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>
#include <string.h>
#include "LesData.h"

#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
 */
int main ()  {
    char kommando;

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

    while (kommando != 'Q')  {
        switch (kommando)  {
          case 'N': nyOppgave();                break;
          case 'S': skrivOppgaver();            break;
```

```c
            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
*/
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
*/
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
*/
void skrivOppgaver(){
    if (!gOppgavene[0]){
        printf("Det finnes ingen oppgaver.\n");
    }
    else{
        // Lager en kopi av pekeren til arrayet slik at vi ikke mister starten
        struct Oppgave** oppgaver = gOppgavene;
        printf("Oppgaver:\n");
        do
        {
            oppgaveSkrivData(*oppgaver);
        }while (*(++oppgaver)); // Øker hvor pekeren peker med en og så sjekker
    }                           // om den pekeren er NULL. Hvis sen ikke er NULL
};                              // blir neste oppgave skrevet ut.

/**
 * @brief Skriver ut dataen i en oppgave
*/
```

```c
void oppgaveSkrivData(const struct Oppgave* oppgave){
    printf("\tNavn: %s\n", oppgave->navn);
    printf("\tMax antall på oppgvaen: %i\n", oppgave->antallTotalt);
    printf("\tAntall på oppgvaen: %i\n", oppgave->antallNaa);
    printf("\tPerson id-er på oppgven: ");
    // Looper gjennom alle oppgavene som er satt.
    // Sjekker om større enn 0 fordi compileren på windows satte variablene
    // negative i stedet for 0.
    for (int i = 0; oppgave->hvem[i]>0; i++){

        printf("%i, ", oppgave->hvem[i]);
    }
    printf("\n");
};

/**
 * @brief Går gjennom alle oppgavene og skriver ut de som ikke er fulle
*/
void ledigeOppgaver(){
    if (!gOppgavene[0]){
        printf("Det finnes ingen oppgaver.\n");
    }
    else{
        // Lager en kopi av pekeren til arrayet slik at vi ikke mister starten
        struct Oppgave** oppgaver = gOppgavene;
        printf("Ledige oppgaver: \n");
        while(*(++oppgaver)){ // Øker hvor pekeren peker med en og så sjekker
                              // om den pekeren er NULL. Hvis sen ikke er NULL
                              // blir neste oppgave skrevet ut.
            if (oppgaveLedigPlass((*oppgaver))){
                oppgaveSkrivData((*oppgaver));
            }
        };
    }
};

/**
 * @brief Finner ut om en oppgave har ledig plass
*/
bool oppgaveLedigPlass(const struct Oppgave* oppgave){
    return oppgave->antallNaa < oppgave->antallTotalt;
};

/**
 * @brief Legger til en eller flere personer til en oppgave
*/
void personerTilknyttesOppgave(){
    if (!gOppgavene[0]){
        printf("Det finnes ingen oppgaver.\n");
    }
    else{
        char* oppgaveNavn = lagOgLesText("Hvilken oppgeve");
        if(strcmp("0", oppgaveNavn)==0){ // Sjekker om brukeren angrer
            return;
        }
        else{
            // Lager en kopi av pekeren til arrayet slik at vi ikke mister
            // starten
            struct Oppgave** oppgaver = gOppgavene;
```

```
            // Sjekker om navnet brukeren ga stemmer med
            do{
                if(strcmp(oppgaveNavn, (*oppgaver)->navn)){
                    oppgaveTilknyttPersoner((*oppgaver));
                }
            }while(*(++oppgaver)); // Øker hvor pekeren peker med en og så
        }                          // sjekker om den pekeren er NULL. Hvis sen
    }                              // ikke er NULL blir neste oppgave skrevet ut
};

/**
 * @brief Legger til en person i en spesifisert oppgave
*/
void oppgaveTilknyttPersoner(struct Oppgave* oppgave){
    oppgaveSkrivData(oppgave);
    if(!oppgaveLedigPlass(oppgave)){
        printf("Oppgaven er full.");
        return;
    }
    else{
        // Spørr brukeren hvor mange som skal legges til.
        // Må være mellom oppgavens maksantall og nåværende
        int antallNye = lesInt("Hvor mange nye skal legges til", 0,
                               oppgave->antallTotalt - oppgave->antallNaa);
        for (int i = 0; i < antallNye; i++){
            // Legger inn IDen og øker med 1 etter
            oppgave->hvem[oppgave->antallNaa++] = lesInt(
                "Id på personen som legges til", 0, 1000);
        }
        oppgaveSkrivData(oppgave);
    }
};

/**
 * @brief Fjerner den oppgaven brukeren spesifiserer
*/
void fjernOppgave(){
    if (!gOppgavene[0]){
        printf("Det finnes ingen oppgaver.\n");
        return;
    }
    else{
        char* oppgaveNavn = lagOgLesText("Hvilken oppgave vil du fjerne");
        if(strcmp("0", oppgaveNavn)==0){
            printf("Ingenting ble fjernet.\n");
            return;
        }
        // Lager en kopi av pekeren til arrayet slik at vi ikke mister starten
        struct Oppgave** oppgaver = gOppgavene;

        if (strcmp("siste", oppgaveNavn)==0){
            char bekreftelse = lesChar(
                "Er du sikker på at du vil fjerne den siste oppgaven: (J/N)");
            if (bekreftelse=='J')
            {
                oppgaveSlettData(gOppgavene[gSisteOppgave]);
                gOppgavene[gSisteOppgave] = NULL;
                gSisteOppgave--;
                printf("Oppgaven ble slettet.\n");
```

```
234                 return;
235             }
236             else{
237                 printf("Avbryter! Ingenting ble endret.\n");
238             }
239         }
240         else{
241             while(*oppgaver){
242                 if (strcmp(oppgaveNavn, (*oppgaver)->navn)==0){
243                     char bekreftelse = lesChar(
244                         "Er du sikker på at du vil fjerne den siste oppgaven: (J/N)");
245                     if (bekreftelse=='J'){
246                         oppgaveSlettData(*oppgaver);
247                         (*oppgaver) = gOppgavene[gSisteOppgave];
248                         gOppgavene[gSisteOppgave] = NULL;
249                         gSisteOppgave--;
250                         printf("Oppgaven ble slettet.\n");
251                         return;
252                     }
253                     else{
254                         printf("Avbryter! Ingenting ble endret.\n");
255                     }
256                 }
257                 oppgaver++;
258             }
259             printf("Fant ikke oppgaven.\n");
260         }
261     }
262 };
263 
264 /**
265  * @brief Frigjør allokert minne til den spesifiserte oppgaven
266 */
267 void oppgaveSlettData(struct Oppgave* oppgave){
268     free(oppgave->navn);
269     free(oppgave);
270 };
271 
272 /**
273  * @brief Skriver ut en meny som forklarer komandoene
274  */
275 void skrivMeny(){
276     printf("Kommandoer:\n");
277     printf("\tN: Legg til en ny oppgave i arrangent oversikten.\n");
278     printf("\tS: Skriv ut alle arangementene og dataene deres.\n");
279     printf("\tL: Skriv ut arrangementer med ledig plass\n");
280     printf("\tP: Legg til en eller flere personer på et arrengment.\n");
281     printf("\t   - Skriv 0 for å kanselere, og ingenting blir endret.\n");
282     printf("\tF: Fjern et arrengement.\n");
283     printf("\t   - Skriv 0 for å kanselere, og ingenting blir endret.\n");
284     printf("\t   - Skriv \"siste\" for å fjerne den siste oppgaven.\n");
285     printf("\tQ: Avslutt programmet.\n");
286 };
```