NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Kontinuasjonseksamensoppgave i PROG1001 – Grunnleggende programmering

Faglig kontakt under eksamen:	**Frode Haug**
Tlf:	**950 55 636**

Eksamensdato:	**11.august 2021 - HJEMME**
Eksamenstid (fra-til):	**09:00-13:00 (4 timer)**
Hjelpemiddelkode/Tillatte hjelpemidler:	**F - Alle trykte og skrevne.** (kalkulator er *ikke* tillatt)
Annen informasjon:	

Målform/språk:	**Bokmål**
Antall sider (inkl. forside):	**8**

Informasjon om trykking av eksamensoppgaven

Originalen er:

1-sidig X **2-sidig** ☐

sort/hvit X **farger** ☐

Skal ha flervalgskjema ☐

Kontrollert av:

_____

Dato Sign

NB: Oppgave 1a, 1b og 2 er totalt uavhengige og kan derfor løses separat.

Oppgave 1 (30%)

I hele oppgave 1, være nøye med å :

- skrive *hele* funksjonen, dvs. *både* headingen og *hele* innmaten
- bruke `const` (når dette er aktuelt) ifm. parametre
- kommentere funksjonen (hva den gjør, evt. parametre og evt. returverdi) etter Doxygen

a) Lag funksjonen `int returnerDenStorste(.....)`

Funksjonen får inn tre *ulike* `int`-verdier som parametre. Den finner ut hvem som er størst av dem, og returnerer denne verdien.

b) Lag funksjonen `int gangArrayer(.....)`

Funksjonen får inn to *eksakt like lange* `int`-arrayer (`arr` og `arr2`), samt `n` som angir antall elementer i hver av dem. Funksjonen skal oppdatere `arr`, slik at *alle* elementene er ganget med det elementet som har samme indeks i `arr2`. I tillegg skal funksjonen returnerer *total*summen av *alle* de oppdaterte elementene i `arr`. Eksempel:

ArrayA før:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ArrayB før:	2	2	2	4	4	4	8	8	8	16	16	16	32	32	32
ArrayA etter:	2	4	6	16	20	24	56	64	72	160	176	192	416	448	480

Totalsummen av ArrayA: 2136

Oppgave 2 (70%)

Les *hele* teksten for denne oppgaven (2a-2g) *nøye*, før du begynner å besvare noe som helst. Studér vedlegget, som inneholder mange viktige opplysninger som du trenger/skal bruke. Legg spesielt merke til `#define`/`const`, `enum`, structen med datamedlemmer, funksjoner, globale variable, `main()` og to ferdiglagde funksjoner. Husk også på de ferdiglagde funksjonene for å lese inn data på `LesData.h`. Bruk *alt* dette *svært* aktivt.

Det skal holdes orden på produkter med rakfisk og pultost, samt hvem som har produsert/laget disse. En og samme produsent kan faktisk produsere/lage en eller begge av disse produktene.

Datastrukturen

Datastrukturen består (se vedlegget) av arrayen `gProdukter`. I denne er indeksene fra 0 til `gAntallProdukter-1` i bruk. Vedlegget angir også hvilke datamedlemmer `Produkt` inneholder. **NB:** Produktene angis for/av brukeren (ved I/O) fra 1 til `gAntallProdukter`, selv om de i arrayen ligger lagret fra indeks nr.0 til `gAntallProdukter-1`. Legg også merke til arayen `gProdusent`. I denne ligger det ferdig hardkodet navnene til de aktuelle produsentene i programmet. Disse lagres som verdien 0 (null) til 4 i hvert `Produkt` sin `produsentNr`.
*Vedlegget inneholder alt du trenger av structer, datamedlemmer og globale variable for å løse denne eksamensoppgaven. Dessuten er **prototyper** for alle funksjoner også ferdig deklarert/definert.*

Oppgaven

a) **Skriv innmaten til** `void skrivAlleProduktene()` **og**
`void produktSkrivData(const struct Produkt* p)`

