Laboration

MasterMind

Henrik Bäck, 850611-6253
Anders Ellvin, 851013-6255
Civilingenjör IT

Programmeringsteknik
Nils Dåverhög
Christer Andersson

Inledning

Denna laboration gick ut på att skapa ett MasterMind med hjälp av det som har avhandlats i kursen. MasterMind är ett gammalt spel, det fanns som fick-spel i kalkylatorformat i mitten och slutet av 70-talet.

I denna rapport kommer det förklaras hur ett liknande spel har konstruerats i programspråket C++. Det kommer även tas upp hur projektet gått och vilka problem som uppstått under tiden som projektet har pågått. Till rapporten hör även två bilagor. En med hela programmets kod samt en med diagram över hur programmet fungerar.

Innehållsförteckning

1. Antaganden

Ett antal antagande gjordes vid skrivning av funktioner för att mata in gissningar till programmet. Detta antagande bestod i att en gissning inte kan vara längre än 99 tecken (100 inklusive strängsluttecken) långt.

Ett annat antagande var att det hemliga talet inte får innehålla dubbletter och därmed inte innehålla fler än 10 siffror (0-9) och att ett imatat värde aldrig kan innehålla ett null-tecken (\0).

Antal gissningar får inte heller överstiga 32 767 stycken, då detta resulterar i orimliga värden.

2. Översikt

MasterMind är ett spel. Detta spel går ut på att datorn slumpar fram ett hemligt tal, detta tal kan variera i längd från olika versioner av spelet. Efter att det hemliga talet har genererats så gäller det för användaren att gissa sig fram till vilket tal datorn har skapat. Genom att spelet lämnar vägledning i gissningen gäller det att på så få försök som möjligt lista ut talet. Talet kan inte innehålla två likadana siffror, som 8212.

Programmet består i stor av tre delar. Dessa är, spela MasterMind, visa HighScore samt att visa en viss hjälp om hur spelet går till.

Spela MasterMind

Denna del har hand om själva spelandet. Här slumpas fram ett hemligt tal och låter sedan användaren gissa vilket detta hemliga tal är. Genom en vägledning som skrivs ut efter varje gång gissningen har skett kan man lista sig fram till viket det hemliga talet är.

Efter avslutad spelomgång kontrolleras det antalet försök man gjort mot den aktuella HighScore-listan. Är den aktuella användarens resultat bättre än någon av de som redan ligger i HighScore så lägg användaren in före denna/dessa användare.

HighScore

Denna del av spelet visar en lista över de bästa spelarna. Dessa läses in vid programstart. Den information som visas är namn på spelaren, hur många poäng spelaren har samt vilken tid som användaren hamnade på HighScore-listan.

Hjälp

Denna del av spelet skriver ut fullständiga regler för spelet på användarens skrämbild. Hjälpen är sådan att användaren kan förstå programmet struktur och tolka den vägledning som ges av programvaran.

3. Detaljerad beskrivning

Öppna HighScore-listan

När programmet startas körs funktionen, lasFranFil(). Denna funktion har till uppgift att, om möjligt, hämta in HighScore-listan från fil.

Först skapas objektet, lasFil, och försöker öppna filen HIGHSCORE_DATA, som är bestämd i headerfilen. Om detta misslyckas kommer en text, ”Det gick inte öppna filen…” att skrivas ut på skärmen. I annat fall så kommer funktionen att läsa in varje position för sig och lagra den i HighScore-vektorn tills inga fler poster finns i filen, (EOF). Genom en while-sats stegas varje position i filen fram, då man vet storleken på varje post (sizeof(score)). Därefter stängs filen.

Spela MasterMind

När användaren har valt att spela så körs först och främst funktionen spelaMM(). Härifrån anropas flera olika funktioner som sköter bland annat det hemliga talet, ta emot gissningen samt kontroll av gissning.

Hemligt tal

När användaren väljer att denne vill spela ett parti MasterMind slumpas först ett hemligt tal fram. Denna slumpning görs funktionen slumpaTal(). Här finns det möjlighet att slumpa fram ett tal som är tio siffror långt. Det hemliga talet kan inte innehålla dubbletter. Med hjälp av en slumpfunktion slumpas en positionen fram i en temporär vektor. Denna temporära vektorn innehåller från början siffrorna 0 till och med 9.

Efter att den framslumpade positionens siffra, alltså den siffra som ligger i vektorn, har kopierats till det hemliga talet ersätts talet på den slumpade positionen med det högsta talet i den temporära vektorn. Därefter minskas en variabel innehållande antalet kvarvarande oanvända siffror i den temporära vektorn.

