Oppgaver

PROG1001 - Grunnleggende programmering

Forord

Dette kompendie/hefte innholder oppgaveteksten for ulike oppgaver i emnet ”PROG1001 - Grunnleggende programmering” ved NTNU.

Opplysninger om emnet er å finne på: **http://folk.ntnu.no/frh/grprog**

Under ”Opplegg” er det angitt hvilke oppgaver i dette heftet som til enhver tid er relevante som ukeoppgaver.

(Data)filer som oppgavetekstene henviser til og alle løsningsforslag er begge deler å finne på katalogen ”OPPGAVER”.

Nye/andre oppgavetekster *kan* bli laget i løpet av semesteret. Disse vil da fortløpende bli lagt til bakerst i dette heftet.

NTNU

Frode Haug

Oppgave 1

Skriv algoritmer (vha. pseudokode og skrittvis forfining) for å:

a) beregne en trekants areal

b) finne fram til og bøte hullet i et sykkeldekk (forhjulet)

Oppgave 2

Skriv pseudokode (vha. modularisering og skrittvis forfining) for et system som holder orden på et musikksamling (mp3, CD'er, plater og kassetter).

Oppgave 3

Start opp Code::Blocks (CB) (eller den kompilatoren/IDE du har valgt å bruke på din maskin). På katalogen «EKSEMPLER» ligger bl.a. EKS_01.c. Kjør dette programmet (jfr. pkt.3 i pdf'en om enkel bruk av CB). Endre bevisst på koden, og se hva slags feilmeldinger kompilatoren da kommer med. Kjør også EKS_02.c, og endre litt på koden, slik at den fortsatt kan kjøres (kompilerer korrekt), og se hva slags andre/nye utskrifter du får. Eksperimenter litt med verktøyet ellers, dvs. bli litt kjent med de ulike menyvalgene.

Oppgave 4

Skriv et selvlaget program (jfr. pkt.4 i pdf'en om enkel bruk av CB) som skriver ut ditt navn, gateadresse, postnummer, poststed, mobilnummer og alder. De tre tallene legges i hver sin heltallsvariabel (som altså skrives ut). Utskriften *skal* være på fem ulike linjer, der bare postnummer og poststed står på samme linje, med *en* blank mellom. Linjene starter med ledetekstene «Navn:», «Adresse:», «Sted:», «Mobilnr:» og «Alder:». Prøv å få utskriften av alle dataene til å starte på samme kolonne.

NB1: Alle tekster skal *ikke* leses inn fra tastaturet, men skrives rett i koden mellom gåsøyne.

NB2: Husk god kommentering ut fra standarden vi bruker/følger.

Oppgave 5

Skriv et program som inneholder de *totalt* uavhengige kodebitene for å:

a) beregne og skrive ut volumet for en rettvinklet prisme (kloss). Sidene er henholdsvis 8, 12 og 16. Lagre disse verdiene i hver sin variabel, før volumet beregnes og skrives ut.

b) har en variabel med verdien «19600406», og som finner årstallet 1960 og skriver ut dette.

Oppgave 6

a) Skriv «Bli-kjent-mail» og læringsassistenene dine
(se eget punkt *helt* til slutt under «Informasjon» på emnets hjemmeside).

b) Koble opp og logg inn på skolens datanett. Bli litt bedre kjent med NTNUs hjemmeside og spesielt: Innsida, timeplanen din, Studentweb, Blackboard, emnenes hjemmesider og NTNU-mailen.

Oppgave 7

Skriv et program som:

- inneholder heltallene `tall1`, `tall2`, `tall3`, `sum`, `kvadratSum`
- inneholder flyttallene `flyt1`, `flyt2`, `gjsnittSum`, `gjsnittKvadrat`
- initierer/setter de tre første heltallene og de to første flyttallene til noen selvvalgte/ønskelig verdier
- lar `sum` bli inneholdende summen av de tre første heltallene, og skriver ut *alle* tallene som har vært involvert i regnestykket (`tall1`, `tall2`, `tall3` og `sum`)
- lar `gjsnittSum` bli inneholdende gjennomsnittet for de tre tallene i `sum`, og skriver ut dette svaret/tallet. Hvorfor må du huske å dele/dividere med 3.0 ?
- lar `kvadratSum` bli inneholdende summen av de tre første heltallene kvadrert (ganget med seg selv), og skriver ut dette svaret/tallet
- lar `gjsnittKvadrat` bli inneholdende gjennomsnittet for summen av de tre kvadratene i `kvadratSum`, og skriver ut dette svaret/tallet. Del også her med 3.0.
- skriver ut gjennomsnittet av flyt1 og flyt2, *uten* å bruke en hjelpevariabel. Hvorfor er det strengt tatt ikke påkrevd å dele med 2.0 her?

