Laboration 3 – HI1024, Programmering, grundkurs, 8.0 hp

Dataingenjörsprogrammet, elektroingenjörsprogrammet och medicinsk teknik
KTH – Skolan för Teknik och Hälsa

Redovisning: Se Kurs-PM om hur redovisningen ska gå till. Läs och följ dessa instruktioner noga!

Den här laborationen är den viktigaste examinationen i denna kurs. Klarar man av att utföra denna självständigt har man de kunskaper som behövs för de fortsatta studierna (kom ihåg att inte hjälpa varandra med koden). Den är krävande och kommer att ta er mycket tid och möda men när ni är klara är ni programmerare, om än ej fullärda. Till er hjälp har ni sist i detta dokument en ledning. Det är nämligen vår erfarenhet att det just nu för de flesta är lite för tidigt att helt själv strukturera upp arbetet runt en sådan här större uppgift. Att använda den är dock frivilligt. Det är dock inte tillräckligt att få ihop ett program som utför uppgifterna utan det måsta vara bra programmerat. Detta betyder först och främst att det bra uppdelat med hjälp av funktioner och att det använder arrayer och structar där så är lämpligt. Inga globala variabler får förekomma. Vidare ska ni välja bra variabel- och funktionsnamn. Kommentarer i koden bör det finnas en del men ni behöver inte överjobba denna del. Kommentera ganska lite och fokusera istället på att koden i sig blir lättläst.

Uppgiften

I denna laboration skall du skapa en databas för hantering av matcher i en liga (typ fotboll). Varje match skall representeras av en struct och innehålla speldatum, hemmalag, bortalag samt resultat (databasen hanterar endast matcher där resultatet består av antal gjorda mål för hemmalaget och bortalaget, t.ex. 4 – 2). När programmet startar får användaren ange ett filnamn och därefter via en meny möjlighet att utföra följande handlingar så länge användaren vill:

Registrera nya matcher
Skriva ut alla matcher
Sortera matcherna
Söka efter matcher
Avregistrera en match
Presentera statistik
Avsluta

Uppstart och avslut

När programmet startar får användaren ange en fil. Finns filen skall programmet läsa in data från filen och spara matcherna i en array av structar (glöm inte att kontrollera att de får plats). När programmet avslutar ska det skriva ut alla matcher i arrayen till filen. Filen skall vara en textfil (absolut ej binärfil) och får förutom matchdata också innehålla information om antalet matcher. Om fil saknas skall programmet börja med en tom array. När programmet avslutas ska det skapa en textfil med det tidigare angivna namnet och där spara ner matcherna i arrayen så att de kan läsas in nästa gång programmet körs. Observera att all filhantering sker vid programstart och vid avslut. Däremellan hanterar programmet endast matcherna i den interna arrayen.

Registrera nya matcher

Här har man möjlighet att registerar nya matcher. För varje match anger användaren datum (20140521), hemmalag, bortalag, mål gjorda av hemmalag och mål gjorda av bortalag. Efter att användaren registrerat en match får denne direkt möjlighet att registrera en ny match. Om användaren trycker enter utan att ange datum avslutas registreringen och programmet presenterar åter huvudmenyn. Det är viktigt att vid varje ny registrering kontrollera att arrayen inte är full. Om den är det skall användaren få ett felmeddelande och sedan återgår programmet till huvudmenyn.

Skriva ut alla matcher

När användaren väljer detta alternativ skrivs alla matcher ut i den ordning de är lagrade i programmet och sedan kommer huvudmenyn upp igen.

```
Datum      Hemma      Borta      Resultat
_____________________________________________
20140518   AIK        DIF        9-0
20140602   NIF        BP         3-1
20130912   HIF        GAIS       0-0
```

Sortera matcherna

Här får man välja på att sortera matcherna efter datum eller hemmalag. Beroende val sorteras matcherna i den array de lagras i och programmet återvänder till huvudmenyn. För att se resultatet av sorteringen får man i huvudmenyn välja att skriva ut alla matcher.

