**TRF**

**Tigerälskarnas Riksförbund**

Rapport

**Programmering 2 C#**

**NTI-Skolan**

**Johan Kämpe**<https://www.linkedin.com/in/johankampe><https://github.com/GoblinDynamiteer>

**2017-07-17**

Innehåll

[***1 Inledning*** 3](#_Toc488052648)

[**1.1 Syfte** 3](#_Toc488052649)

[**1.2 Noteringar** 3](#_Toc488052650)

[**1.3 Länkar** 3](#_Toc488052651)

[***2 Genomförande och resultat*** 4](#_Toc488052652)

[**2.1 Använd programvara och litteratur** 4](#_Toc488052653)

[**2.2 Avgränsningar och krav** 4](#_Toc488052654)

[2.2.1 Grundläggande funktionalitet 4](#_Toc488052655)

[2.2.2 Utökad funktionalitet 4](#_Toc488052656)

[**2.3 Metod** 5](#_Toc488052657)

[**2.4 Programmets funktion** 5](#_Toc488052658)

[**2.5 Flödesschema** 9](#_Toc488052659)

[**2.6 Pseudokod** 10](#_Toc488052660)

[**2.7 Skillnader från uppgift 3** 11](#_Toc488052661)

[**2.8 Programmets kod** 11](#_Toc488052662)

[2.8.1 Klassen Logbook 11](#_Toc488052663)

[2.8.2 Klassen Menu 12](#_Toc488052664)

[***3 Diskussion och slutsats*** 13](#_Toc488052665)

[**3.1 Förslag på förbättringar** 13](#_Toc488052666)

[**3.2 Val av algoritmer, variabler och datastrukturer** 15](#_Toc488052667)

[**3.3 Utvärdering av insats** 16](#_Toc488052668)

# 1 Inledning

## Syfte

Syftet med uppgiften är att skapa ett program i programspråket C#, som underlag finns en projektspecifikation med vissa krav som måste uppfyllas.

Utöver dessa krav får programmet skapas med fria händer.

## Noteringar

Skärmklipp i rapporten visar ibland en äldre version av programmet, vissa skillnader kan förekomma i utseende och text i det inlämnade programmet.

## 1.3 Lösenord

Programmet startar med ett loginfönster, där ett lösenord måste skrivas in för att komma vidare.



Figur loginfönster

Lösenordet sparas i en fil kallad *login.pwd*. Om denna fil av någon anledning saknas i den katalog där programmets exekverbara fil ligger används ett standardlösenord.

|  |  |
| --- | --- |
| Lösenord för att logga in i programmet | |
| TigrarÄger123! | Om filen login.pwd finns tillgänglig för programmet |
| 123 | Standardlösenord om login.pwd inte används |

Användaren har möjlighet att själv byta lösenord i programmet, om så önskas.

## Länkar

Uppgiften på GitHub  
<https://github.com/GoblinDynamiteer/P2CS/tree/master/course_exercises/project>

Ikoner som används i programmet  
<https://www.iconfinder.com/icons/381599/error_icon>  
<https://www.iconfinder.com/icons/285654/cat_icon>

# 2 Genomförande och resultat

## 2.1 Använd programvara och litteratur

Programvara

* Microsoft Visual Studio Community 2017 v4.7.02045
* Microsoft Word 2016 v1701
* Draw.io: <https://www.draw.io>

Litteratur

* Alishenas, T *Programmering 2 med C#* (ISBN 9789197420433)

## 2.2 Avgränsningar och krav

### 2.2.1 Kravspecifikation

Medlemmarna i TRF består av människor som på något sätt skaffat sig en tiger som husdjur. Ditt jobb här blir nu att skapa ett program som håller reda på namn och adress till medlemmarna och namn på tigern/tigrarna.

**Du får helt fria händer, bortsett från nedanstående punkter:**

* Programmet **ska** innehålla en databas, och det ska givetvis då gå att lägga till/ta bort medlemmar.
* Det **bör** någonstans finnas en länk till Tiger på Wikipedia, klickar man på den så ska webbsidan visas (<http://sv.wikipedia.org/wiki/Tiger>)

**Tillägg för högre betyg:**

* När man startar programmet så ska man komma till någon sorts inloggningsruta, och lösenord osv. ska i så fall sparas i en textfil på hårddisken (gärna krypterad).
* Kom ihåg att nämna inloggningsuppgifterna i rapporten

**Mer att tänka på:**

* Programmet bedöms till mycket stor del efter hur du löst allt (inklusive upplägg av databasen).
* Använd minst två klasser och försök att dela upp koden med metoder osv.
* Klassdiagram ***ska*** finnas med.
* Någon form av felhantering för att undvika körfel osv. ***bör*** finnas med.