Oppgave 8

Skriv et program som:

- inneholder en heltallskonstant (`ANTALL`) lik 17
- inneholder heltallene `tall1`, `tall2`, `tall3` og `svar`
- setter de tre heltallene til noen selvvalgte/ønskede verdier
- lager noen regnestykker (ved å bruke +, -, *, / og parenteser) der disse tre heltallene og `const`'en er i bruk, svarene på regnestykkende legges inn i `svar`, og denne skrives ut for hvert regnestykke
- lager og skriver ut svarene på regnestykker der `+=`, `-=`, `*=` og `/=` brukes på `svar` sammen med `int`'ene og `const`'en.
- øker `tall1` med en og en tre ganger, minsker `tall2` med en og en fire ganger (skriver ut hver av dem før og etter operasjonene), og så til slutt skriver ut produktet av de nåværende tallene fratrukket `ANTALL`.

Oppgave 9

Skriv et program som inneholder heltallsvariablene `totaltSekunder`, `timer`, `minutter` og `sekunder`. Denne første settes til 312304. Programmet beregner så, inn de de tre andre variablene, hva dette tilsvarer i timer, minutter og sekunder. Alt dette skrives ut, slik at utskriften blir: *312304 sekunder tilvarer: 86 time(r), 45 minutt(er) og 4 sekund(er)*
Hint: Både '/ ' (divisjon) og '%' (modulo) bør brukes.

Oppgave 10

Skriv et program som:

- inneholder tegnene `tegn1`, `tegn2` og flyttalene `flyt1`, `flyt2`
- setter disse verdiene henholdsvis til: 'F', 'H', 17.52 og 451.87
- skriver ut både tegnene og flyttallene i baklengs ift. definisjons-rekkefølgen
- skriver ut `flyt2` på eksponentiell form (dvs. utskriften blir: 4.518700e+002)
- skriver ut *kun* heltallsdelen av `flyt1` og `flyt2`
- skriver ut `tegn1` castet om til en `int`. Hva er dette for slags utskrift/tall?
- skriver ut 'b' castet om til en `int`.

Oppgave 11

NB: Husk at arrayer starter indekseringen på element nr.0 (det første elementet). Derfor vil f.eks. det tredje elementet ha indeks nr.2. Ikke noe heter derfor «det nulte elementet».

Skriv et program som:

- inneholder to `int`-arrayer (`tall` og `tall2`). `tall` har fem elementer, men er uinitiert. `tall2` har ikke-definert lengde, men blir satt til å inneholde henholdsvis verdiene: 6, 2, 10, 12, 19, 3 og 7.
- inneholder tre `char`-arrayer (`tekst`, `navn` og `adresse`). `tekst` har fem elementer, men er uinitiert. `navn` har ikke-definert lengde, men blir satt til å inneholde teksten «Lars Hansen». `adresse` har også ikke-definert lengde, men blir satt til å inneholde teksten «Ringgata 111».
- legger inn verdiene 13 og 67 i de to første elementene i `tall`
- beregner summen av disse to elementene, legger svaret inn i og skriver ut det tredje elementet
- halverer verdien i det tredje elementet og skriver ut svaret
- i det fjerde elementet i `tall` legger summen av de fire første elementene av `tall2`
- i det femte elementet i `tall` legger summen av de tre siste elementene av `tall2`
- skriver ut innholdet av de to siste nettopp oppdaterte elementene
- øker verdien i det tredje elementet i `tall` fire enkelt-ganger (vha. ++), og skriver ut resultatet

- legger ‘A’ inn i `tekst[1]` og ‘E’ inn i det fjerde elementet
- kopierer `navn` sitt sjette element inn i det første i `tekst`, og `navn` sitt første element inn i det tredje i `tekst`
- skriver «Kua har » etterfulgt av innholdet av de fire første elementene i `tekst`
- skriver kun ut «111», hentet fra `adresse`
- *hele* `navn` og `adresse` på hver sin linje

Oppgave 12

Skriv et program inneholdende variabler av typen `int`, `float`, `char`,
samt `char`-arrayer (tekster/strenger). Bruk `const` ifm. arrayenes lengde.
Eksperimenter/«lek» litt med innlesning og utskrift i alle disse variablene.

