Inlämningsuppgift – Sudoku

Kortfattad beskrivning

Uppgiften går ut på att skriva ett program för att lösa sudokuproblem. Användare presenteras för en utgångsställning med några siffror ifyllda och försöker sedan komma fram till (den unika) lösningen genom att skriva in siffror i resterande rutor. De ursprungliga siffrorna skall inte gå att ändra och det skall finnas en funktion för att avgöra om en lösning är korrekt. Programmet ska ha ett grafiskt användargränssnitt.

1 Problemet

Sudoku är ett ”pussel” som består av 9*9 rutor. Några rutor är från början ifyllda med siffror mellan 1 och 9. Uppgiften går ut på att hitta en lösning som uppfyller följande krav:

- Varje ruta är fylld med en siffra mellan 1 och 9.
- Varje rad, vågrät och lodrät, innehåller siffrorna 1–9.
- Varje ”region” innehåller också siffrorna 1–9. Med region avses här en grupp av nio rutor motsvarande de grupper som färgats ljusa repektive mörka i figur 1.
- Ingen av siffrorna förekommer mer än en gång per rad, kolumn eller region.

Exempel finns i figur 1 och 2. Figur 1 visar vilka rutor som från början var ifyllda och figur 2 visar en lösning.

		8			9		6	2
								5
1		2	5					
			2	1			9	
	5					6		
6							2	8
4	1		6		8			
8	6			3		1		
						4		

Figur 1: Sudoku med ett antal från början ifyllda rutor.

5	4	8	1	7	9	3	6	2
3	7	6	8	2	4	9	1	5
1	9	2	5	6	3	8	7	4
7	8	4	2	1	6	5	9	3
2	5	9	3	8	7	6	4	1
6	3	1	9	4	5	7	2	8
4	1	5	6	9	8	2	3	7
8	6	7	4	3	2	1	5	9
9	2	3	7	5	1	4	8	6

Figur 2: Lösning till Sudokun i figur 1.

2 Programskrivning

Programmet består av två delar. En del som utgör en modell av sudokun och en del som utgör ett grafiskt användargränssnitt.

Modellen kan till stor del lösas och testas först. Därefter kan man implementera det grafiska användargränssnittet. Innan man har ett sådant kan man testa modellen, t.ex genom att programmet läser in filer för att skapa sudokurutnät med alla eller vissa rutor givna och sedan anropa den metod i modellen som avgör om problemet är löst/lösbart (dvs. ingen av siffrorna förekommer mer än en gång per rad, kolumn eller region).

2.1 Modellen för Sudoku

Ett Sudoku representeras lämpligen av en klass som har en heltalsmatris motsvarande sudokuns rutnät som medlemsvariabel.

I klassen ska det bl.a. finnas en konstruktor och funktioner för att sätta respektive hämta värdet för en ruta. Det skall också finnas en funktion för att sätta en ursprungsställning inläst från fil samt en funktion för att avgöra om en ruta ingår i ursprungsställningen eller ej.

Den viktigaste funktionen i klassen är den som avgör om problemet är löst/lösbart eller ej. Utgående från ett helt eller delvis fyllt rutnät ska denna funktion kunna meddela att ingen lösning finns.

2.2 Grafiskt användargränssnitt

Programmet ska ha ett grafiskt användargränssnitt programmerat i `wxWidgets`. Se figurerna 1 och 2 som exempel på hur det kan se ut. När programmet startar ska ett tomt rutnät med 9*9 rutor samt tre knappar visas.

Rutorna kan vara av typen `wxTextCtrl` med plats för ett tecken vardera. Dessa textfält placeras på en panel av typen `wxPanel` (som i sin tur placeras på ett fönster av typen `wxFrame`). För att få det önskade utseendet kan man använda sig av `wxGridSizer` för panelen. För att tydligt markera olika regioner kan olika bakgrundsfärg användas för textfälten.

Under Rutorna skall tre knappar placeras. En knapp (”New”) är till för att generera ett nytt sudoku. En (”Check”) är till för att kontrollera en lösning som användaren fyllt i. Den tredje knappen (”Clear”) ska användas om man ger upp och vill starta sudokut från början.

När programmet startar visas ett tomt rutnät. Man ska då inte kunna skriva något i rutorna. För ett textfält `tf` går det att ange att det inte ska vara editerbart på följande sätt:

```
tf.setEditable(false);
```

När användare trycker på den knapp som anger att ett nytt sudoku önskas, ska man från en fil läsa och skapa ett sådant. Mer om hur detta kan gå till finns nedan. Det sudoku som skapats ska visas i vyn. De rutor som då är ifyllda ska fortfarande inte gå att editera. De tomma rutorna, däremot, ska nu gå att editera. Man gör ett textfält `tf` editerbart på följande sätt:

```
tf.setEditable(true);
```

Användare skriver in en siffra i en ruta. Därefter kan man be att få se status på lösningen genom att trycka på knappen ”Check”. Knappen ”Clear” ska ta bort inmatade siffror från alla rutor.

I den eventhanterare som kopplas till textfältet läser man det ändrade värdet när inmatningen av fältet är avslutad (t.ex. när man går till ett annat textfält eller trycker på en knapp). Om det ändrade värdet är en siffra 1-9 så uppdateras modellen med det nya värdet. I annat fall visas (t.ex. med ett dialogfönster) att det finns felaktig data i textfältet. Ett alternativ sätt för felkontrollen är att koppla en `wxTextValidator` eller en `wxIntegerValidator` till textfälten. En sådan validator kan sättas upp så att bara går att mata in giltliga värden.

Programmet ska vara bekvämt att använda. Det ska inte krascha om användaren skriver in konstiga värden i rutorna.

3 Hantering av textfil i programmet

indata

Ett utgångsläge för ett sudoku kan skrivas in på textfil enligt följande mönster:

```
0 0 6 5 1 0 7 0 0
0 0 0 0 0 0 3 0 0
9 1 0 0 4 0 0 0 8
0 0 0 9 0 1 0 0 2
7 0 9 0 0 0 1 0 4
1 0 0 4 0 8 0 0 0
3 0 0 0 2 0 0 5 6
0 0 4 0 0 0 0 0 0
0 0 7 0 9 4 2 0 0
```

En rad på filen motsvarar alltså en rad i rutnätet. Nollor motsvarar rutor som inte är ifyllda. Flera sudokun kan skrivas efter varandra enligt samma mönster.

Hämta 4-5 sudokun från tidningar eller från nätet och skriv in på en fil. Filen placeras lämpligen i projektkatalogen.

Läsa indata från fil i programmet

Man kan läsa från en fil en rad i taget till en `string`-variabel med `getline`. Genom att sedan koppla en `istringstream` till den inlästa sträng kan man sedan läsa de 9 tal som står på denna. Med denna metod kan man få en kontroll att filformatet stämmet.

Val av sudoku på filen

Om man skrivit in ett antal sudokun på filen kan man, då användaren vill ha ett nytt, t.ex. göra så här:

- Beräkna antalet sudokun i filen (t.ex. genom att räkna antalet rader med `getline` – det skall vara 9 rader per sudoku).

- Bestäm genom slumpdragning vilket sudoku som skall visas
- Håll reda på vilka sudokun som visats så dessa kan undantas från slumpdragningen.
- Öppna filen och stega fram till positionen som motsvarar det valda sudokut och läs in sudokut från de 9 följande raderna. Att stega fram till rätt position kan t.ex. göras med rätt antal `getline`.
- Om alla sudokun på filen vistats så börjar man från ursprungsläget genom att ta bort informationen om vilka sudokun som visats.