Mittuniversitetet

MID SWEDEN UNIVERSITY

DT018G, Introduktion till programmering i C++

Projektuppgift

I projektuppgiften skall studenten redovisa kunskaper gällande...

- … användning av strängar!
- … användning av std::vector!
- … användning av funktioner!
- … användning av uppräknande datatyper!
- … användning av datastrukturer!
- … skrivning och läsning av data från textfiler!
- … skapande av kod med god struktur!
- … sammanhållning samt upprätthållande av god filstruktur!

Innehållsförteckning

Projektuppgift

Du ska skapa ett större program som är uppbyggt av många funktioner vilka placeras i ett antal filer. Programmets uppgift är att hantera uppgifter om ett större antal personer i en lista. Programmet skall vara menybaserat och användaren skall kunna…

- … lägga till personer i listan
- … skriva ut listan med personer på skärmen
- … söka en person i listan
- … ta bort angivna personer från listan
- … sortera listan på några olika sätt
- … slumpa ordningen
- … spara listan till angiven textfil
- … läsa listdata från angiven textfil

I ditt tilldelade studentrepo finner du en katalogen `Project` som är avsedd för projektfilerna. `main.cpp` är programmets utgångspunkt och du fyller på med nödvändiga filer enligt följande filstruktur:

```
Project/include/... // *.h
```

```
Project/src/... // *.cpp
```

Notera att byggskriptet `CMakeLists.txt` behöver laddas om för att inkludera nya filer som läggs till. I **CLion** görs detta genom menyn `Tools → CMake → Reload CMake Project`

> *Läs igenom hela projektbeskrivningen innan du påbörjar arbetet och planerar för genomförandet!*

Kravspecifikation

Data

- Skapa en `struct` som hanterar persondata:
  - … Förnamn (`string`)
  - … Efternamn (`string`)
  - … Signatur (`string`). Se nedan för detaljer gällande signaturen!
  - … Längd i meter (`flyttal`)
- Deklarera denna `struct` i en headerfil med namnet `constants.h`
- Använd en `std::vector` för att hantera en lista med ett antal personer!

Användarinput, persondata

- Låt användaren mata in personens **förnamn**, **efternamn** och **längd**!
- Skriv kod som skapar en **signatur** (se nedan)!
- Tilldela **förnamn**, **efternamn**, **längd** och **signatur** till en instans av datastrukturen `person` som sedan läggs in i vectorn!
- Det ska vara möjligt att mata in namn som `Bo Einar von Trapp`, där `Bo Einar` är **förnamn** och `von Trapp` är **efternamn**!
- Programmet behöver **inte** ta hänsyn till andra bokstäver än `a-z`, `A-Z` i namnen!
- **Längd** anges med två decimaler. Om någon är `1.80 m` ska det räcka att mata in `1.8`

Signatur, tre svårighetsnivåer

1. Signaturens format är `xxxyyy`, där `xxx` är de tre första bokstäverna i **förnamnet** och `yyy` de tre första bokstäverna i **efternamnet**. Signaturen ska enbart bestå av gemener. Om förnamn eller efternamn innehåller färre än tre bokstäver så får signaturen bli kortare än sex tecken!
2. Som under punkt **1)** men om **förnamn** eller **efternamn** innehåller färre än tre bokstäver skall signaturen fyllas ut med tecknet `x`, så att den alltid blir sex tecken lång. Ex: `Bo Ek` blir `boxekx`
3. Som under punkt **2)** men kontroll ska göras så att varje signatur är **unik**. Det är lämpligt att använda formatet `xxxyyyzz` där `zz` är ett löpnummer!

Utskrift

```
***** NAME LIST *****
Number of persons in list: 6

Nr  Sign      Name                  Length [m]
 1. pelasp01  Pelle Asp                   1.94
 2. boxekx01  Bo Ek                       1.77
 3. evakas01  Eva Kask                    1.65
 4. evakas02  Eva Kaskad                  1.67
 5. olevon01  Ole Einar von Rosen         1.88
 6. betboo01  Betty Boop                  2.10

Tryck ned valfri tangent för att fortsätta...
```

- Skriv ut informationen kolumnvis enligt figur!
- Skriv **löpnummer** för varje rad!
- Kolumnerna skall innehålla **löpnummer**, **signatur**, **namn** och **längd**!
- **Längd** skrivs **högerjusterad** med **två decimaler**!
- Visa högst 20 personer åt gången. Låt användaren trycka på en tangent för att visa nästa 20 personer!

Sökning / Borttagning av person

- Låt användaren skriva in **signatur** som sökord!

- **Sökning:** Om sökt person finns i listan skrivs personen ut på skärmen med **signatur**, **namn** och **längd** enligt ovan!
- **Borttagning:** Om angiven person finns i listan raderas denna med en utskrift som bekräftar borttagningen!

- Om sökt person **inte** finns i listan skrivs ett felmeddelande som meddelar att personen saknas!

Sortering

- Använd `std::sort` från `<algorithm>`

- Sortera listan på **efternamn**…
  … i **stigande** bokstavsordning!
  … om efternamnen är lika skall listan sorteras på **förnamn**!
  … oberoende av om namnen är skrivna med gemener eller versaler: ***icke case sensitive!***
  … ingen hänsyn behöver tas gällande `ÅÄÖåäö`

- Sortera listan på signatur i **stigande** bokstavsordning!