Oppgave 13

Lag er program som skriver ut sammenhengen mellom radie og volum for en sylinder.
Høyden skal fast være 4 (`const`. PI er også en `const`). Høyeste radie leses inn fra brukeren (som skriver et tall i intervallet 5-20). Programmet går så i løkke, fra 1 til
det innleste tallet (høyeste radie), og skriver *en* sammenheng (mellom radie og utregnet volum) pr.linje på skjermen. Høyrejuster gjerne alle tallene.

Oppgave 14

Lag et program som ber om positive heltall fra brukeren. Når brukeren slår inn 0 eller et negativt tall, skal programmet stanse. Det skal da skrive ut antallet som er lest, totalsum og gjennomsnittet av tallene.

Oppgave 15

Lag et program som leser inn to heltall. Deretter skriver det ut svaret for når disse to er addert, subtrahert og multiplisert.
Tilslutt spørres brukeren om vedkommende vil gjøre dette en gang til.
Om svaret er 'j', så utføres det ovenfor beskrevne enda en gang, helt til hun/han svarer 'n'.

Oppgave 16

Skriv et program som:

- leser inn fem verdier i en `int`-array
- sikrer at *alle* disse verdiene er mellom `const`-ene MINVERDI og MAXVERDI (konstantene er henholdsvis 1 og 10000).
- skriver ut alle de fem innleste heltallene
- leser inn ti tegn (ett og ett ad gangen) i en `char`-array
- gjør hver av dem evt. om til en stor bokstav (vha. `toupper`)
- sikrer at *alle* tegnene virkelig *er* bokstaver (A-Z, Æ, Ø og Å regnes ikke med)
- skriver ut alle de ti innleste bokstavene

Oppgave 17

Skriv et program som beregner pongsummen for et visst antall skøyteløpere på en gitt distanse. Programmet (*lovlig* betyr at det loopes inntil det angitte er innskrevet/lest):

1. leser inn en *lovlig* distanse (500, 1500, 3000 eller 5000) for kvinner.
2. beregner antall 500-metre i denne distansen (brukes ved beregning av poeng)
3. leser et *lovlig* antall løpere det skal leses tider for (1-MAXLOPERE (`const`, =10))
4. går gjennom det innleste antallet løpere, og for hver av dem utføres pkt.5 og 6:
5. leser vedkommendes tid (minutt, sekund og hundredel – alle er `int`).
   Sikre at disse er *lovlige* ved at de alle er større eller lik 0 (null). Du trenger *ikke* å sikre at de ellers er rimelige (f.eks. at 10.000 meter gås på tiden 2:03:24).
6. beregner og skriver ut løperens poengsum (`float`) ut fra de innleste tidene.
   poeng = (minutt * 60 + sekund + hundredel / 100) / antall-500-metre
   F.eks. 1500m på tiden 1 52 13: poeng = (1*60 + 52 + 13/100) / 3 = 37.377
7. helt til slutt spørres brukeren om vedkommende vil kjøre *hele* programmet fra begynnelsen av igjen. Dvs. starte på pkt.1 igjen.

Oppgave 18

Skriv et program som går igjennom tallene fra 1 til 100. Skriver hvert av dem på hver sin linje, og bak hvert tall skrives det om tallet er et partall eller oddetall.
Klarer du å få alle tallene pent høyrejustert innenfor tre posisjoner?
Klarer du i stedet å få til fem og fem tall (med utskrift) på hver linje?

Oppgave 19

Skriv et program som leser 20 tall inn i en `int`-array. *Alle* tallene *skal* være i intervallet 1-100 (lag en `do`-loop som sikrer dette). Deretter gås det gjennom alle tallene, og det telles opp (i en annen `int`-array, 5 lang) antall innleste tall i intervallene 1-20, 21-40, 41-60, 61-80 og 81-100. Til slutt gås det gjennom denne siste arrayen, og det lages en utskrift som:

Antall tall i intervallet 1 - 20: 5
Antall tall i intervallet 21 - 40: 6
Antall tall i intervallet 41 - 60: 5
Antall tall i intervallet 61 - 80: 0
Antall tall i intervallet 81 - 100: 4

Tallene helt til høyre er bare eksempler. Kursiv er bare for å illustrere utskrift.

