizaciji procesa izmjene stanja i evidencije, najjednostavnije je odmah krenuti u digitalizaciju. Potrebno bi bilo kreirati bazu podataka u koju bi se smješatle nove izmjene, a neophodno bi bilo i za bazu podataka izraditi DBMS (DataBase Management System) sa jednostavnim sučeljem koje bi krajnjem korisniku na jednostavan način pružilo mogućnosti izvršavanja CRUD (Crate, Read, Update, Delete) naredbi i više (Sort, Search). Cilj ovog rada je da se razvije jedna takva aplikacija koja će omogućiti uposlenicima da prate i mijenjaju stanja artikala na svim policama, stanje kasa kao i da vode evidenciju o zaposlenicima.

## **Tehnički zahtjevi**

Prvo je bilo potrebno utvrdit korisničke zahtjeve. Pošto zahtjevi klijenta nekada mogu biti vrlo neodređeni ili nedorečeni, bilo je potrebno sprovesti detanjan intervju sa klijenton nakon čega smo došli do sljedećih zahtjeva:

1. Omogućiti operacije unosa, izmjene, pregleda i brisanja: radnika, kasa, polica i artikala sa potrebnim detaljima za vođenje supermarketa.
2. Unos novih stavki, te izmjenu postojećih vršiti sa tastature.
3. Željenu radnju (unos, izmjena, pregled, brisanje) korisnik unosi preko tastature na osnovu ponuđenih opcija u terminalu.
4. Za novi unos definirati potrebne kolone i ograničiti unos ukoliko se ne unesu potrebni podaci.
5. Za povezivanje tabela koristit će se ključevi, koji predstavljaju jedinstveni identifikator određene stavke:

* radnik: tekstualni zapis JMBG od 13 cifara,
* kasa: tekstualni zapis od 3 znaka (K01, K02, …, K0n),
* polica: tekstualni zapis od 5 znakova, gdje su prva 3 slova i označavaju kategoriju proizvoda koje sadrže (proizvoljne skraćenice za mliječne proizvode, pića, sredstva za čišćenje, školski pribor, itd.), a posljednja 2 brojevi i označavaju redni broj police.
* artikal: tekstualni zapis od 10 znakova, gdje su prva 3 slova i označavaju kategoriju proizvoda (proizvoljne skraćenice za mliječne proizvode, pića, sredstva za čišćenje, školski pribor, itd.), sljedeća 3 također slova i bliže označavaju proizvod, a posljednja 4 brojevi i označavaju redni broj određenog artikla.

1. Napraviti dokumentaciju projekta u obliku Word dokumenta. Koristiti ovaj dokument kao šablon, te početi sa 5. poglavljem svoj izvještaj o projektu.

# Metodologija rada

Dok smo razvijali ovu aplikaciju koristili smo IDE CodeBlocks I najnoviji standard iz 2017.godine C++17. Korištene su tehnologije:

-Osnovu objektno orijentiranog pristupa programiranju čine klase odnosno razredi (engl. class) .Klasa je korisnički definiran tip podatka koji objedinjuje stanje (objekti) I ponašanje (funkcije) u jednu logičku cjelinu. Nakon definiranja klase stvaramo objekte pomoću kojeg pozivamo metode(Funkcije čiji su prototipovi navedeni unutar same klase).Poznate su greške koje u programiranju mogu nastati kao posljedicaneinicijaliziranih vrijednosti promjenljivih. Ozbiljnost ovih grešaka raste sa porastom složenosti tipovapromjenljivih. Stoga objektno zasnovano programiranje nudi izlaz iz ove situacije uvođenjem konstruktora, kojiprimoravaju korisnika da mora izvršiti inicijalizaciju objekata (i to na strogo kontroliran način) prilikom njihovedeklaracije ili stvaranja dinamičkih objekata.konstruktori definiraju skupinu akcija koje se automatski izvršavaju nad objektom onogtrenutka kada dođe do njegovog stvaranja.Za razliku od konstruktora, koji definiraju skupinu akcija kojese automatski izvršavaju prilikom stvaranja nekog objekta, destruktori predstavljaju skupinu akcija koje seautomatski izvršavaju prilikom uništavanja objekta, tačnije kada objekat prestaje postojati.Zadatak destruktora je da oslobodi dodatne resurse koje je objekat zauzeotokom svog ţivota, recimo prilikom poziva konstruktora ili neke metode koja vrši zauzimanje dodatnihresursa (ti resursi su obično dinamički alocirana memorija.