### 2.2.2 Redovisning

**Programmet**

* Programmet ska fungera och uppfylla kravspecifikationen.
* Redovisa programmet genom att **skicka in källkodsfilerna** via din elevsida (komprimera ihop hela projektmappen).
* Programmet **ska** vara försett med **utförliga kommentarer** som förklarar de olika delarna i programmet.
* Programmet **ska** vara utan stilfel, dvs. med korrekta indrag och radbrytningar.
* För högre betyg **bör** programmet **utökas** med annan funktionalitet, utöver det som anges i beskrivningen, så länge detta inte strider mot den grundläggande beskrivningen, och uppfyller kraven.

**Dokumentation**

* **Projekt-rapport** ska skickas in tillsammans med källkoden en där du **beskriver** hur du gick till väga för att skapa programmet. Vilka avvägningar du gjort i programupplägget och motiveringar till dessa. Du **bör** också ge exempel på eventuella förbättringar.
* Uppgiften ska planeras med hjälp av pseudokod och/eller flödesschema som du sedan skickar in tillsammans med programmet.

## 2.3 Metod

Programmets källkod skrevs uteslutande i utvecklingsmiljön *Visual Studio.* Som projektmall användes *Windows Forms App (.NET Framework).*

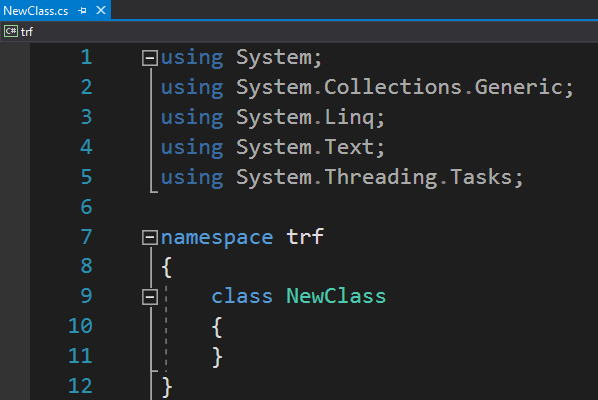
I Visual Studio finns ett hjälpverktyg kallat *IntelliSense,* som underlättar vid kodskrivning.

Visual Studio har också stöd för debugging med breakpoints, vilket användes för att felsöka programmet.



Figur Kod i Visual Studio, debugläge med break point

Egna klasser lades till i projektet genom att välja *”Project -> Add Class…”* i Visual Studios meny. Visual studio skapar då en ny cs-fil i projektet, med färdiggenererad kod för att skapa en ny klass.



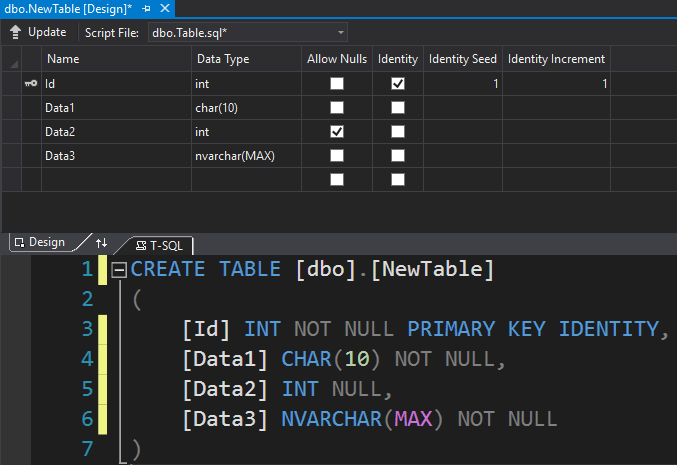
Figur Ny klassfil skapad i Visual Studio

På liknande sätt skapades nya Windows Forms till projektet, med *”Project -> Add Windows Form…”*

Databasen som används i programmet skapades och hanterades också i Visual Studio.

En ny databas läggs till i ett projekt med *"Project -> Add New Item -> Service-based Database"*  
Kapitel 5 i läroboken *Programmering 2 med C#* användes i stor grad som hjälpmedel vid arbetet med databasen.