Oppgave 20

Skriv et program som leser inn ett telefonnummer (null eller flere blanke mellom hvert siffer), og skriver ut dette i klartekst.
Eks: input: *22 34 45 56* output: *to-to-tre-fire-fire-fem-fem-seks*
Klarer du å få til '-' mellom alle tallene, unntatt til slutt?
Klarer du å få utskriften til å komme med en feilmelding, og umiddelbart stanse utskriften, om det kommer noe annet enn sifre og blanke? Det skal *ikke* brukes exit, return eller goto.

Oppgave 21

Skriv funksjonen `void skrivTid (int time, int min, int sek)` som skriver dette ut på formatet: `tt:mm:ss` Husk '`:`' mellom verdiene.
Tilslutt skriver den ut hva dette *totalt* tilsvarer i sekunder. Lag også et hovedprogram (`main`) som tilkaller/bruker denne funksjonen med litt ulike verdier som input.
Klarer du også å få til en innledende '0' (null) om `min` og/eller `sek` er et en-sifret tall?

Oppgave 22

Skriv funksjonen `void lagFirkant(int bredde, int hoyde)` som skriver ut en ramme med '*' ut fra parametrene. Kalles den f.eks som `lagFirkant(20, 6)`, så skrives følgende ramme på skjermen:

```
********************
*                  *
*                  *
*                  *
*                  *
********************
```

Husk at det er `bredde-2` blanke tegn på linjene med: * *, og at det er `hoyde-2` slike linjer. Lag et hovedprogram som looper tre ganger, og som for hver gang spørr brukeren om ønsket bredde og høyde, før disse verdiene sendes med til funksjonen.
(Du trenger *ikke* å sjekke at innskrevne verdier fra brukeren virkelig er positive.)

Oppgave 23

Skriv funksjonen `int summerKvadrater(int tall, int tall2, int tall3)` som kvadrerer hver av parametrene, summerer disse kvadratene og returnerer denne summen. Lag et hovedprogram som som tilkaller/bruker denne funksjonen med litt ulike verdier som input. **NB:** *Husk kommentering etter Doxygen!*

Oppgave 24

Lag funksjonene:

`int hentMinsteVerdi(const int n)`
som leser 'n' verdier/tall fra brukeren, og som returnerer det minst av dem.

`int hentStorsteVerdi(const int n)`
som leser 'n' verdier/tall fra brukeren, og som returnerer det største av dem.

Lag et hovedprogram som tester disse to funksjonene.

Antagelig har de to funksjonene blitt *meget* like. Lag i stedet *en* funksjon:
`int hentVerdi(.....)` som ivaretar begge de to funksjonenes funksjonalitet, bare at noe av innmaten styres av nye parametre du selv lager (bruk gjerne `enum`).
Lag hovedprogrammet også teste den nye funksjonen for minste og største verdi.

Oppgave 25

Lag funksjonene:

`int mstrlen(const char s[])`
som returnerer lengden på strengen "s".

`void mstrcpy(char s[], const char t[])`
som kopierer strengen "t" inn i "s".

`void mstrcat(char s[], const char t[])`
som kopierer strengen "t" inn på enden av "s", slik at sluttresultatet blir en sammenhekting (konkatenering) av de to strengene.

`int mstrcmp(const char s[], const char t[])`
som sammenligner strengene "s" og "t", og som returnerer:

0	dersom "s" == "t"	
< 0	dersom "s" < "t"	(dvs. kommer *alfabetisk før*)
> 0	dersom "s" > "t"	(dvs. kommer *alfabetisk etter*)

Lag et hovedprogram som tester funksjonene ovenfor. Pass på at det hele tiden *virkelig* er plass til sluttresultatet i den strengen det kopieres til/skjøtes på med en annen tekst.
NB: *Ikke* bruk de tilsvarende ferdiglagde funksjonene fra `string.h` i denne oppgaven.

Oppgave 26

Skriv et program med en `struct Dato` som inneholder heltallene `dag`, `maaned` og `aar`. Definer to struct-variable: `dato` og `dato2`. Les inn fra brukeren verdier til *alle* medlemmene i begge disse to variablene. (Du trenger *ikke* å bruke `lesTall`-funksjonen, men bare vanlig `scanf`. Vi regner med at brukeren skriver lovlige/rimelige verdier.)
Programmet skriver til slutt ut hvilken av datoene som har det laveste/tidligste årstallet, evt. om de er like.

