**Examinerande uppgift Loggboken**

Rapport

**Dokumentation för uppgiften ”Loggboken”**

**Kurs Programmering 1 C#**

**NTI-Skolan**

**Johan Kämpe  
2017-07-12**

# 1 Inledning

## Syfte

Syftet med uppgiften är att skapa en loggbok i programspråket C# där en användare kan skriva och hantera loggboksinlägg. Uppgiften används som examineringsunderlag i kursen Programmering 1 C#.

Som underlag för programmet används pseudokod som producerades in en tidigare inlämningsuppgift i kursen.

## Noteringar

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

## Länkar

FIXA

# 2 Genomförande och resultat

## 2.1 Använd programvara och dokumentation

SKRIV

## 2.2 Avgränsningar och krav

### 2.2.1 Grundläggande funktionalitet

Enligt uppgiftens dokumentation ska följande grundläggande funktionalitet för programmet uppnås:

* Programmet ska ha en fungerande meny med val för programmets funktionalitet, använd switch-villkor ska användas.
* Det ska finnas åtgärder för att förhindra körtidsfel.
* Det ska finnas möjlighet att skriva ut innehållet i loggboken.
* Det ska finnas möjlighet att spara nya inlägg i loggboken, med åtminstone titel och meddelande. Loggboken ska vara en lista medans varje enskild logg ska vara en vektor (array).
* Det ska finnas möjlighet att söka inlägg i loggboken, på exempelvis postens titel. Använd pseudokod eller flödesschema för att planera logiken för sökningen. Utgå från en linjär sökning i kursboken.

Källkoden ska vara väl kommenterad. Val av datatyper, kontrollinstruktioner, metoder, datastrukturer och algoritmer ska motiveras.

Programmet ska köras i ett konsolfönster, i en Windows-miljö.

### 2.2.2 Utökad funktionalitet

Enligt uppgiftens dokumentation kan programmer utökas med följande funktionalitet:

* Möjlighet att spara datum för loggboksinlägg.
* Möjlighet att redigera eller ta bort loggboksinlägg.
* Skapa en eller flera metoder för programmet.
* Implementera algoritmer för sökning och sortering av loggboksinlägg.

Programmets funktion, och förslag på förbättringar ska utvärderas i rapporten.

## 2.3 Metod

Programmets källkod skrevs uteslutande i utvecklingsmiljön *Microsoft Visual Studio Community 2017.* Som projektmall användes *Console App (.NET Framework).*

BESKRIV VARFÖR VS ÄR BRA, INTELLISENSE ETC

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

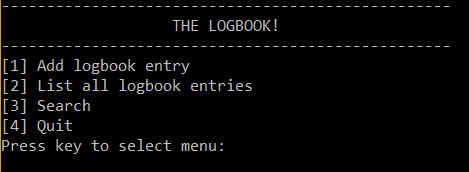


Figur 1 Kod i Visual Studio, debugläge med break point

Projektmallen skapar en första källkodsfil kallad *Program.cs* för projektet. Detta var den enda källkodsfil som användes för programmet.

## 2.4 Programmets funktion

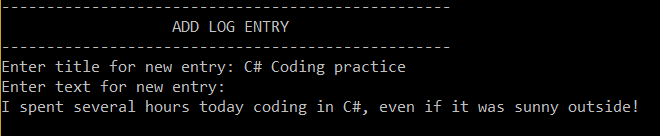
Vid programmets start presenteras en huvudmeny för användaren. Menyval utförs genom att trycka på en siffra på tangentbordet. I huvudmenyn behövs inget returslag efter sifferinmatning.



Figur 2 Programmets huvudmeny

Programmet kan användas som en loggbok eller dagbok. Användaren har möjlighet att skapa, visa, radera och ändra inlägg. OBS UPPDATERA SORTERING

När ett inlägg skapas med menyalternativ "[1] Add logbook entry" uppmanas användaren att skriva in en titel och sedan texten (innehållet) för inlägget.

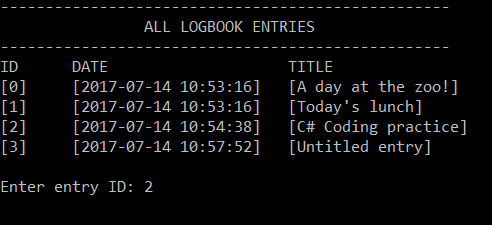


Figur 3 Skapande av ett nytt logginlägg

Ett inlägg kan skapas utan att skriva in en titel, det får då automatiskt titeln "Untitled entry". Ett inlägg kan dock inte ha tom text, om användaren inte skriver in någon text för inlägget visas ett felmeddelande som uppmanar användaren att försöka igen. Detta repeteras tills en text har matats in.

