```c
/**
*Obligatorisk oppgave 1
*
*Oppgave: Bibliotek
*
*Fremgang
*1.Globale variabler og funksjoner
*2.Metoder/funksjoner.
*3.Kjore program, debug
*4.Forklaring og kommentere
*5.Optimaliser og husk minnefrigjoring!
*/

/**
*Interne includes
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


/**
*Eksterne includes
*/



/**
* Definisjoner statiske verdier
*/
#define STRLEN_MAX 25


/**
* Enum for handteringav  switch caser
*/
enum HANDLING
{
    NY, /**< enum verdi NY */
    LES, /**< enum verdi LES */
    AVSLUTT /**< enum verdi AVSLUTT */
};


/**
* @struct Bok
* @brief struct for Boker:
*En bok ineholder data en vanlig bok ville hat.
*/
struct Bok
{
    char mTittel[STRLEN_MAX]; /**< char array mTittel */
    char mSjanger[STRLEN_MAX]; /**< char array mSjanger */
```

```
54      int mSider; /**< integer lokal variabel mSider */
55  };
56
57
58  /**
59  * For deklarasjon av funksjoner
60  */
61  void Init();
62  void legg_til_bok();
63  void print_boker();
64  void fri_minne();
65
66  /**
67  *Globale variabler
68  */
69  struct Bok** Bibliotek = NULL; /** Array av boker */
70  int antall_boker = 0;
71
72  /**
73  * main funksjon:
74  * Kjorer programmet.
75  */
76  int main()
77  {
78      printf("_______Init: Oblig1_______\n\n");
79      Init();
80      return 0;
81  }
82
83  /**
84  * Initialiserings funksjon:
85  * Kjorer terminalen og tar imot input fra bruker.
86  */
87  void Init()
88  {
89      /**
90         Init sorger for at terminalen gir riktig alternativer.
91         Kjorer dermed valgt alternativ.
92      */
93
94      enum HANDLING tilstand = 0;
95      printf("Handlinger:\n");
96      printf("0 = legg til ny bok.\n");
97      printf("1 = printer alle bokene.\n");
98      printf("2 = avslutt.\n");
99      printf("Velg Handling:");
100     scanf_s("%d", &tilstand);
101     while (getchar() != '\n'); //Tommer buffer
102     printf("\n");
103     switch (tilstand)
104     {
105     case NY:
106     {
```

```
107         legg_til_bok();
108         break;
109     }
110     case LES:
111     {
112         print_boker();
113         break;
114     }
115     case AVSLUTT:
116     {
117         fri_minne();
118         exit(0);
119         break;
120     }
121     default:
122     {
123         printf("Velg ett gyldig alternativ:\n");
124         Init();
125         printf("\n");
126         break;
127     }
128     }
129     tilstand = 0;
130 }
131 
132 /**
133 * Legg til bok funksjon:
134 * Lager en ny bok og legger den til i biblioteket
135 */
136 void legg_til_bok()
137 {
138     // Lokale variabler for parameter verdier til bok
139     char tittel[STRLEN_MAX] = "\0";
140     char sjanger[STRLEN_MAX] = "\0";
141     int sider = 0;
142 
143     //Ber om at bruker skriver inn info
144     printf("[LOG]:Legger til Bok\n-----------------\n");
145     printf("Legg til tittel:");
146     gets_s(tittel, (unsigned)_countof(tittel));
147     printf("Legg til sjanger:");
148     scanf_s("%s", sjanger, (unsigned)_countof(sjanger));
149     printf("Legg til antall sider:");
150     scanf_s("%d",  &sider);
151     while (getchar() != '\n'); //Tommer buffer
152 
153     /*Ny bok med allokert minne(allokerer minne :
154     gjor klar en peker med samme antall bytes som parameteren) */
155     struct Bok* ny_bok = malloc(sizeof(struct Bok));
156 
157     //Flag som sjekker om Bok-structen har storresle 0
158     if (ny_bok == NULL)
159     {
```

```
160         printf("[WARNING]: Feil med minneallokering..\nReturnerer\n");
161         exit(0);
162     }
163 
164     //Kopier string fra parameter bok til lokal bok
165     strcpy_s(ny_bok->mTittel,STRLEN_MAX, tittel);
166     strcpy_s(ny_bok->mSjanger,STRLEN_MAX, sjanger);
167     ny_bok->mSider = sider;
168 
169     //Bibliotek med oppdatert minne
170     struct Bok** temp_lib = realloc(Bibliotek, sizeof(struct Bok*) *
          (antall_boker + 1));
171     if (temp_lib == NULL)
172     {
173         printf("[WARNING]: Feil med minneallokering..\nReturnerer\
              n");
174         exit(0);
175     }
176 
177     Bibliotek = temp_lib; //Oppdaterer arrayet sin storrelse
178     Bibliotek[antall_boker] = ny_bok; //Oppdaterer elementet for
          gjeldende indeks
179     antall_boker++; // Oppdaterer antall boker
180 
181     printf("-----------------\n[LOG]:Bok Lagt til\n\n");
182     Init();
183 }
184 
185 /**
186 * print boker funksjon:
187 * Printer ut alle boker i biblioteket.
188 * Printer i tillegg ut antall sider og boker lest totalt
189 */
190 void print_boker()
191 {
192 
193     //Deklarerer midlertidige variabler
194     int totalt_sider_lest = 0;
195 
196     printf("[LOG]:Printer info\n-----------------\n");
197 
198     //Flag som sjekker om arrayet er en null-peker eller ikke
          inneholder noen boker
199     if (Bibliotek == NULL || antall_boker == 0)
200     {
201         printf("[WARNING]: Ingen boker lagt til. Legg til boker\n");
202         printf("-----------------\n\n");
203         Init();
204         return;
205     }
206 
207     //Itterer over alle bokene
208     for (int i = 0; i < antall_boker; ++i)
```

```
209     {
210         totalt_sider_lest += Bibliotek[i]->mSider;
211         printf("Navn: \x1b[1m\x1b[4m%s\x1b[0m \nSjanger: \x1b[1m\x1b[4m
              %s\x1b[0m\n\n", Bibliotek[i]->mTittel, Bibliotek[i]-
              >mSjanger);
212     }
213     printf("Boker lest: \x1b[1m\x1b[4m%d\x1b[0m \nSider lest totalt:
          \x1b[1m\x1b[4m%d\x1b[0m\n", antall_boker, totalt_sider_lest);
214     printf("-----------------\n[LOG]:Printing ferdig\n\n");
215 
216     Init();
217 }
218 
219 /**
220 * fri minne funksjon:
221 * Frigjor minne for alle boker.
222 * Frigjor minne for arrayet av bok-pekere
223 */
224 void fri_minne()
225 {
226     // Frigjor minne: forst hver enkelt bok, dermed arrayet av Bok-
          pekere
227     for (int i = 0; i < antall_boker; ++i)
228     {
229         free(Bibliotek[i]);
230     }
231     free(Bibliotek);
232 
233     Bibliotek = NULL;
234 }
```