Denna procedur upprepas ända tills det hemliga talet har så många siffror som det skall ha. Detta bestäms av en konstant som ligger inkluderad i programmets headerfil.

Ta emot en gissning

Efter att det hemliga talet skapats startar själva spelet. Nu ges användaren en chans att mata in sin gissning. En gissning måste bestå av ett bestämt antal siffror. Genom funktionen taEmotGissn() får användaren mata sin gissning. Här ges även tillåtelse att mata in ”EXIT” för att avbryta själva spelomgången.

Det inmatade värdet kontrolleras först mot texten ’EXIT’ och om denna kontroll returnerar ’true’ kommer hela funktinen returnera 0. Denna retur avslutar i sin tur spelomgången av MasterMind. Om detta inte är fallet kommer gissningen att kontrolleras så att den endast innehåller siffror. Om det inmatade värdet innehåller annat än siffror kommer användaren att få mata in ett nytt värde som sedan kontrolleras mot texten ’EXIT’ och om även denna inmatning endast innehåller siffror. Detta upprepas tills användaren matat in ett värde enbart bestående av siffror eller ’EXIT’.

Efter detta kommer en for-sats som sätter den inskickade vektorns alla tillgängliga platser till ett orimligt värde, i detta fall värdet -1. Detta görs för att senare kunna kontrollera så att man matat in ett korrekt antal siffror.

Nästa for-sats kopierar över gissnings-vektorns värden till en vektor som är inskickad via funktionen taEmotGissn() som referens. Om någon utav siffrorna skulle vara ett negativt tal måste användaren mata in en ny gissning. Detta medför också att ifall man matar in ett för kort värde så kommer användaren att få mata in en ny gissning.

Slutligen kontrolleras så att det inmatade talet är korrekt antal siffror långt. Detta görs genom att före inmatning så sätts ett strängslutstecken (\0) in i vektorn, där användarens gissning senare lagras, en position längre bort än så många siffror man får mata in. Genom att kontrollera att detta strängslutstecken finns kvar på sin ursprungliga plats kan man fastställa att det inmatade värdet är korrekt antal siffror långt.

Om alla delar i funktionen har lyckats så kommer funktionen att returnera ett värde som är skiljt från 0. Om funktionen returnerar 0 innebär detta att användaren vill avbryta spelet och återgå till menyn.

Kontrollera gissning

När programmet tagit emot en korrekt gissning från användaren kontrolleras denna mot det hemliga talet. Detta görs i en funktion, kontrollera().

Först skapas en ’vägleднings’-vektor. Denna har till uppgift att lagra en vägledning för användaren. Genom den första for-satsen sätts all vägledning till fel, alltså ett ’_’. Detta innebär att följande funktioner endast behöver kontrollera ifall ett tecken skall vara med fast på annan plats eller ifall alla tecken är korrekta.

Den andra for-satsen i funktionen kontrollerar om det första tecknet i vektorn, innehållande det gissade talet, finns i någon av de andra positionerna, i samma vektor. Om så är fallet sätts motsvarande position i 'vägledning's'-vektorn till ett'S'. Därefter kontrolleras om denna siffra finns på rätt plats, om så är fallet kommer motsvarande position i 'vägledningsvektorn' att sättas till ett 'R'. Detta görs för varje position i vektorn med det gissade talet.

Slutligen kommer funktionen att returnera 'true' om användaren har gissat alla siffror rätt och 'false' om ett eller flera siffror är fel.

I spelaMM()
Tillbaka i funktionen spelaMM() kontrolleras om användaren vill avsluta, returvärdet från taEmotGissn() är satt till 0. Om så är fallet kommer spelaMM() avslutas och gå till menyn. I annat fall kontrollerar spelaMM() om användaren har gissat rätt, returvärdet från kontrollera() är 'true', och kommer därmed försöka att lägga till spelaren i HighScore.

HighScore (kontrollera / lägga till)

När spelomgången är avslutat och kontrollera() returnerat 'true' så kommer funktionen addScore() att försöka lägga till användaren i HighScore-listan om användaren har tillräckligt låga poäng.

Genom den första for-loopen kommer HighScore-listan itereras igenom och ta reda på vilken placering som det nya HighScore-värdet skall lagras. Programmet kontrollerar så att denna plats inte överstiger maximalt antal platser i HighScore-listan. Om så är fallet kommer en text, "Du kom inte in på HighScore-listan…", att skrivas ut på skärmen.

Om användaren har tillräckligt låga poäng för att få komma in på HighScore-listan så flyttas alla de andra användarna, med högre poäng, ett steg nedåt i listan. Därefter får användaren mata in sitt namn och detta lagras tillsammans med tid och poäng i listan.