När ett inlägg skapas sätts också automatiskt ett ID-nummer och datum till inlägget.

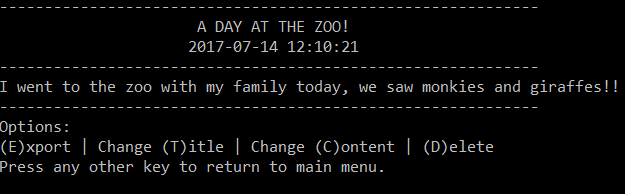
Med menyalternativ "[2] List all logbook entries" listas alla logginläggs titlar med datum och ID-nummer.



Figur 4 Lista med alla inlägg som finns i loggboken

Användaren ombeds skriva in ID-numret på det inlägg som ska visas. Om ett felaktigt ID-nummer matas in visas ett felmeddelande och programmet återgår till huvudmenyn.

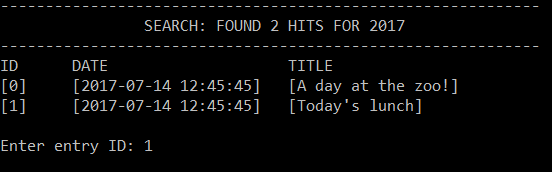
Vid visning av ett inlägg kan användaren välja att redigera (ändra titel eller textinnehåll), radera, eller exportera inlägget till en textfil.



Figur 5 Visning att ett inlägg, med alternativ

Alternativen väljs genom att trycka på den bokstav på tangentbordet som visas inom parenteser, exempelvis tangent ”*T*” för att välja alternativet "Change (T)itle".

Med menyalternativ "[3] Search" kan användaren skriva in en sträng för att söka i samtliga inlägg. Efter att sökningen har genomförts listas de inlägg där söksträngen förekommer. Likt menyalternativ 2 ombeds användaren också här att skriva in ett ID-nummer för att visa ett inlägg.

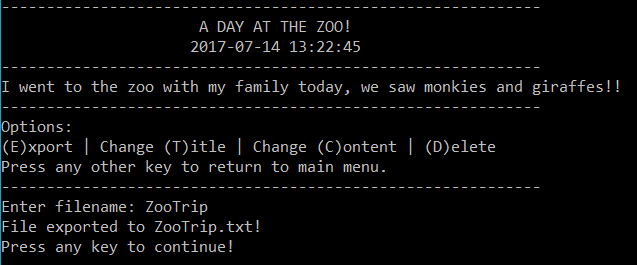


Figur 6 Sökning av logginlögg

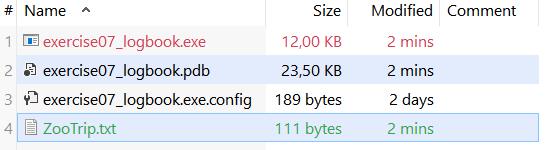
Notera att det går att skriva in ett ID-nummer för inlägg som inte visas i sök-listan, och detta ändå visas om det finns i loggboken.

Titlar, datum och inläggens text genomsöks för matchning.

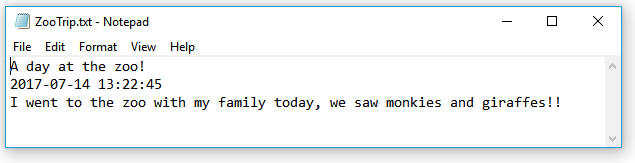
Vid exportering av ett inlägg ombeds användaren att skriva in ett filnamn. Filen sparas till samma katalog där programmets exekverbara fil finns. Om filen inte går att exportera visas ett felmeddelande.



Figur 7 Exportering av logginlägg



Figur 8 Exporterad textfil



Figur 9 Öppnad exporterad textfil

Loggboken data sparas mellan programkörningar, till en datafil kallad FIXA

## 2.5 Flödesschema

## 2.6 Pseudokod

## 2.7 Skillnader från uppgift 3

## 2.8 Programmets kod

### 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.

### 2.8.2 Klassen Menu

Klassen ”class Menu” innehåller metoder för att visa menyer och diverse text.

### 2.8.3 Metoder

# 3 Diskussion och slutsats

HOPPA ÖVER PILNAVIGERING – jobbigt vid listande av titlar

ÅÄÖ input konstigt…

Val av algoritmer