-U većini programa postoje cjeline koje se ponavljaju više puta. Kako bi se izbjeglo ponovno pisanje istih ili vrlo sličnih dijelova programa, može ih se izdvojiti, imenovati i spremiti na tu predviđenu lokaciju. Izdvojeni nizovi naredbi koji čine logičke cjeline, a obavljaju tačno utvrđene zadatke nazivaju se funkcije. Svaka se funkcija sastoji od skupa naredbi koje određuju što i kako funkcija radi. Taj se skup naredbi naziva tijelo funkcije. Tijelo funkcije započinje nakon otvorene vitičaste zagrade ({), a završava zatvorenom vitičastom zagradom (}).Funkcija završava ključnom riječi return. Iza ključne riječi return se zapisuje podatak kojeg funkcija vraća pozivatelju kao rezultat svog rada.

-Vector je definiran u zaglavlju vector. Klasa Vector je generički kontejner za objekte istog tipa I može sadržati proizvoljan broj objekata. Vektorska promjenljiva u toku svog života može povećavati i smanjivati broj svojih elemenata(samim tim, zauzeće memorije koje zauzima neka vektorska promjenljiva može se dinamički mijenjati, doknizovi od trenutka svoje deklaracije do kraja postojanja zauzimaju uvijek istu količinu memorije). Brojelemenata pri deklaraciji vektora navodi se u običnim, a ne u uglastim zagradama,čime je istaknuta razlikaizmeđu zadavanja broja elemenata i navođenja indeksa za pristup elementima.Nije na odmet napomenuti da se pri radu sa vektorima također često koriste “typedef” deklaracije sa ciljem pojednostavljenja sintakse i lakšeg kucanja.

# Plan za unaprijeđenje

Kroz izradu projekta uočili smo da ovakav program zahtjeva dugotrajno razvijanje i unaprijeđenje kako bi u svakom trenutku na najbolji mogući način mogao odgovoriti zahtjevima korisnika. Program u trenutnom stanju daje prividan izgled funkcionalnosti koje bi se očekivale od programa koji bi bio spreman za implementaciju, te stoga sadrži neke nedorečene stavke, funkcionalnosti i podfunkcionalnosti, te neke od neistraženih scenarija koji bi mogli dovesti do nestabilnosti sistema. Iz tog razloga program još uvijek zahtijeva dodatnu doradu. Tijekom izrade ovog programa dobili smo nekoliko ideja koje bi u budućnosti mogle imati značajan utjecaj na kvalitet i efikasnost korištenja ovog programa. Na žalost, tehnologijom koju smo koristili za izradu ovoga programa ne bi na efikasan način mogli kreirati program koji bi olakšao korištenje, te je stoga naš prijedlog za usavršavanje ovog programa izrada visualne GUI aplikacije koja bi pojednostavila upisnu formu za upravljanje podacima u “bazi podataka”, te je prikazala u prozoru aplikacije umjesto u konzoli. Korištenjem ove tehnologije bi se umnogome olakšao put za kreiranje stvarne baze podataka i povezivanje iste na naš program.

# Zaključak

Koristeći sve gore navedene metodologije stvorena je aplikacija koja će pomoći radnicima u njihovim svakodnevnim poslovnim obavezama. Radnici će sa ovom aplikacijom moći lakše voditi evidenciju o svim promjenama u cijenama, artiklima, zaposlenicima, stanjima na policama i sl. Stvarajući ovu aplikaciju prvenstveno smo imali na umu korisnike. Željeli smo da stvorimo aplikaciju koja je jednostavna za korištenje i razumijvanje, samim tim je i sučelje prilagođen njima. Uvođenjem ove tehnologije očekuje se veća efikasnost poslovanja, smanjeni troškovi papira, ali i smanjenje broja potrebnih zaposlenika na prijemu, sortiranju i evidentiranju robe. Neophodno će biti uposliti osobu koja će raditi na održavanju ovakvog sistema i koja će biti obučena za “troubleshooting” svih mogućih scenarija za dati program.