Näst sist kommer det totala antalet personer i HighScore-listan att ökas med ett om det inte är så att HighScore-listan redan är full. Är HighScore-listan full kommer den sista personen att "puttas" ut ur listan.

Slutligen kommer addScore() att returnera hur många personer det finns i HighScore-listan.

I spelaMM()
Ännu en gång tillbaka i spelaMM() så kommer toppen på HighScore-listan, som bestäms av en konstant i headerfilen, att skrivas ut för användaren. Om det inte finns tillräckligt många personer i listan, som konstanten i

headerfilen föreskriver, så kommer endast så många personer som för tillfället är inlagda att skrivas ut. Se mer under nästa rubrik.

Visa HighScorelistan

Funktionen showScore() skriver ut ett bestämt antal (totplats) placeringar med de bästa spelarna först. Till en början ställs rätt värden in för utskrift. Först vänsterställs funktionen setw() och därefter skrivs ett tabellhuvud ut.

En for-loop försöker att iterera genom HighScore-listan för att skriva ut alla personer som finns inom intervallet. Om inget skrivs ut, alltså HighScore-listan är tom, kommer ’i’ att bli 0.

Därefter kontrolleras ifall ’i’ är lika med 0. Om så är fallet kommer texten ”Ingen har ännu spelat…” att skrivas ut på skärmen.

Rensa HighScore-listan

Genom funktionen highRens() så sätts referensparametern, ant, till 0. Därefter skrivs texten ”HighScore-listan har rensats…” ut på skärmen. Funktionen är till för att mata in variabeln som håller antalet poster i HighScore-listan och få den satt till 0 och samtidigt meddela användaren om att denne har rensat HighScore-listan. Genom att denna variabel blir satt till 0 kommer programmet ’tro’ att inte det finns några lagrade personer i vektorn för HighScore.

Avsluta

Denna funktion, avsluta(), skriver ut en tack-text till användaren och tackar denna för att hon har spelat.

Efter att avsluta() har körts så kommer programmet att stängas. Innan detta kommer HighScore-listan att sparas till fil.

Spara HighScore-listan

För att kunna hålla kvar HighScore-listan efter att programmet avslutats så måste HighScore-listan sparas till fil. Detta sköts av funktionen skrivTillFil(). Det första som sker i denna funktion är att ett objekt, skrivFil, skapas och öppnar filen som har bestämts av konstanten HIGHSCORE_DATA som förbestämts i headerfilen.

Om öppningen misslyckas så kommer en text skrivas ut på skärmen att filen inte gick att öppna för skrivning. I annat fall kommer en for-loop att spara varje post i HighScore-listan till filen och sedan stänga filen.

4. Problem

Under projektets gång uppstod mindre problem med programkoden och hur denna skulle skrivas för att ge bästa resultat.

Inmatning av gissning

När användaren matar in sin gissning så måste denna inmatning vara så kallad säker. Det ska inte vara tillåtet att mata in bokstäver eller tal som är för långa respektive för korta.

Genom en färdig funktion, isOnlyDigits, kontrolleras att det inmatade värdet från användaren bara innehåller siffror. Skulle inmatning innehålla annat än siffror ges användaren ett nytt försök att mata in. Denna procedur är tvungen att upprepas ända till användaren har matat in ett korrekt värde.

Därefter måste även inmatningen kontrollera så att den innehåller korrekt antal siffror. Det skall inte vara tillåtet att mata in tal som är för korta eller för långa i jämförelse med det hemliga talet.

Genom att föra in det imatade värdet till en temporär variabel där alla o-inmatade värden är initierade negativa så kontrolleras att alla siffror som kommer användas är positiva, skulle de inte vara detta så får användaren ett nytt försök att mata in sin gissning.

Referensanrop

Ett utav de mer udda felen uppstod under programmering med referensanrop i funktionen addScore(). I denna var ’totplats’ en refererad variabel och returnerades inte. Vid vissa tillfällen ändrades dess värde från 2 till 16 utan att detta stod i programkoden. Felet kunde inte avhjälpas med att förändra programkoden på annat sätt än att göra totplats till ett värdeanrop och i slutet av funktionen returnera dess nya värde.

Troligtvis uppstod detta problem för att exikveriningen skedde i cygwin, dock är detta inte konstaterat då programkoden ändrades så att problemet löstes snarast.

HighScore-listan

Problem uppstod då programmet har försökt öppna filen där HighScore-datan har lagrats. Då programmet misslyckades öppna filen och man därpå försökte visa innehållet i HighScore-listan så resulterade detta i att denna visade felaktiga värden.

Problemet låg i att referensvariabeln vars innehåll styrde antalet platser i HighScore-listan inte initierades om filen inte gick att öppna.

