Norges teknisk–naturvitenskapelige universitet
Institutt for datateknologi og informatikk

TDT4102 Prosedyre- og objektorientert programmering Vår 2024

Øving Inspera

Praktisk:

Det er veldig viktig at du leser det praktiske før du begynner på oppgavene.
Alle oppgavene i del 3 er satt opp slik at de skal besvares i eksisterende funksjoner i .cpp-filene i den utdelte koden. I filene du skal skrive i vil du finne to kommentarer for hver oppgave, som definerer begynnelsen og slutten på koden du skal føre inn. Kommentarene er på formatet

`// BEGIN: 1a` og `// END: 1a`

Det er veldig viktig at alle svarene dine er skrevet mellom slike kommentar-par. Kode utenfor et slikt par blir ikke vurdert. BEGIN- og END-kommentarene skal **IKKE** fjernes!

Den utdelte koden kan hentes slik:

- Last ned .zip-filen med utdelt kode som ligger på Inspera. Pakk denne ut på ønsket sted på PC-en din, og åpne deretter den utpakkede mappen i VSCode (slik du er vant til å åpne prosjektmapper fra øvingsopplegget). Pass på at du kun har ett nivå i mappen din i VS Code (det kommer til å bli krøll hvis kodefilene ikke ligger i den ytterste mappen).

Den utdelte koden inneholder **IKKE** konfigurasjonsfilene som er nødvendig for å kjøre programmet. Du må derfor kjøre "TDT4102: Create Project from TDT4102 Template -> **Configuration only**" fra Comand Palette i VS Code. Dette skal gjøres i mappen du legger den utpakkede koden i, altså i mappen du skal jobbe med øvingen.

Når du vil levere, må du lagre filene, og levere dem som en zippet mappe i Inspera. Du kan bruke fagets VS Code Extension for å få til dette (cmd/ctrl + shift + P -> TDT4102: Prepare a zip file for delivery). **Husk at hvis du endrer på koden, må du først lagre og deretter lage zip-filen på nytt.**

PS: Når du løser en deloppgave kan du anta at alle de andre funksjonene du har laget fram til da fungerer som de skal, selv om dette ikke er tilfellet.

Intro:

Tema: Sit-kantinen – Kulinarisk Eventyr
Sit-kantinene på Gløshaugen har gått tom for ideer for matretter på den ukentlige menyen, og trenger din hjelp for å **spice** den opp. Din oppgave er å utvikle et system for å generere en tilfeldig ukentlig meny for hver kantine på Gløshaugen.

For å komme i gang, har du fått utdelt klassen `Cafeteria`, som blant annet inneholder medlemsvariabelen `weeklyMenu`. Denne ukentlige menyen består av 7 daglige menyer, representert av `DailyMenu`-structen. En daglig meny består av en hovedrett og en siderett. Økonomisjefen i sit-kaninene har i tillegg funnet ut av at ved å legge ved et beskrivende ord foran hovedretten, så kan de øke prisen på retten. Derfor skal alle retter ha et adjektiv som f.eks. vegansk, glutenfri, fullkorn, osv. foran hovedretten. Adjektivene ligger i et `map`, `adjectivePriceModifiers`, der adjektivet er nøkkelen til mapet, og verdien er den tilsvarende prisøkningen for menyen. Videre ligger hovedrettene og siderettene i tilsvarende maps som heter henholdsvis `mainDishPrices` og `sideDishPrices`.

For å gjøre oppgaven overkommelig, har du fått utdelt starter-kode. Blant annet ligger det kode for å teste programmet i `main.cpp`. Etter hvert som du lager kode, kan du kjøre programmet, og sjekke om oppførselen er som forventet.

Lykke til!

