Oblig 4/Oblig 4.c

```c
/**
 *   Programeksempel nr 27 - Større programeksempel.
 *
 *   Programmet holder oversikt over et arrangement med ulke oppgaver.
 *   Man kan legge til og fjerne oppgaver, legge personer til i oppgaver og
 *   sette et maks antall per oppgave.
 *
 *   @file     Oblig 4.c
 *   @author   Gjermund H. Pedersen
 */
#include <stdbool.h> // bool
#include "LesData.h" // lesInt, lesChar, lagOgLesText

#define MAXPERS 6  ///< Maks personer som kan være per oppgave
#define MAXOPPG 20 ///< Maks oppgaver man kan legge inn

// Datamedlemene til en oppgave
struct Oppgave {
char* navn;        // Navnet på oppgaven
int antallTotalt,  // Hvor mange som totalt kan være på oppgaven
    antallNaa;     // Hvor mange som er på oppgaven nå
int hvem[MAXPERS]; // IDen til de som er på oppgaven
};

struct Oppgave* gOppgavene[MAXOPPG]; ///< Arrayet som holder pekerne til oppgavene
int gSisteOppgave = 0;               ///< Hvor mange oppgaver som er lagt til

void nyOppgave();
void oppgaveLesData(struct Oppgave* oppgave);
void skrivOppgaver();
void oppgaveSkrivData(const struct Oppgave* oppgave);
void ledigeOppgaver();
bool oppgaveLedigPlass(const struct Oppgave* oppgave);
void personerTilknyttesOppgave();
void oppgaveTilknyttPersoner(struct Oppgave* oppgave);
void fjernOppgave();
void oppgaveSlettData(struct Oppgave* oppgave);
void skrivMeny();


/**
 *  Hovedprogrammet:
 *
 *  Leser en komando og kjører den korresponderende funksjonen
 *  @see nyOppgave
 *  @see skrivOppgaver
 *  @see ledigeOppgaver
 *  @see personerTilknyttetOppgave
 *  @see fjernOppgavene
 *  @see skrivMeny
 */
int main ()  {
```

```c
    char kommando;

    skrivMeny();
    kommando = lesChar("\nKommando");

    while (kommando != 'Q')  {
        switch (kommando)  {
          case 'N': nyOppgave();                 break;
          case 'S': skrivOppgaver();             break;
          case 'L': ledigeOppgaver();            break;
          case 'P': personerTilknyttesOppgave(); break;
          case 'F': fjernOppgave();              break;
          default:  skrivMeny();  break;
        }
        kommando = lesChar("\nKommando");
    }

    printf("\n");
    return 0;
}

/**
 * @brief Lager en ny oppgave
 *
 * @see oppgaveLesData
*/
void nyOppgave(){
    // Legger ikke til oppgave hvis det ikke er plass
    if (gSisteOppgave>MAXOPPG){
        printf("Det er ikke plass til flere oppgaver");
    }
    else{
        gOppgavene[gSisteOppgave] = (struct Oppgave*) malloc(sizeof(
            struct Oppgave)); // Akkokerer minne til oppgaven
        oppgaveLesData(gOppgavene[gSisteOppgave]);
        gSisteOppgave++; // Oppdaterer hvor mange oppgaver som er lagt til
    }
};

/**
 * @brief Leser data inn i en oppgave
 *
 * @param oppgave - oppgaven som skal fylles ut
*/
void oppgaveLesData(struct Oppgave* oppgave){
    oppgave->navn = lagOgLesText("Skriv inn et navn");
    oppgave->antallTotalt = lesInt("Hvor mange kan jobbe på denne oppgaven",
        0, MAXPERS);
    oppgave->antallNaa = 0;
};

/**
 * @brief Skriver ut alle oppgavene og infoen
 *
 * @see oppgaveSkrivData
```

```
108 */
109 void skrivOppgaver(){
110     if (!gOppgavene[0]){
111         printf("Det finnes ingen oppgaver.\n");
112     }
113     else{
114         for (int i = 0; i < gSisteOppgave; i++){
115             oppgaveSkrivData(gOppgavene[i]);
116         }
117     }
118 };
119 
120 /**
121  * @brief Skriver ut dataen i en oppgave
122  *
123  * @param oppgave - oppgaven som skal bli skrevet ut.
124 */
125 void oppgaveSkrivData(const struct Oppgave* oppgave){
126     printf("\tNavn: %s\n", oppgave->navn);
127     printf("\tMax antall på oppgvaen: %i\n", oppgave->antallTotalt);
128     printf("\tAntall på oppgvaen: %i\n", oppgave->antallNaa);
129     printf("\tPerson id-er på oppgven: ");
130     // Looper gjennom alle oppgavene som er satt.
131     // Sjekker om større enn 0 fordi compileren på windows satte variablene
132     // negative i stedet for 0.