Fler problem uppstod när HighScore-vektorn skulle skrivas till. Funktionen skrev utanför vektorns utrymme, alltså i annan del av minnet vilket resulterade i konstiga krascher, detta löstes genom att skapa en plats större HighScore-vektor.

Ett annat problem som uppstod var att när användaren skrev in sitt namn, alltså när denne kom in på HighScore-listan. Då ett för långt namn lagrades i structen skrevs användarens poäng över. Detta löstes genom att en ny buffer på 100 tecken skapades som namnet först sparades i. Sedan kopierades det antal tecken som var möjliga till namn-vektorn i structen, då skrevs poängen inte över.

7. Diagram

I början av projektet gjordes ett par diagram, se bilaga B. Dessa var över hur programmets struktur var tänkt att fungera. Efter hand som projektet har fortlöpt så har dessa diagram ändrats en aning på grund av att funktioner som inte tidigare var påtänkta var tvungna att implementeras.

Det som har förändrats är det översiktliga diagrammet, se bilaga B, där det har varit nödvändigt att införa två stycken sekvenser. En i början av diagrammet och en i slutet. Dessa behövdes då all data från HighScore-listan skulle sparas till fil. Här har även en selektion som har hand om att kontrollera ifall användaren har valt att rensa HighScore-listan lagts till.

Diagrammet över ’Visa HighScore-listan’, ModVisaHigh i bilaga B, har skrivits om. Inläsning av HighScore-filen görs inte i denna funktion utan i programstarten. Även delen som har hand om att rensa listan har tagits bort från detta diagram då denna funktion flyttats ut till menyn för programmet. Detta medförde att i diagrammet över menyn har ett menyalternativ lagts till, vilket har hand om rensningen av HighScore-listan. Diagrammet har också kompletterats med en sekvens som talar om ifall det inte finns några lagrande HighScore-värden. Diagrammet gjordes samtidigt om så att det blev mer detaljerat och utförligt än tidigare.

Vad det gäller att lagra HighScore så har diagrammet för denna del, ModLagraHigh i bilaga B, redigerats så att den delen av diagrammet som hade hand om att spara HighScore-listan till fil tagit bort. Denna del hanteras som sagt vid programstart och programslut.

Diagrammet över spelet, ModMaster i bilaga B, har ändrats så sett att om användaren avbryter spelomgången så skriv det hemliga talet ut. För att lösa detta införde vi en selektions-ruta där det hemliga talet endast skrivs ut om användaren avbryter omgången, i annat fall har användaren gissat rätt och behöver då inte denna information.

6. Sammanfattning

Projektet har gått allmänt bra! Genom att följa inlämningsdatum för varje del labb låg alltid projektet alltid lite före de aktuella datumen. Fastän inte MasterMind var en obligatorisk inlämning, vad det gällde delmål, så fick det detta projektet att alltid vara i fas och inte skapa några större problem fram emot sluttampen.

Den lösningen som framställts har varit den som känts bäst. Det finns säkerligen många förbättringar som kan göras i programmet. Den lösning som har valts är den som tycks gå mest i stil och följa de moment som gjorts i kursen, DAV A07 vid Karlstads Universitet.

En alternativ lösning på problemet att spara ned HighScore-listan till en fil vid avslutning skulle kunna vara att spara ned listan vid varje tillfälle som ändrar i HighScore-listan. Detta skulle spara HighScoren även om programmet avslutas anormalt. En nackdel med detta skulle dock vara att om programmet skulle köras på en diskett skulle disketten slitas ut fortare och till slut bli obrukbar. Istället har vi satsat på säkerhet så att inte programmet skall avslutas anormalt eller hänga sig.

Något som skulle kunna gjort annorlunda var att anteckna varje del och varje delproblem mycket mer noggrant. Detta hade förenklat skrivningen av rapporten. Något man bör dra en lärdom av.

Eftersom även detta projekt med MasterMind har följt de deadlines som varit för övriga labbar inom kursen, DAV A07, så har varje delmoment kunnat göras väldigt fort. Genom att samköra lite av både MasterMind och den ordinarie laborationen så har arbetet gått fortare än vad det annars kanske skulle ha gjort. I genomsnitt har antagligen två timmar per delmoment lagts ner, frånsett den sista delen (sammansättningen) som tagit betydligt mer tid. Denna del har tagit runt 4-5 timmar på grund av de problem som uppstod med vissa funktioner när de implementerades i programmet.

Den största lärdomen man kunnat dra av detta projekt var att under tiden man löste problem och skrev kod tränade man sig även mycket inför kommande tenta.

7. Referenslista

Internetresurs, www.cplusplus.com

8. Bilagor

Bilaga A - Programkod
Bilaga B - Diagram över Program