Oppgave 27

Skriv et program med `struct Person` som inneholder `navn` (`char`-array, 80 lang), `alder` (`int`) og `vekt` (`float`). Definer en passe lang array med slike structer.
Spør brukeren om hvor mange personer vedkommende vil lese inn data om (dette *må* være et tall mellom 1 og så mange det er plass til i arrayen). Deretter leses *alle* data inn om så mange personer. Bruk `lesTall`- og `lesTekst`-funksjonene, men modifiser den første slik at den returnerer en `float`. Til slutt skriver programmet ut:

- *total* samlet alder for *alle* personene
- deres gjennomsnittlige vekt
- nummeret, navnet og alderen til den som er eldst
  (er det flere med samme alder, skrives bare den første ut)

Oppgave 28

Skriv et program med `struct Ansatt` som inneholder `ansattNr` (`int`),
`navn` (`char`-array, 80 lang) og `lonn` (`float`).

- Definer to `struct`-variable i `main`: `ansatt` og `ansatt2`
- Initier den første med noen passende verdier
- Den andre sine data leses inn fra brukeren (bruk *kun* `scanf` og `gets`. `lesTall` og `lesTekst` trenger altså *ikke* å brukes/lages)
- Skriv så ut begge de to structenes innhold. Lag funksjonen `skriv(…)` for dette, som tar structens nummer (1 eller 2, *ikke* `ansattNr`) og structen som parameter.
- Bytt om begge de to structenes innhold (lag en hjelpe-struct til å kopiere over i)
- Skriv så ut igjen begge de to structenes «nye» innhold

Oppgave 29

Skriv et program med `struct Brok` som inneholder de to heltallene `teller` og `nevner`. Lag funksjonene:

1. `struct Brok lesBrok()`
   Leser inn og returnerer en `Brok`, der `teller` er et hvilket som helst slags heltall, og `nevner` *er* et positivt heltall (> 0).

2. `void skrivBrok(const struct Brok b)`
   Skriver parameterens data på formen: teller / nevner f.eks: 14 / 17

3. `struct Brok addisjon(const struct Brok b, const struct Brok b2)`
   Adderer/summerer de to brøkene og returnerer svaret.

4. `struct Brok subtraksjon(const struct Brok b, const struct Brok b2)`
   Subtraherer/trekker fra hverandre de to brøkene og returnerer svaret.

5. `struct Brok multiplikasjon(const struct Brok b, const struct Brok b2)`
   Multipliserer/ganger sammen de to brøkene og returnerer svaret.

6. `struct Brok divisjon(const struct Brok b, const struct Brok b2)`
   Dividerer/deler det to brøkene med hverandre og returnerer svaret.

Lag et hovedprogram som:

- inneholder `struct Brok brok, brok2, svar;`
- i starten fyller `brok` og `brok2` med data ved å bruke funksjon nr.1
- fyller `svar` fortløpende etter tur ved å bruke/tilkalle funksjonene nr.3-6 med `brok` og `brok2` med som parametre. For hver gang skrives (vha. funksjon nr.2) begge brøkene og svaret ut på skjermen.

Minner om at regneoperasjonene er definert ved:

$$\frac{a}{b} + \frac{c}{d} = \frac{a*d + b*c}{b*d} \qquad \frac{a}{b} - \frac{c}{d} = \frac{a*d - b*c}{b*d}$$

$$\frac{a}{b} * \frac{c}{d} = \frac{a*c}{b*d} \qquad \frac{a}{b} / \frac{c}{d} = \frac{a*d}{b*c}$$

Oppgave 30 (= « oblig 3.5 »)

I denne oppgaven skal det lages et lite, men komplett program som holder orden på utlån av traller i en fornøyelsespark. Slike traller lånes ut til publikum (for en gitt sum pr. dag), slik at de lettere kan frakte med seg den håndbagasjen (ekstra klær, mat, drikke, o.l) de har med seg til parken.
Trallene lånes ut fra ei lita bu like ved parkens inngang. På en datamaskin inne i denne bua kjører det programmet som her skal lages. Trallene står "linet opp" utenfor bua.
På hver tralle er det påklistret et unikt nummer i intervallet 1-30. Når en besøkende ønsker å låne ei tralle, tar vedkommende bare ei ledig ei utenfor bua, og går deretter inn i bua og oppgir trallenummeret, sitt navn og mobilnummer, og betaler.

