PROJEKTRAPPORT

INTRODUKTION TILL PROGRAMMERING

Utförd av: Andreas Johansson Dis17

Datum: 2017-12-13

Innehåll

1 Introduktion

1.1 Bakgrund

Vi har fått ett projekt som handlar om att skapa ett program som kan hantera magiska kvadrater. Summan av varje rad, kolumn och diagonal ska vara trettio. I kvadraten ska alla hexadecimala siffror finnas och de får inte förekomma mer än en gång. Programmet ska kunna läsa in kvadrater från terminalen och filer. Kvadrater som har lästs in ska kunna testas och se om de är magiska kvadrater. Det ska finnas med ett spel där spelaren får fylla i en kvadrater så att den blir magisk. Programmet ska även kunna generera alla delmängder som finns. En delmängd består utav fyra stycken olika hexadecimala siffror där summan blir trettio. Om siffrorna i delmängden byter plats räknas det fortfarande som samma delmängd. Efterföljande till det ska även programmet kunna generera ut ifrån delmängderna alla kvadrater där varje rad eller kolumn har summan trettio och där hexadecimala siffror inte får förekomma mer än en gång.

1.2 Målsättningar

Målsättning inför detta projekt var att kunna klara alla uppgifter med så snygg, effektiv och praktisk kod som möjligt.

2 Analys av projektuppgiften

Projektet består av att vi ska skapa ett program som kan hantera magiska kvadrater. För att kvadraten ska ses som magisk ska summan av varje rad, kolumn och diagonal vara trettio. Varje hexadecimal siffra måste finnas med i kvadraten och får endast förekomma en gång. Programmet ska kunna läsa in kvadrater från terminalen och från filer. Dessa kvadrater som har lästs in till programmet ska kunna testas och ses om de är magiska. Det ska även finnas med ett spel där ett antal slumpmässigt valda (beroende på svårighetsgraden) siffror tas bort ur en magisk kvadrat och spelaren får då fylla i dessa så att kvadraten blir magisk. Spelaren kan välja att ta bort en till åtta siffror. Programmet ska kunna generera alla delmängder som finns. En delmängd består utav fyra stycken hexadecimala siffror där varje siffra endast får förekomma en gång och summan av siffrorna ska bli trettio. Om siffrorna byter plats i delmängden räknas det ändå som samma delmängd. Av de delmängder som har genererats ska programmet kunna generera kvadrater där varje kvadrat består av alla hexadecimal siffror och där summan av varje rad eller kolumn blir trettio.

Projektet har slagits isär till varje uppgift, varje uppgift har slagits isär till så generella funktioner som möjligt som uppgiften behöver. Plan från början vara att utgå simpelt och få allting att fungera. Därav har delar tagits bort som till exempel att introducera binärträd för lagring av de variabler som skapas till kvadraterna, mer avancerade sökalgoritmer för att vid snabbare och effektivare körning kunna se om en kvadrat är magisk eller om en viss delmängd redan existerar, att kunna ta bort en specifik variabel. Så många funktioner som möjligt har skapats så att de hanterar enbart en funktionalitet. Simpelhet genomsyrar projektet för att projekt ska kunna bli klart inom tidsramen.

3 Genomförande

Arbetat har utförts utefter planen som beskrivs i analysen, där varje uppgift har delats isär till funktioner som alla sköter en specifikt funktionalitet. Koden är skriven så simpelt, effektivt och snyggt som möjligt. Att hantera flera kvadrater inom programmet var något som inte programmerades in från början, utan en helt ny variabeltyp implementerades med funktioner till så att kvadraterna kunde skapas och tas bort. Den svårighet har hanterats bra. Ett relativt komplext system har ändå kunnat implementerats på ett effektivt och simpelt sätt. Eftersom saknaden av avancerade sök och lagringsalgoritmer har framförallt genererandet av delmängder och de kvadrater som genereras av delmängderna hanterats på ett komplext och ineffektivt sätt. Genererandet av delmängder och dess kvadrater fungerar, men implementationen och svårigheten med det har inte hanterats på ett bra sätt.

4 Resultat

Koden är skriven i programmeringsspråket C. Menyn i programmet har delats upp i en switch – case sats i huvudfunktionen och funktioner för varje val i en separat .c fil på grund av att få enkla och överskådliga funktioner som lättare kan hanteras vid till exempel avlusning.