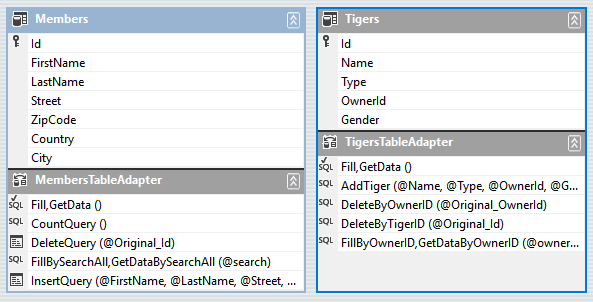
Databasens tabeller skapades i Visual Studio med hjälp av dess *”Table Designer”.*



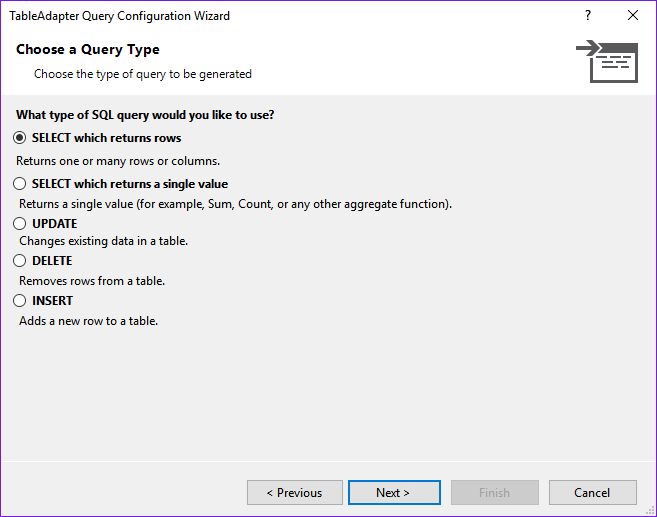
Figur Visual Studios Table Designer

Visual Studio använder sig av så kallade ”Table Adapters” för att sköta kommunikation mellan databaser och projektets kod. Till dessa adapters är det möjligt att skapa SQL-kommandon, som sparas som metoder till adapter-klassen.

Till projektet skapades flertalet SQL-kommandon (query), för att exempelvis söka bland medlemmar i TRF eller lägga till eller ta bort medlemmar. SQL-kommandona kan skapas med hjälp av guider i programmet.



Figur 5 Table Adapters med SQL-queries



Figur SQL-guide

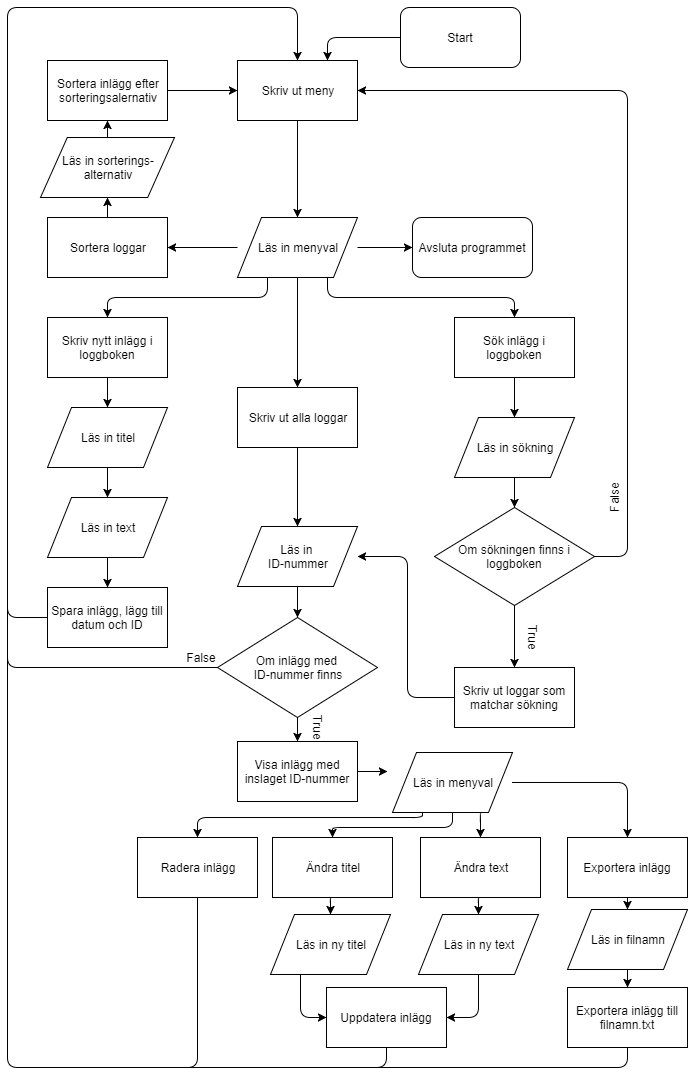
## 2.4 Planering

## 2.5 Programmets funktion

BESKRIV PROGRAMMET

## 2.5 Flödesschema

Här visas ett flödesschema för programmets funktion



Figur 7 Flödesschema

DATABASEN/SQL/ADAPTER ETC

## 2.8 Programmets kod

Källkoden beskrivs enbart övergripande i denna rapport, i den inlämnade källkodsfilen finns ytterligare kommentarer som beskriver koden.

Kod som är automatiskt genererad av Visual Studio beskrivs eller kommenteras ej.