Den første funksjonen sørger for at *alle* produktene og deres numre blir skrevet ut. Vha. den andre funksjonen blir *alle* data inni hvert `Produkt` skrevet, men for `produsentNr` skrives det aktuelle produsent*navnet*. De to `enum`-ene skrives ut med en tilsvarende tekst som selve verdien, med *store* bokstaver.

b) **Skriv innmaten til** `void nyttProdukt()` **og**
`void produktLesData(struct Produkt* p)`

Den første funksjonen kommer med en melding om det ikke er plass til flere produkter. I motsatt fall skrives det nye produktets nummer ut på skjermen, og et nytt produkt opprettes. Den andre funksjonen leser inn produktets fire datamedlemmer. `produsentNr` leses inn som 1 til 5, men lagres altså som 0 (null) til 4. `antallKg` *kan* også være 0 (null).

c) **Skriv innmaten til** `void skrivAlleAvType()` **og**
`bool produktErType(const struct Produkt* p, enum Type t)`

Den første funksjonen kommer med en melding om det er tomt for produkter. I motsatt fall spør den om en `Type` (vha. ferdiglaget funksjon). Så skrives *alle* produktene av denne typen (som man finner svar på hva den andre funksjonen) ut på skjermen. Det kommer en egen melding om *ingen* ble skrevet ut/funnet.

d) **Skriv innmaten til** `void endreKilo()` **og**
`void produktEndreKilo(struct Produkt* p)`
Den første funksjonen kommer med en melding om det er tomt for produkter. I motsatt fall spør den om et produktnummer, 0 (null) inkludert. Skrives '0' kommer det en melding om at ingenting skjer. I motsatt fall kalles den andre funksjonen. Denne igjen skriver først ut *alle* produktets nåværende data. Deretter leser den inn en ny verdi for `antallKg` (0 (null) inkludert). Til slutt skriver den ut den nye innleste verdien for dette datamedlemmet.

e) **Skriv innmaten til** `void skrivAlleMedNavn()` **og**
`bool produktMedSubtekst(const struct Produkt* p,const char* nvn)`
Den første funksjonen skal gjøre *eksakt* det samme som den første i oppgave 2c, bortsett fra at den ber om en tekst i stedet for en `Type`, og at den bruker den andre funksjonen nevnt her i oppgave 2e og *ikke* den andre i 2c. Den andre funksjonen finner ut og returnerer `true/false` til om `nvn` er *hele eller deler av* produktets `navn`.

f) **Skriv innmaten til** `void skrivTilFil()` **og**
`void produktSkrivTilFil(FILE* ut, const struct Produkt* p);`
Funksjonene sørger til sammen for at *hele* datastrukturen blir skrevet til «PRODUKT.DTA». `gAntallProdukter` *skal* ligge aller først på filen. **Filformatet** ellers bestemmer du helt selv, men ***skal*** **oppgis i besvarelsen**.

g) **Skriv innmaten til** `void lesFraFil()` **og**
`void produktLesFraFil(FILE* inn, struct Produkt* p);`
Funksjonene sørger til sammen for at *hele* datastrukturen blir lest inn fra «PRODUKT.DTA», etter det formatet du selv bestemte i forrige deloppgave.

Annet (klargjørende):

- Vi forutsetter at brukeren av programmet husker/kan hvilke produsenter som har hvilke numre. For det er ingen egen utskrift av dette noe sted i programmet.
- Gjør dine egne forutsetninger og presiseringer av oppgaven, dersom du skulle finne dette nødvendig. Gjør i så fall klart rede for disse *i starten* av din besvarelse av oppgaven.
- **NB:** Det skal *ikke* brukes C++-kode, dvs. slikt som f.eks: string-klassen, kode fra STL, templates eller andre større hjelpebiblioteker. Men, de vanligste includer brukt i hele høst er tilgjengelig.

Måtte både rakfisk og pultost smake deg godt i høst!