1 Implementasjon av daglig meny (85 poeng)

a) *Oppgaven løses i `DailyMenu.cpp` (15 poeng)*
Sit trenger hjelp til å opprette en daglig meny ut ifra gitte adjektiv og retter.
Implementer funksjonen `createDailyMenu(std::string adjective, std::string mainDish, std::string sideDish)`.
Funksjonen skal opprette en ny DailyMenu ut ifra de gitte parametrene, og kalkulere prisen ut ifra følgende regnestykke:
`price = adjectiveModifier * (mainDishPrice + sideDishPrice)`
Anta at alle parametrene har gyldige verdier.

b) *Oppgaven løses i `DailyMenu.cpp` (10 poeng)*
For at studentene skal kunne se menyen, trenger Sit-kantinen en måte å skrive ut den daglige menyen.
Implementer overlastingen av `operator<<` for `DailyMenu`-structen.
Hvordan du velger å skrive ut menyen og prisen er opp til deg.

c) *Oppgaven løses i `Cafeteria.cpp` (10 poeng)*
Hver Sit-kantine skal ha individuell meny, og derfor trenger vi en måte å hente ut en meny fra en vilkårlig kantine. Kantinene beskrives av `Cafeteria`-klassen.
Implementer medlemsfunksjonen `Cafeteria::getDailyMenu(Weekday w)`.
Denne skal hente ut den daglige menyen til en kantine for en gitt ukedag.

d) *Oppgaven løses i `Cafeteria.cpp` (15 poeng)*
For at en Sit-kantine skal kunne servere en daglig meny, må kokkene kunne sjekke om de har de riktige rettene.
Implementer medlemsfunksjonen `Cafeteria::isInStock(DailyMenu menu)`.
Funksjonen skal sjekke om kantinen har de riktige rettene for å lage den daglige menyen `menu`.

e) *Oppgaven løses i `Cafeteria.cpp` (20 poeng)*
Det er ønskelig å lagre den ukentlige menyen til en fil slik at den kan sendes til sjefskokken for godkjenning.

Implementer funksjonen `Cafeteria::saveToFile(std::string fileName)`.

Funksjonen skal lagre den ukentlige menyen til en fil. Du velger selv format for hvordan menyen skal bli lagret. Her er det lurt å bruke tidligere funksjoner du har definert.

2 Generere ukentlig meny fra lager-fil (40 poeng)

I gjennom hele denne koden er det viktig å merke seg hvilke funksjoner som allerede er implementert, da flere av disse kan gjøre oppgavene enklere.

a) *Oppgaven løses i `Cafeteria.cpp` (20 poeng)*
Det er ønskelig å kunne importere lagerbeholdningen fra en fil.

Implementer funksjonen `void Cafeteria::importStock(std::string fileName)`.

Her skal du lese fra fil `fileName` som blir tatt inn i funksjonen og legge til innholdet fra filen til `std::map<std::string, int> stock` som er definert i klassen `Cafeteria`. I `stock.txt`-filen ligger hoved- og sideretter sammen med hvor mye som er tilgjengelig i lageret. Verdiene er separert med komma.

b) *Oppgaven løses i `DailyMenu.cpp` (25 poeng)*
Kokkene er lei av å måtte tenke ut nye retter hele tiden, og ønsker en funksjon for å generere tilfeldige menyer.

Implementer funksjonen `DailyMenu getRandomMenu()`.

Denne skal bruke tilfeldig generert tall til å produsere en tilfeldig meny og returnere en instans av structen `DailyMenu`. Her kan det være lurt å ta i bruk `mapSize`, som angir størrelsen på relevante maps, og funksjonen `getDishFromNumber`, som tar inn et `map` og en `int` og returnerer en nøkkel fra mappet (merk at denne er null-indeksert).

c) *Oppgaven løses i `Cafeteria.cpp` (15 poeng)*
For å spare tid skal hele ukesmenyen fylles opp av tilfeldige retter.

Implementer funksjonen `std::vector<DailyMenu> generateWeeklyMenu()`.

Denne skal returnere en vector med 7 elementer av typen `DailyMenu`. DailyMenus skal genereres tilfeldig, husk at du kan bruke tidligere funksjoner du har implementert.