133     for (int i = 0; oppgave->hvem[i]>0; i++){
134 
135         printf("%i, ", oppgave->hvem[i]);
136     }
137     printf("\n");
138 };
139 
140 /**
141  * @brief Går gjennom alle oppgavene og skriver ut de som ikke er fulle
142  *
143  * @see oppgaveLedigPlass
144 */
145 void ledigeOppgaver(){
146     if (!gOppgavene[0]){
147         printf("Det finnes ingen oppgaver.\n");
148     }
149     else{
150         for (int i = 0; i < gSisteOppgave; i++){
151             if (oppgaveLedigPlass(gOppgavene[i])){
152                 oppgaveSkrivData(gOppgavene[i]);
153             }
154         }
155     }
156 };
157 
158 /**
159  * @brief Finner ut om en oppgave har ledig plass
160  *
161  * @return Om det er ledig plass i oppgaven
162 */
```

```
163 bool oppgaveLedigPlass(const struct Oppgave* oppgave){
164     return oppgave->antallNaa < oppgave->antallTotalt;
165 };
166 
167 /**
168  * @brief Legger til en eller flere personer til en oppgave
169  *
170  * @see oppgaveTilknyttPersoner
171 */
172 void personerTilknyttesOppgave(){
173     if (!gOppgavene[0]){
174         printf("Det finnes ingen oppgaver.\n");
175     }
176     else{
177         int oppgaveId = lesInt("Hvilken oppgeve", 0, MAXOPPG+1);
178         if(oppgaveId==0){ // Sjekker om brukeren angrer
179             printf("Angrer. Ingenting endret\n");
180             return;
181         }
182         else{
183             oppgaveTilknyttPersoner(gOppgavene[oppgaveId-1]);
184         }
185 
186     }
187 };
188 
189 /**
190  * @brief Legger til en person i en spesifisert oppgave
191  *
192  * @see oppgaveSkrivData
193  * @see oppgaveLedigPlass
194 */
195 void oppgaveTilknyttPersoner(struct Oppgave* oppgave){
196     oppgaveSkrivData(oppgave);
197     if(!oppgaveLedigPlass(oppgave)){
198         printf("Oppgaven er full.");
199         return;
200     }
201     else{
202         // Spørr brukeren hvor mange som skal legges til.
203         // Må være mellom oppgavens maksantall og nåværende
204         int antallNye = lesInt("Hvor mange nye skal legges til", 0,
205                                oppgave->antallTotalt - oppgave->antallNaa);
206         for (int i = 0; i < antallNye; i++){
207             // Legger inn IDen og øker med 1 etter
208             oppgave->hvem[oppgave->antallNaa++] = lesInt(
209                 "Id på personen som legges til", 0, 1000);
210         }
211         oppgaveSkrivData(oppgave);
212     }
213 };
214 
215 /**
216  * @brief Fjerner den oppgaven brukeren spesifiserer
217  *
```

```c
 * @see oppgaveSlettData
*/
void fjernOppgave(){
    if (!gOppgavene[0]){
        printf("Det finnes ingen oppgaver.\n");
        return;
    }
    else{
        int oppgaveId = lesInt("Hvilken oppgave vil du fjerne", 0, MAXOPPG+1);
        if(oppgaveId==0){
            printf("Ingenting ble fjernet.\n");
            return;
        }
        else{
            char bekreftelse = lesChar(
                "Er du sikker på at du vil fjerne den siste oppgaven: (J/N)");
            if (bekreftelse=='J'){
                if(oppgaveId==gSisteOppgave){
                    printf("Den siste oppgaven ble fjernert.\n");
                    oppgaveSlettData(gOppgavene[gSisteOppgave-1]);
                    free(gOppgavene[gSisteOppgave-1]);
                    gOppgavene[gSisteOppgave-1] = NULL;
                    gSisteOppgave--;
                }
                else{
                    printf("Oppgaven ble fjernet\n");
                    oppgaveSlettData(gOppgavene[oppgaveId-1]);
                    free(gOppgavene[oppgaveId-1]);
                    gOppgavene[oppgaveId-1] = gOppgavene[gSisteOppgave-1];
                    gOppgavene[gSisteOppgave-1] = NULL;
                    gSisteOppgave--;
                }
            }
            else{
                printf("Avbryter! Ingenting ble endret.\n");
            }
        }
    }
};

/**
 * @brief Frigjør allokert minne til den spesifiserte oppgaven
*/
void oppgaveSlettData(struct Oppgave* oppgave){
    free(oppgave->navn);
};

/**
 * @brief Skriver ut en meny som forklarer komandoene
 */
void skrivMeny(){
    printf("Kommandoer:\n");
    printf("\tN: Legg til en ny oppgave i arrangent oversikten.\n");
    printf("\tS: Skriv ut alle arangementene og dataene deres.\n");
    printf("\tL: Skriv ut arrangementer med ledig plass\n");
```

```
273     printf("\tP: Legg til en eller flere personer på et arrengment.\n");
274     printf("\t   - Skriv 0 for å kanselere, og ingenting blir endret.\n");
275     printf("\tF: Fjern et arrengement.\n");
276     printf("\t   - Skriv 0 for å kanselere, og ingenting blir endret.\n");
277     printf("\t   - Skriv \"siste\" for å fjerne den siste oppgaven.\n");
278     printf("\tQ: Avslutt programmet.\n");
279 };
```