```c
/**
 *  Oblig 4 i grunnleggende programmering.
 *
 *  Programmet utfører forskjellige kommandoer fra brukeren:
 *  -Lager oppgaver, og leser inn navn og hvor mange som trengs til oppgaven.
 *  -Tilknytte personIDer opp mot de forskjellige oppgavene.
 *  -Sjekker om de forskjellige oppgavene er ledige.
 *  -Kan også slette oppgaver.
 *
 *  @file    oblig4.c
 *  @author  William Seiner
 */

#include <stdio.h>          // scanf, printf
#include <stdbool.h>        // bool
#include "LesData.h"        //  Verktøykasse for lesing av diverse data.

#define MAXPERS 6
#define MAXOPPG 20
const int STRLEN = 90;

/**
 *  Struct for oppgave:
 */
struct Oppgave {
    char* navn;
    int antallTotalt,
    antallNaa;
    int hvem[MAXPERS];
};

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


struct Oppgave* gOppgavene[MAXOPPG];    // Peker til array med alle oppgavene
int gSisteOppgave = 0;      // Teller antall oppgaver

/**
 *  Hovedprogrammet:
 */
int main ()  {
    char kommando;

    skrivMeny();
    kommando = lesChar("\n\nSkriv kommando ");

    while (kommando != 'Q')  {
        switch (kommando)  {
            case 'N':  nyOppgave();  break;
            case 'S':  skrivOppgaver();    break;
            case 'L':  ledigeOppgaver();  break;
            case 'T':  personerTilknyttesOppgave();  break;
            case 'F':  fjernOppgave(); break;
            default:   skrivMeny(); break;
        }
        kommando = lesChar("\n\nSkriv kommando ");
    }

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

/**
 *  Fjerner valgt oppgave.
 *
 *  @see  lesInt(...)
 *  @see  oppgaveSlettData(...)
 */
void fjernOppgave() {
    int valgtOppgaveNr;

    if (gSisteOppgave>0) {
        valgtOppgaveNr=lesInt("Skriv inn oppgave nr som du vil slette: ", 1, gSisteOppgave)-1;

        if (valgtOppgaveNr+1==(0)) printf("Du har valgt å avslutte denne kommandoen, og det slettes ingenting");

        else {
            char svar;

            svar=lesChar("Skriv inn (J/j) hvis du virkelig vil slette oppgaven");

            // Hvis bruker brukersvar er ja:
            if (svar=='J'){
                oppgaveSlettData(gOppgavene[valgtOppgaveNr]);

                if (valgtOppgaveNr==gSisteOppgave) {
                    printf("Du har fjernet de siste oppgaven");
                }
                else {
                    printf("Du har valgt å fjerne oppgavenr: %i", valgtOppgaveNr);

                    // Bakerste oppgave flytter inn for slettet oppgave:
                    gOppgavene[valgtOppgaveNr]=gOppgavene[gSisteOppgave-1];
                }
                gSisteOppgave--;
            }
            else printf("Du har valgt å avslutte denne kommandoen, og det slettes ingenting");
        }

    }
    else printf("Det finnes ingen oppgaver");
}

/**
 *  Allokerer nok plass i memory til struct for oppgave, og sjekker at det ikke er fullt med oppgaver.
 *
 *  @see  oppgaveLesData(...)
 */
void nyOppgave() {
    if (gSisteOppgave<=MAXOPPG) {
        printf("Ny oppgave nr:  %i\n", gSisteOppgave+1);

        // Allokerer plass i memory for den nye oppgaven:
        gOppgavene[gSisteOppgave] = (struct Oppgave*) malloc(sizeof(struct Oppgave));

        oppgaveLesData(gOppgavene[gSisteOppgave]);     //Leser inn data til opppgaven

        gSisteOppgave++;
    }
    else {
        printf("Det er fullt med oppgaver");
    }

}

/**
 *  Skriver ut de ledige oppgavene
```

```
 *
 *  @see  oppgaveLedigPlass(...)
 *  @see  oppgaveSkrivData(...)
 */
void ledigeOppgaver() {
    for (int i=0; i<gSisteOppgave; i++) {

        //Hvis det er ledig plass i oppgaven, skrives data om oppgaven ut:
        if(oppgaveLedigPlass(gOppgavene[i])) oppgaveSkrivData(gOppgavene[i]);
    }
}

/**
 *  Sjekker om oppgave har ledig plass
 *
 *  @param  oppgave - oppgave som sjekkes om har ledig plass
 */
bool oppgaveLedigPlass(const struct Oppgave* oppgave) {
    if (oppgave->antallNaa<oppgave->antallTotalt) return true;
    else return false;
}

/**
 *  Leser inn data til struct for oppgave
 *
 *  @param oppgave - Oppgave som det leses inn informasjon til
 *
 *  @see  lesInt(...)