### 2.8.1 Klassen Logbook

Klassen ”class Logbook” innehåller en lista av string-arrayer ”List<string[]> entries”, som används för att hålla loggbokens inlägg. Varje inlägg består av en array med fyra string-variabler; Titel, ID-nummer, Text-innehåll, och datum/tid.

Metoder i klassen Logbook

Flera metoder i klassen Logbook finns som överlagrade, en variant där parametrar skickas till metoden direkt och en där användaren ombeds skriva in data manuellt.

|  |  |
| --- | --- |
| Metod | Beskrivning |
| Logbook | Konstruktor för klassen. Laddar in sparad loggboksdata från fil om den existerar. Konstruktorn kan användas för att skapa generera testinlägg. |
| Count | Ger antal inlägg i loggboken. |
| ClearSearchHits | Nollställer sökresultat. |
| Search | Söker i logginlägg efter en angiven söksträng, finns som överlagrad. Metoden använder *linjär sökning*. |
| DisplaySearchHits | Listar titlarna på inlägg från det senaste sökresultatet. |
| DisplayAllTitles | Listar titlarna på samtliga inlägg i loggboken |
| DisplayTitle | Listar titeln på angivet inlägg, inläggets ID skickas som parameter. |
| AddEntry | Lägger till ett inlägg till loggboken, finns som överlagrad. |
| DeleteEntry | Raderar ett inlägg från loggboken. |
| DisplayEntry | Visar innehållet i ett inlägg, finns som överlagrad. Visar också en lista med alternativ för det aktuella inlägget: exportera, radera, redigera titel, redigera text. |
| EditTitle | Redigerar titel på ett inlägg. |
| EditContent | Redigerar texten i ett inlägg. |
| ExportEntry | Exporterar ett inlägg till en textfil. |
| Save | Sparar loggboken till en datafil. |
| Load | Laddar loggboken från en datafil. |
| UpdateEntry | Ändrar titel och/eller text för ett inlägg. Anropas av metoderna EditTitle och EditContent. |
| Sort | Sorterar loggboksinlägg. Finns som överlagrad. Metoden använder *insertion sort.* |
| SortSwap | Hjälpmetod för Sort, byter indexplats på två loggboksinlägg. |
| FindEntryIndex | Letar upp indexplats för ett inlägg, tar ID som parameter. |
| AddSamples | Lägger till några förskrivna logginägg till loggboken, för testning. |

Variabler och datastrukturer i klassen Logbook:

|  |  |  |
| --- | --- | --- |
| Namn | Typ | Beskrivning |
| entries | List<String[]> | Håller loggboksinlägg med titel, text, ID-nummer och datum. |
| searchHits | int[] | Array som håller sökträffar, som ID-nummer. |
| lastSearchString | string | Senaste söksträngen som användes för sökning |
| dataFileName | string | Filnamn för datafil som sparar loggboken mellan programkörningar. |
| id | int | Används för att sätta ID-nummer på nya logginlägg. |
| SearchData | enum | Värden för sökningsmetoder, tom och max antal sökträffar. |
| sortMenu | string[] | Text för den meny som visas vid val av sortering. |
| displayEntryMenu | string[] | Text för de alternativ som kan göras när ett inlägg visas. |
| SortBy | enum | Sorteringsalternativ. |
| EntryData | enum | Indexnummer för text, ID-nummer, datum och titel i char-arrayen i listan *entries*. |
| errorMsg | string[] | Felmeddelanden. |
| ErrorId | enum | Index för felmeddelanden. |

### 2.8.2 Klassen Menu

Klassen ”class Menu” innehåller metoder för att visa menyer och diverse text, samt funktionalitet för användarinmatning.

Metoder i klassen Menu:

|  |  |
| --- | --- |
| Metod | Beskrivning |
| DisplayMenu | Visar programmets huvudmeny, ger i retur användarens inmatade menyval. |
| DisplayTitle | Genererar titlar för programmets olika menyer. Titeltexten anges som parameter. Texten centreras. |
| DisplayLine | Ritar ut en linje till konsollfönstret. |
| Wait | Uppmanar användaren att trycka på en valfri tangent för att fortsätta programmet. Meddelande kan ges som parameter. |
| Confirm | Ger i retur ett sanningsvärde, beroende på om användaren trycker in en viss tangent. Exempel: *”Tryck på ’J’ för att ta bort filen”.* |
| Error | Visar ett felmeddelande, meddelandet anges som parameter. |

Variabler och datastrukturer i klassen Menu:

|  |  |  |
| --- | --- | --- |
| Namn | Typ | Beskrivning |
| MenuItem | enum | För navigering i huvudmenyn. |
| lineLen | int | Längd för linjer, används i DisplayLine och DisplayTitle |

# Diskussion och slutsats

## 3.1 Förslag på förbättringar

ETC