FrodeH

Vedlegg til PROG1001, august 2021: Halvferdig programkode

```
#include <stdio.h>               //  printf, FILE
#include <stdlib.h>              //  sizeof, malloc
#include <string.h>              //  strcpy, strlen, strstr
#include <stdbool.h>             //  bool, true, false
#include "LesData.h"             //  Verktøykasse for lesing av diverse data


#define   MAXPRODUKTER  200      ///<  Max. antall produkter.
#define   ANTPRODUSENTER  5      ///<  Max. antall produsenter.
const int STRLEN   =     80;     ///<  Max. tekstlengde.
const int MAXKILO  = 100000;     ///<  Max. antall kilo årlig produsert.


/**
 *  Type (et produkt er enten 'Rakfisk' eller 'Pultost').
 */
enum Type { Rakfisk,  Pultost };


/**
 *  Produkt (med navn, produkttype, produsentens nr/indeks og årlige kilo).
 */
struct Produkt {
    char* navn;                  //  Produktets navn/beskrivelse.
    enum Type type;              //  Produkttype:  Rakfisk eller Pultost.
    int produsentNr;             //  Produsentens nr (navn) - 0 til 4.
    int antallKg;                //  Ca. antall kilo i årlig produksjon.
};


void skrivMeny();
enum Type lesType();
void skrivAlleProduktene();                                      //  Oppgave 2A
void produktSkrivData(const struct Produkt* p);                  //  Oppgave 2A
void nyttProdukt();                                              //  Oppgave 2B
void produktLesData(struct Produkt* p);                          //  Oppgave 2B
void skrivAlleAvType();                                          //  Oppgave 2C
bool produktErType(const struct Produkt* p, enum Type t);        //  Oppgave 2C
void endreKilo();                                                //  Oppgave 2D
void produktEndreKilo(struct Produkt* p);                        //  Oppgave 2D
void skrivAlleMedNavn();                                         //  Oppgave 2E
bool produktMedSubtekst(const struct Produkt* p,const char* nvn); // Oppg.2E
void skrivTilFil();                                              //  Oppgave 2F
void produktSkrivTilFil(FILE* ut, const struct Produkt* p);      //  Oppgave 2F
void lesFraFil();                                                //  Oppgave 2G
void produktLesFraFil(FILE* inn, struct Produkt* p);             //  Oppgave 2G


int    gAntallProdukter;             ///<  Antall produkter hittil registrert.
struct Produkt* gProdukter[MAXPRODUKTER];  ///<  Alle produktene.
char*  gProdusent[ANTPRODUSENTER] =        ///<  Aktuelle produsentnavn.
         { "LofossBakken", "TineRøn", "WangenFinden",
           "HaademHolen", "SynnøveNoraker" };
```

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

    lesFraFil();                                                //  Oppgave 2G

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

    while (kommando != 'Q') {
        switch (kommando) {
           case 'A': skrivAlleProduktene();         break;  //  Oppgave 2A
           case 'N': nyttProdukt();                 break;  //  Oppgave 2B
           case 'T': skrivAlleAvType();             break;  //  Oppgave 2C
           case 'E': endreKilo();                   break;  //  Oppgave 2D
           case 'I': skrivAlleMedNavn();            break;  //  Oppgave 2E
           default:  skrivMeny();                   break;
        }
        kommando = lesChar("\nØnske");
    }

    skrivTilFil();                                              //  Oppgave 2F

    return 0;
}



/**
 *  Skriver/presenterer programmets muligheter/valg for brukeren.
 */
void skrivMeny() {
    printf("\nFØLGENDE KOMMANDOER ER LOVLIG:\n");
    printf("\tA   = skriv Alle produktene\n");
    printf("\tN   = Ny produkt\n");
    printf("\tT   = skriv produktene av en Type\n");
    printf("\tE   = Endre ca.antall kilo årlig produsert\n");
    printf("\tI   = skriv produktene Inneholdende tekst i navnet\n");
    printf("\tQ   = Quit/avslutt\n");
}