Programmet *skal* inneholde følgende *globale* `struct` og data:

```
struct Tralle  {
  bool utlaant;
  char navn[80];
  int  tlfNr;
};

struct Tralle gTraller[30];          // Array av Tralle-struct'er.
int    gAntallUtlaant = 0;           // Antall traller som pt. er utlånt.
const int MAXTRALLER  =  30;         ///< Max. antall traller til utlån.
```

`gAntallUtlaant` forteller *totalt* hvor mange av de 30 trallene som for tiden er utlånt. Dvs. siden de besøkende *ikke* låner trallene i stigende nummerrekkefølge,
vil `gAntallUtlaant` *ikke* inneholde siste indeks som så langt er tatt i bruk i arrayen, men den vil inneholde hvor mange av de 30 trallene som *totalt* er i bruk for øyeblikket.

Lag `main`, etter mønsteret i bl.a EKS_27.c. Denne må tidlig initiere alle trallene til «ikke-utlånt». Dessuten blir det bruk for funksjonene: `skrivMeny` (der det må lages en ny tilpasset utskrift), `lesKommando`, `lesTall` og `lesTekst`.

Lag også de fem funksjonene (som alle tilkalles/startes fra `main`):

`void oversikt()` – kommando 'O'
Det kommer en egen melding om *ingen* traller er utlånt. I motsatt fall gås det gjennom *alle* trallene, og *kun* for de som er utleid skrives trallenummeret, navnet og telefonnr.

`void ledige()` – kommando 'L'
Det kommer en egen melding om *alle* trallene er utlånt. I motsatt fall gås det gjennom *alle* trallene, og nummeret skrives på *alle* de ledige/ikke-utleide.

`void utlaan()` – kommando 'U'
Det kommer en egen melding om *alle* trallene er utlånt. I motsatt fall spørres det om et trallenummer. Er denne faktisk utlånt allerede, kommer det også en melding. Ellers markeres den som utlånt, lånerens navn og telefonnummer leses og lagres, samt at total antall utleide telles opp.

`void innlevering()` – kommando 'I'

Det kommer en egen melding om *ingen* traller er utlånt. I motsatt fall spørres det om et trallenummer. Er denne *ikke* utlånt, kommer det også en melding. Ellers markeres den som ikke-utlånt, total antall utleide telles ned, og navnet skrives ut for den som hadde lånt trallen.

`void finnLaaner()` – kommando 'F'

Det spørres om og leses inn navnet for en potensiell tralleleier. Det gås gjennom *alle* trallene, og nummeret på den *ene eller flere* trallene vedkommende for øyeblikket leier, blir skrevet ut på skjermen. Om vedkommede faktisk *ikke* leier noen i det hele tatt, kommer det også en egen melding/tekst om dette.

Oppgave 31

Skriv et program med `struct Person` som inneholder `navn` (`char`-array, 80 lang), `skoleklasse` (`char`, bokstaven A-H) og `skonummer` (`int`). Definer *en* variabel av denne structen og *en* peker til slike structer. Sett pekeren til å peke på struct-variabelen. Les så inn *alle* structens datamedlemmer vha. pekeren. Skriv til slutt ut igjen *alle* disse verdiene, *både* vha. pekeren og struct-variabelen.

NB: Det er *ikke* påkrevd at du bruker `les`…-funksjoner, eller sikrer at bokstaven/tallet er i lovlig/fornuftig område. Men, det *er* lov å legge inn slik kode også …..

Oppgave 32

Skriv et program som (er todelt):

- inneholder en `int`-array (20 lang, bruk `const`, og bruk denne *aktivt* i koden)
- en peker til `int`
- initierer hele arrayen med noen passenede verdier (*ikke* bruk peker)
- går gjennom arrayen og skriver hele dens innhold på skjermen (*ikke* bruk peker)
- setter `int`-pekeren til å peke på arrayens start
- ganger *alle* elementene i arrayen med 2 (*bruk* peker, men la den *ikke* øke/endre seg)
- skriver ut *hele* arrayens nye innhold (*bruk* peker, men la den *ikke* øke/endre seg)

- også inneholder en `char`-array/tekst med en selvvalgt lengde
- initierer denne arrayen til å inneholde en selvvalgt sekvens med blanding av bokstavene 'A', 'B', 'C', 'D' og 'E'
- skriv ut hele teksten på skjermen (*ikke* bruk peker)
- en peker til `char` som ettes til å peke på tekstens start
- går gjennom *hele* teksten (frem til '\0') og bytter *alle* 'C' med 'X' og *alle* 'E' med '8' (*bruk* peker, og la den stadig øke til neste tegn)
- går gjennom *hele* teksten *igjen* (frem til '\0') og skriver ut dens nye verdier/bokstaver (*bruk* peker, og la den stadig øke til neste tegn)

Oppgave 33

Lag funksjonene:

`int finnLengde(const char* tp)`
: som finner og returnerer lengden på teksten tilpekt av `tp`. Funksjonen skal *ikke* bruke `strlen`, men selv telle antall tegn i teksten ved å bruke pekeren.