- Sortera listan på längd i **fallande** ordning, d.v.s. **största** längden först!

- Skriv **en** funktion som utför alla sorteringar...
  … anropas från menyn (se nedan)!
  … använder en parameter för att avgöra vilken sortering som skall utföras när funktionen anropas!
  … de olika sorteringssätten skall anges med en `enum` (uppräkningsbar datatyp) som deklareras i `constants.h`

Slumpad ordning

- Blanda om listan så att namnen kommer i slumpad ordning!

Filhantering

- Spara listan med personer till en textfil

- Skriv en person per rad!

- Format: `first_nameDELIMlast_nameDELIMsignatureDELIMheight`

- `DELIM` är en avdelare som representeras av ett tecken som med säkerhet inte ingår i något namn eller signatur (exempelvis `|`). Deklarera `DELIM` som konstant i headerfilen `constants.h`

- Vid inläsning från fil skall aktuell lista ersättas!

Huvudprogram, innehåll

- En `std::vector` med data för personerna!

- Utskrift av menyval (se nedan)!

- Anrop av funktioner för de olika menyvalen!

- En utskrift av aktuellt antal personer i listan. Skriv antalet personer i anslutning till utskriften av menyvalen!

Funktioner

- Programmet ska vara funktionsbaserat. Detta innebär att enbart funktionsanrop får göras från switch-satsen i `main()`

- Placera funktionsprototyperna i en eller flera headerfiler, `*.h`

- Placera funktionsdefinitionerna i en eller flera definitionsfiler, `*.cpp`

Meny

- Skapa en meny som ger användaren följande möjligheter…
  … lägga till en person!
  … skriva ut listan med personer på skärmen!
  … söka en person i listan!
  … ta bort en person från listan!
  … sortering efter namn, signatur, längd samt slumpad ordning!
  … spara listan till en textfil. ***Låt användaren ange filnamn!***
  … läsa listan från en textfil. ***Låt användaren ange filnamn!***
  … avsluta programmet!

- Placera koden för menyn i en funktion. **Indata**: vector med menyvalstexterna, U**tdata**: valt menyalternativ som heltal!

Allmänt

- Huvudprogrammet (`main()`) skall placeras i en egen fil!

- Globala variabler är **inte** tillåtna!

- Globala konstanter är tillåtna!

- Placera egendefinierade datatyper och konstanter i en egen headerfil (`constants.h`)!

- Egendefinierade klasser är **inte** tillåtna!

- Använd `C++ 11` – kod! C-kod, såsom `printf()`, är **inte** tillåten!

Extrauppgifter

1. Implementera svårighetsnivå **3)** för signaturen!

2. Kontrollera att en person är unik. Två personer är lika om **förnamn**, **efternamn** samt **längd** är lika. Det skall inte spela någon roll huruvida namnen anges som gemener eller versaler. Om två personer inte är unika skall användaren ges möjlighet att antingen **ändra** eller **avbryta** inmatningen!

3. Kryptera data innan de skrivs till fil, och dekryptera när data läses från fil. Använd förskjutningskryptering (`rot`) och låt användaren ange krypteringsnyckel, d.v.s. hur mycket varje tecken skall förskjutas!

   OBS!! Om kryptering används gäller inte det filformat som angivits ovan, utan spara en krypterad person per rad och kryptera även `DELIM`!

Examination

Projektet ger något av betygen **A**, **B**, **C**, **D**, **E**, **F** eller **F(x)**. Detta blir också kursens slutbetyg!

Bedömningspunkter

- Kravspecifikation: huruvida kraven är uppfyllda!

- Källkod…
  … konsekvent typografi!
  … indentering!
  … beskrivande namngivning (variabler, konstanter, funktioner osv)!
  … kommentering av programkod!

- Struktur…
  … uppdelning i funktioner!
  … uppdelning i filer!

- Användarvänlighet…
  … informativa utskrifter!
  … enkelhet vid inmatning / menynavigering!

Bedömningsterminologi

Avvikelse	*Avsteg från kravspecifikationen!*
Anmärkning	*Brister som antyder kunskapsluckor. Omfattande brister kan i värsta fall leda till avvikelse!*
Påpekande	*Svagheter och sådant som kan förbättras. Omfattande svagheter kan leda till anmärkning!*

Betyg

E	Max 8 avvikelser!
D	Max 5 avvikelser!
C	Max 2 avvikelser!
B	Godkänd för betyg C samt godkänt på 2 av extrauppgifterna!
A	Inga avvikelser samt godkänt på alla extrauppgifter!

OBS!! Inlämning / modifiering av projektrepo som sker efter angiven deadline sänker betyget ett steg!

Redovisning

Lokala ändringar skall löpande sparas med lämpliga commit-meddelanden och inför redovisningen är det viktigt att dessa ändringar synkroniseras med repots ***remote origin***. Utför sedan en formell inlämning i Moodle, via avsedd inlämningslåda, tillsammans med en kommentar (lämpligen länk till repot)!

Programkoden skall innehålla ett fungerande program och vara layoutmässigt genomarbetad vilket innebär att *indragningar*, *beskrivande variabelnamn*, *extra radbrytningar* samt *kommentarer* använts för att göra koden lättläst!

Arbetet skall genomföras enskilt men ni uppmuntras att såväl diskutera som stötta varandra i kursens tillhörande kommunikationsplattform!