/**
 *  Leser forbokstaven i 'Rakfisk/Pultost', og returnerer aktuell enum-verdi.
 *
 *  @return  Enum-verdi ut fra forbokstaven i enum-verdien
 */
enum Type lesType() {
  char tegn;
  do  {                                                 //  Leser og sikrer kun
    tegn = lesChar("\tType (R(akfisk), P(ultost))");    //    bokstavene
  } while (tegn != 'R'  &&  tegn != 'P');               //    'R' og 'P'.
  return ((tegn == 'R') ? Rakfisk : Pultost);           //  Returnerer aktuell
}                                                       //    enum-verdi.
```

```
/**
 *  Oppgave 2A - Skriver ALT om ALLE produkter.
 *
 *  @see   produktSkrivData(...)
 */
void skrivAlleProduktene() {                                /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2A - Skriver ALT om ETT produkt ut på skjermen.
 *
 *  @param   p - Produktet som skrives ut
 */
void produktSkrivData(const struct Produkt* p)  {           /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2B - Legger inn (om mulig) et nytt produkt i datastrukturen.
 *
 *  @see   produktLesData(...)
 */
void nyttProdukt() {                                        /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2B - Leser inn alle relevante datamedlemmer.
 *
 *  @param   p -  Produktet som får innlest sine data
 */
void produktLesData(struct Produkt* p) {                    /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2C - Skriver (om mulig) ALT om ALLE produkter av EN GITT type.
 *
 *  @see   lesType()
 *  @see   produktErType(...)
 *  @see   produktSkrivData(...)
 */
void skrivAlleAvType() {                                    /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2C - Returnerer om produktet er av en gitt type.
 *
 *  @param    p  -  Aktuelt produkt å sjekke 'type' for
 *  @param    t  -  Aktuell enum-type å sjekke opp mot
 *  @return   Om produktet 'p' har 'type' lik 't' eller ei
 */
bool produktErType(const struct Produkt* p, enum Type t){   /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2D - Endrer (om mulig) et produkts antall kilo årlig produsert.
 *
 *  @see   produktEndreKilo(...)
 */
void endreKilo() {                                          /*  LAG INNMATEN  */  }
```

```
/**
 *  Oppgave 2D - Endrer et produkts antall kilo årlig produsert.
 *
 *  @param   p  -  Aktuell produkt å endre antall kilo for
 *  @see     produktSkrivData(...)
 */
void produktEndreKilo(struct Produkt* p) {                  /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2E - Skriver (om mulig) ALLE produkter med (sub)tekst i navnet.
 *
 *  @see   produktMedSubtekst(...)
 *  @see   produktSkrivData(...)
 */
void skrivAlleMedNavn() {                                   /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2E - Returnerer om produktet innholder en tekst i navnet sitt.
 *
 *  @param    p    -  Aktuelt produkt å sjekke om inneholder subteksten 'nvn'
 *  @param    nvn  -  Aktuell tekst å sjekke om er (sub)tekst i 'p->navn'
 *  @return   Om produktet 'p' inneholder subteksten 'nvn' eller ei
 */
bool produktMedSubtekst(const struct Produkt* p, const char* nvn)  {
                                                            /*  LAG INNMATEN  */  }

/**
 *  Oppgave 2F - Skriver ALLE produktene til fil.
 *
 *  @see   produktSkrivTilFil(...)
 */
void skrivTilFil() {                                        /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2F - Alle dataene for ETT produkt skrives ut på fil.

 *
 *  @param   ut  - Filen det skal skrives til
 *  @param   p   - Produktet som skrives til fil
 */
void produktSkrivTilFil(FILE* ut, const struct Produkt* p) {  /* LAG INNMATEN  */ }


/**
 *  Oppgave 2G - Leser ALLE produktene fra fil.
 *
 *  @see     produktLesFraFil(...)
 */
void lesFraFil() {                                          /*  LAG INNMATEN  */  }


/**
 *  Oppgave 2G - Leser ALT om ETT produkt fra fil.
 *
 *  @param  inn - Filen det skal leses fra
 *  @param  p   - Produktet som får innlest sine data
 */
void produktLesFraFil(FILE* inn, struct Produkt* p) {       /*  LAG INNMATEN  */  }
```