`char* finnLengsteTekst(const char* tp, const char* tp2)`
: som finner ut (vha. den første funksjonen) hvilke av de to tilpekte tekstene (parameterne) som er lengst, og returnerer en peker til denne.

Lag også et hovedprogram som tester den andre funksjonen, ved å sende med char-arrayer (som er fylt med tekster) eller ved å sende med skrevne tekster i gåsøyne.

Oppgave 34

Lag funksjonen:

`void finnTegn(const char* t, char* f, char* m, char* s)`
: som oppdaterer de tre referanseoverførte enkelt-`char`'ene (`f`, `m` og `s`) med henholdsvis det første, midterste og siste tegnet i teksten tilpekt av `t`.
Her må gjerne `strlen` brukes.

Lag også et hovedprogram som tester denne funksjonen, ved å sende med ulike tekster og adressen til tre ulike `char`-variable.

Oppgave 35

Skriv et program som vha. pekere allokerer memory til både en `int`- og `char`-array som begge inneholder 25 elementer (indeks nr. 0-24). Fyll *hele* heltallsarrayen med noen passende tallverdier, og skiv ut dens innhold på skjermen. Fyll også hele tegnarrayen med noen passende tegn, avslutt med '\0'. Skriv ut dens innhold som en *tekst*.
Husk å frigi all allokert memory ifm. begge arrayene. Alle steder du har behov for å bruke verdien '25' - bruk en `const` for dette.

Oppgave 36

Ta utgangspunkt i koden i EKS_33.c. Skriv om denne, slik at en ny `Person` *alltid* legges inn bakerst i listen (altså First-In-First-Out, FIFO, kø/queue).

Hint 1: Lag en ny peker, `siste`, som *alltid* peker direkte til den som for øyeblikket er bakerst i listen. Dermed blir det enklere å hekte på enda en ny aller bakerst.

Hint 2: Det er *kun* behov for å skrive noe om på koden før og inni `do-while`-løkka.

Oppgave 37

Filen 'METROLOG.DTA' har følgende format for **post** (fire **felter** på *en* **linje**):
<dagnr> <min.temp> <max.temp> <nedbør>
Skriv et program som leser disse dataene inn fra filen. Programmet beregner så og skrive ut på skjermen: gjennomsnittlig minimumstemperatur, maksimumstemperatur og nedbør.
Filen består bare av heltall, men klarer du å få til at utskriften er *flyttall* med *en* desimal?

Oppgave 38

Filen 'BAATER.DTA' har følgende format for hver **post** (seks **felter** på to **linjer**):
<reg.nr for/på båten> <årsmodell> <hestekrefter> <telefonnummer> <eiers navn>
<båtens modelltype>

<reg.nr> er en tekst *uten* blanke i. Både <eiers navn> og <båtens modelltype> består av flere ord (med blanke mellom). De tre andre verdiene er `int`.

Skriv et program som:

- inneholder *en* struct-variabel for å lagre alle de seks dataene fra *en* post om *en* båt. (Denne *ene* structen blir stadig fylt med *nye* data, ettersom at dataene om en og en båt leses inn fra filen.)
- leser inn alle dataene om en og en båt fra den angitte filen
- for hver innlesning skriver ut de for øyeblikket lagrede dataene i struct-variabelen

Bruk funksjoner for selve filinnlesningen og skjermutskriften ifm. *en* struct (akkurat som i `EKS_38.c`).

Oppgave 39 (= « oblig 4.5 »)

To *totalt* uavhengige oppgaver:

a) Skriv om `EKS_35.c` til å skrive/lese `Maleri`-dataene til/fra fil.
Den globale variabelen `gSisteMaleri` skal *ikke* ligge på filen.

b) Skriv om din egen oblig nr.4 til å skrive/lese `Oppgave`-dataene til/fra fil.