 *  @see  lagOgLesText(...)
 */
void oppgaveLesData(struct Oppgave* oppgave) {
    oppgave->navn = lagOgLesText("Navn: ");
    oppgave->antallTotalt = lesInt("Antall personer som trengs til oppgaven: ", 0, MAXPERS);
    oppgave->antallNaa = 0;
}

/**
 *  Skriver ut oppgave
 *
 *  @param oppgave - Oppgave som skal skrives ut
 */
void oppgaveSkrivData(const struct Oppgave* oppgave) {
    printf("Navn: %s, Antall totalt: %i , Antall nå: %i , Hvem: ", oppgave->navn, oppgave->antallTotalt, oppgave->antallNaa);
    for (int i=0; i<oppgave->antallNaa; i++) printf("%i, ", oppgave->hvem[i]);    //Skriver ut alle personIDer på hvem som har oppgaven
    printf("\n\n");
}

/**
 *  Frigir allokert memory for valgt oppgave
 *
 */
void oppgaveSlettData(struct Oppgave* oppgave) {
    free(oppgave->antallNaa);
    free(oppgave->antallTotalt);
    free(oppgave->hvem);
    free(oppgave->navn);
    free(oppgave);
}

/**
 *  Tilknytter arbeidere til oppgave
 *
 *  @param oppgave - Structen som skal tilknyttes personer
 *
 *  @see  oppgaveSkrivData(...)
 *  @see  oppgaveLedigPlass(...)
 */
void oppgaveTilknyttPersoner(struct Oppgave* oppgave) {
    oppgaveSkrivData(oppgave);  //Skriver data ut om medsendt oppgave
    int hvorMangeTilknyttes;
    //Hvis oppgaven ikke har ledig plass:
    if (!oppgaveLedigPlass(oppgave)) {
        printf("Denne oppgaven trenger ikke flere arbeidere");
    }
    else {
        //Kan skrive inn hvor mange som skal tilknyttes opptil så mange plasser som er ledige:
        hvorMangeTilknyttes=lesInt("Hvor mange vil du tilknytte til denne oppgaven(0 for å ikke legge til noe noen): ", 0, (oppgave->antallTotalt-oppgave->antallNaa));

        //Hvis hvor mange som skal tilknyttes ikke er 0:
        if (hvorMangeTilknyttes!=(0)){
            //Løkke som går fra hvor mange som har tatt plass i oppgaven til hvor mange man vil tilknytte:
            for (int i=(oppgave->antallNaa); i<((hvorMangeTilknyttes)+(oppgave->antallNaa)); i++) {
                oppgave->hvem[i]=lesInt("Skriv inn arbeiders nummer: ", 1, 1000);
            }
            oppgave->antallNaa += hvorMangeTilknyttes;
        }
        else printf("Du har valgt å ikke legge til noen");
    }
}

/**
 *  Finner ut hvilken oppgave personer skal tilknyttes til
 *
 *  @see  oppgaveTilknyttPersoner(...)
 */
void personerTilknyttesOppgave() {

    int valgtOppgaveNr;

    //Hvis det er fler enn 0 oppgaver:
    if (gSisteOppgave>0) {
        valgtOppgaveNr=lesInt("Skriv inn oppgave nr (0 for å avslutte) ", 0, gSisteOppgave)-1;

        //Hvis valgt oppgave nummer er 0:
        if (valgtOppgaveNr+1==(0)) printf("Du har valgt å avslutte denne kommandoen");

        //Hvis ikke kjøres funksjon som tilknytter personer til valgt oppgave
        else oppgaveTilknyttPersoner(gOppgavene[valgtOppgaveNr]);
    }
    else printf("Det finnes ingen oppgaver");
}

/**
 *  Skriver ut kommandomeny.
 */
void skrivMeny() {
    printf("Kommandomeny:\n");
    printf("N for ny oppgave\n");
    printf("S for å skrive oppgave\n");
    printf("L for å vise ledige oppgaver\n");
    printf("T for å tilknytte person til oppgave\n");
    printf("F for å fjerne oppgave\n");
    printf("Q for å stoppe programmet\n");
}

/**
 *  Skriver data ut om alle oppgaver.
 *
 *  @see  oppgaveSkrivData(...)
 *
 */
void skrivOppgaver() {
    for (int i=0; i<gSisteOppgave; i++) oppgaveSkrivData(gOppgavene[i]);
}
```