Kvadrater hanteras som fält på grund utan att de är lättare att hantera än matriser vid bland annat sökningar. Varje kvadrat lagras i en struct-datatyp. Denna struct-datatyp innehåller fältet där kvadraten lagras i, ett fält med namnet på kvadraten och två stycken heltal som specificerar längden på fälten. Programmet kan max hanterar tvåhundra stycken lagrade variabler.

Varje kontrollfunktion hanterar endast en funktionalitet av kontrollen om kvadraten är magisk. Som till exempel kontroll av att summan av varje rad bli trettio. Funktionen ”kontroll_kvadrat” samlar ihop och alla kontrollfunktioner till en funktion ifall man ska köra alla kontroller.

Läsningen hanteras av funktionen ”read_line” där användaren specificerar var användaren vill läsa till och ifrån. Hanteringen av var användaren vill läsa ifrån och om användaren enbart vill läsa heltal hanteras av andra funktioner. Vid inläsning av kvadrater görs kvadraterna om från tecken till decimala heltal. På grund av detta görs tecken som till exempel b eller B om till heltalet 11 istället för att följa ASCII-standarden. Detta beror återigen på att det är lättare att hantera decimala heltal än hexadecimala tecken.

Utskrift av kvadraten hanteras av flera olika funktioner beroende på om användaren vill skriva ut dem till terminalen eller till en specifik fil. På samma sätt som hos inläsning görs innan utskrift en konvertering från decimala heltal till hexadecimal tecken så att utskriften blir korrekt. Dock används enbart stora bokstäver vid utskrift. Utskrift till fil skriver ut kvadraterna som fält. Utskrift till terminalen skriver ut kvadraterna som kvadrater.

För att kunna göra om kvadrater som läses in till programmet till fält används en funktion som plattar ut kvadraterna till fält.

Två funktioner som kopierar fält har skapats. Ett som kopierar heltalsfält och ett annat som kopierar tecken fält. Varför egna kopierings funktioner har skapats är på grund utav att få mer insikt i hur dess fungerar för att kunna anpassa dem till programmet.

Det finns fyra stycken funktioner som hantera den struct-datatyp som har skapats åt kvadraterna. En funktion som används då en nya kvadrat läggs till. Denna funktion lägger in kvadraten i en oanvänd struct-datatyp. En funktion som släpper minnet på den pool av struct-datatyper som programmet skapar i början av körningen. En funktion som håller koll på antalet struct-datatyper som används av programmet och sist en funktion som letar rätt på den efterfrågade struct-datatypen i poolen av struct-datatyper.

En funktion har skapats som nollställer en matris. Med nollställning menas att varje del av matrisen får ett värde som är högre än femton. Detta på grund av att noll ska vara med i kvadraterna och då måste kunna jämföras.

Delmängder genereras genom fyra stycken nivåer av slingor. Detta på grund utav att ett tal tas fram från varje slinga och då bildar en delmängd. Inuti sista slingan testas om varje siffra är unik i delmängden, delmängdens summa blir 30 och att delmängden inte tidigare har genererats. Om dessa krav uppfylls skrivs då delmängden till en matris där delmängderna lagras.

Kvadraterna som genereras utifrån delmängder genereras på ungefär samma sätt som själva delmängderna. Genom fyra nivåer av slingor där fyra stycken delmängder plockas ut och bildar en kvadrat. Denna kvadrat testas genom att alla siffror finns med, varje siffra är unik och att kvadraten inte redan existerar. Om dessa krav uppfylls skrivs kvadraten till en matris med kvadrater. Precis som i resterande program hanteras även kvadraterna här som fält.

Minnet till matriserna för både delmängderna och dess kvadrater hanteras av två separata funktioner. En som skapar minne och en som släpper minne.

5 Diskussion

Det jag har lärt mig via det här projektet och denna kursen är att ta isär problemet i dess beståndsdelar samt få ett mer flyt i programmeringen.

Om det hade funnits mer tid skulle jag önskat att introducera balanserade binärträd för lagring av de struct-datatyperna som jag använde till kvadraterna. Genom balanserade binärträd hade det varit effektivare och lättare att skapa sökalgoritm och algoritmer som kan dynamiskt lägga till och ta bort specifika kvadrater.

En önskning hade varit att ha gjort en djupare undersökning i hur jag skulle kunna ha gjort funktionerna för generering av delmängderna och dess kvadrater mycket mer effektiva. Då genom att försöka minska ner slingorna, framförallt där de går inuti varandra.

Trådprogrammering hade varit perfekt för detta projekt, då flera beräkningar i detta projektet kan köras separat utan att de är beroende av varandras beräkningar.

6 Referenser

C från början Jan Skansholm