Söka efter matcher

Här ska användaren kunna söka efter matcher på datum eller lagnamn. Vid sökning på datum anger användaren ett datum och får upp en lista på alla matcher som spelades detta datum. Vid sökning på lagnamn ska användaren få upp en lista på alla matcher där ena laget innehåller söksträngen i sitt namn. En sökning på IF skulle då kunna ge följande lista:

```
Nr         Datum      Hemma      Borta      Resultat
_____________________________________________________
4          20140518   AIK        DIF        9-0
8          20140602   NIF        BP         3-1
12         20130912   HIF        GAIS       0-0
```

En sökning på datum skulle se liknande ut men innehålla samma datum på varje rad. Observera första kolumnen. Där står matchens plats i den array programmet använder för intern lagring. Mycket snyggare vore att numrera resultatet av sökningen 1, 2, 3 osv. Det får du också hemskt gärna göra om du vill. Om du gör det kommer funktionen avregistrera att bli något svårare att programmera.

Du får själv välja hur du hanterar stora och små bokstäver när du söker på lag och programmet behöver inte kunna hantera å, ä eller ö.

Avregistrera en match

När man väljer detta alternativ skall man först få möjlighet att göra en sökning för att sedan med hjälp av numret i sökningsresultatlistan välja vilken match som skall avregistreras. Detta skall ske utan återhopp till huvudmenyn. Man får bara möjlighet att avregistrera en match. Vill man avregistrera flera matcher får man välja alternativet flera gånger i huvudmenyn. Dialogen för alternativet bör likna nedanstående:

```
Du har valt att avregistrera en match. Vill du soka efter match pa
datum (1) eller pa namn (2)? 2
Ange sokstrang: IF


Nr          Datum       Hemma       Borta       Resultat
_____________________________________________________________
4           20140518    AIK         DIF         9-0
8           20140602    NIF         BP          3-1
12          20130912    HIF         GAIS        0-0

Vilken match vill du avregistrera? 8
```

Efter det sista valet återgår programmet till huvudmenyn.

Det är viktigt att du inte skapar hål i din array när du tar bort en match. Det naturligaste sättet att undvika hål är att du flyttar upp alla matcher under den avregistrerade ett steg.

Presentera statistik

Om du redovisar denna laboration under något av tillfällena under kursens gång eller under ordinarie tentamenstillfälle (p1 för data och elektro) ska du hoppa över delar av denna uppgift beroende på tidigare resultat enligt nedan. Observera att om du redovisar vid ett senare tillfälle måste du alltid redovisa hela denna del.

Presentera statistik för ett lag

-hoppa över detta om du är godkänd på laboration 1

Här ska användaren få ange ett lag varvid programmet antingen skall svara att laget saknas i databasen eller presentera hur många poäng laget har (3-vinst, 1-oavgjort, 0-förlust).

Presentera en tabell

-hoppa över detta om du är godkänd på inlämningsuppgifterna

Här presenteras en lista på alla lag som ingår i databasen.

Presentera en tabell med poang

-hoppa över detta om du är godkänd på laboration 2

Här presenteras en lista på alla lag som ingår i databasen med respektive lags poäng.

Om du inte behöver göra någon av ovanstående uppgifter kan du helt hoppa över detta alternativ i huvudmenyn. Om du bara behöver göra en av uppgifterna kan man komma direkt till denna funktion när man väljer presentera statistik. Om du behöver göra två eller tre av uppgifterna skall användaren få upp en meny där de väljer mellan dessa efter att de valt presentera statistik i huvudmenyn.

Redovisning

Vid redovisningstillfället skall du ha en fil med minst 20 matcher redo att läsa in. Dessa bör vara någorlunda realistiska så att man kan pröva de olika funktionerna på ett bra sätt. Under programmeringen torde man behöva en dylik fil så det borde inte vara något extra arbete att ha en sådan.

Ledning

Först observerar vi att redan instruktionen ger oss följande programflöde:
1. Programmet börjar med att läsa in data från fil till en array av structar (om filen finns). Programmet stänger sedan filen.
2. Programmet låter användaren välja från huvudmenyn tills denna väljer avsluta. Alla ändringar görs till arrayen. Ingen filhantering utförs.
3. När användaren väljer avsluta skrivs arrayen ut till fil och programmet avslutar.

Nu ska vi dela upp programmet och börja programmera de olika delarna. Här kan man göra många olika val. Nedan följer ett förslag som gör det någorlunda enkelt att testa programmet vartefter. Börjar ni med labben innan ni lärt er någon del kan ni välja en annan ordning men får då lägga lite mera tid på att kunna testa koden.

Arbetsordning

1. Skriv en meny-funktion som du anropar från huvudprogrammet. Denna anropar i sin tur funktioner för de olika alternativen. Börja med att kontrollera att det hela kompilerar och kör utan funktionsanrop. Sedan lägger du till funktionsanrop till funktioner som bara skriver ut en rad med sitt namn. Se också till att man efter ett funktionsanrop får möjlighet att göra ett nytt menyval men att programmet avslutar när man väljer avsluta. Sådär nu har du ett fungerande program som vi kan bygga på bit för bit.

2. Definiera din match-struct och deklarera en array av sådana structar i main. Sätt storleken på arrayen till 1000 med hjälp av #define. Deklarera och initiera i main också en variabel som ska hålla reda på antalet matcher vi har i vår databas (nedan kallar jag denna antal). Skicka nu med arrayen och en pekare till antal till meny-funktionen och se till att den kan ta emot dessa. Vi skickar här en pekare till antal efter som vi behöver kunna ändra variabelns värde. Se nu till att menyfunktionen kan ta emot parametrarna och kompilera sedan och testkör.

3. Skriv nu funktionen för att registrera nya matcher. Även denna behöver arrayen och en pekare till antal (håll tungan rätt i mun med pekarna nu och läs på om du behöver). Använd tillfälliga printsatser för att se att den verkar fungera. Skriv sedan funktionen skriv ut som till att börja med loopar igenom arrayen och väldigt enkelt skriver ut informationen i structarna. Denna behöver arrayen och variabeln antal. Du kan nu köra programmet och lägga till varor och sedan välja skriv ut och se att det fungerar. Förfina nu utskrifterna och rätta eventuella buggar.

4. Vi har nu ett fungerande program som gör något och en funktion skriv ut som ger oss möjlighet att se vad som händer. Vi ska här nu välja att börja jobba med filhanteringen. Anledningen är att det blir jobbigt att skriva in massa matcher varje gång man vill testa sina andra funktioner. Är du inte riktigt redo för filhantering än kan du hoppa detta just nu men får då skriva in matcher vid varje testkörning. Vi börjar med funktionen avsluta. Skriv denna så att den skriver ut all information i alla structar i vår array till en textfil. Låt den skriva antalet structar överst i filen så blir det lättare att läsa in senare. Namnet på filen kan du just nu hårdkoda. Öppna filen i en texteditor och kontrollera att det verkar fungera.

5. Det är nu dags att skriva initiera funktionen som ska anropas en gång i början av main.

   Det vore här naturligt att läsa in antalet matcher i filen och sedan sätta arrayens storlek t.ex. till det dubbla. Pga att vi inte ännu gått igenom allokering blir detta krångligt och svårt att göra i funktioner. Av den anledningen är det mitt råd att du behåller en fast storlek på arrayen (i nuläget 1000 matcher) även om detta känns lite primitivt. Vi kan då få en bra uppdelning av programmet i funktioner (om du bemästrar pekare till arrayer och allokering får du gärna allokera minne beroende på antal matcher i filen).

   Denna funktion skall enligt instruktionen fråga efter filnamn och om det finns en sådan fil lagra alla matcher i filen i vår array. Om filen saknas ska funktionen ta reda på filnamnet som sedan skall användas i avsluta. Ändra nu också funktionen avsluta så att denna skriver till korrekt fil.

6. Nu har vi kommit så långt så att jag tror ni är redo att själva välja i vilken ordning ni jobbar med övriga alternativ. Kom ihåg att kompilera och testköra ofta. Ett sista tips: Kolla in funktionen strstr när ni ska jobba med sök-funktionen.

   Lycka till!

   Nicklas, Jonas och Anders