OPPGAVER

for

”Grunnleggende programmering”

Høsten 2018

NTNU

Forord

Dette kompendie/hefte innholder oppgaveteksten for ulike oppgaver i emnet ”Grunnleggende programmering” ved NTNU. Disse oppgavene brukes både som en del av ukeoppgavene, men også som supplerende og frivillige ekstraoppgaver.

Opplysninger om emnet er å finne på Internet på URL’en:

http://folk.ntnu.no/frh/grprog

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

(Data)filer som oppgavetekstene henviser til er å finne under ”UKE_OPPG”. Løsningsforslag til alle oppgavene i dette heftet er å finne under ”UKE_LOSN”.

Nye/andre oppgavetekster *kan* bli laget i løpet av emnet. Disse vil i såfall etter hvert bli lagt ut under ”UKE_OPPG”, og kalt OPPG_xx.TXT. Der ”xx” står for en fortløpende nummerering fra og med ”53” og oppover.

NTNU

Frode Haug

Oppgave 1

Start opp C++ -miljøet. Eksperimenter med verktøyet, dvs. bli kjent med de ulike menyvalg. På katalogen «UKE_LOSN» finner du noen C++-filer. Hent inn en av disse ad gangen, kompiler/link (kjør) den og se hva som skjer. Endre bevisst på koden i en fil, og se hva slags feilmeldinger kompilatoren da kommer med.

Oppgave 2 **(kap.2)**

Skriv inn programmet på side 38-39 i læreboka. Kjør det. Endre eventuelt teksten til programmet er *identisk* med det i læreboka (dvs. at kompilatoren godtar det uten feilmeldinger).

Oppgave 3 **(kap.2)**

Skriv et selvstendig program som skriver ut ditt navn, gateadresse og postnr/-sted. Skriv også ut din alder (som ligger lagret i en heltallsvariabel). Klarer du å få til alt dette på hver sin linje?

NB: Tekster skal *ikke* leses inn fra tastaturet, men bare skrives inn (hardkodes) mellom gåsøyne i utskriften i programmet ditt.

Oppgave 4 **(kap.2)**

Skriv et program som leser inn tre heltall, og utfører følgende beregninger med disse:

- kvadratet av hvert av tallene
- summen av tallene
- summen av kvadratene
- gjennomsnittet av tallene
- gjennomsnittet av kvadratene

Alle resultatene skal skrives ut.

Oppgave 5 **(kap.2)**

Lag et program som:

- leser inn tre tegn
- skriver ut hvert enkelt tegns ASCII-nummer
- leser inn to tall av typen 'float'
- skriver ut heltallsdelen av de to tallene

Oppgave 6 **(kap.2)**

Lag et program med følgende output:

	Kamper	Seire	Uavgj.	Tap	Poeng
STORHAMAR	18	14	1	3	29
Lillehammer	18	12	4	2	28
Stavanger	18	10	3	5	23
VIF Hockey	18	10	2	6	22

GRATULERER MED SERIEMESTERSKAPET, STORHAMAR !!

Oppgave 7 **(kap.3)**

Lag er program som skriver ut sammenhengen mellom radie og volum for en sylinder. Høyden skal fast være 4, og radien skal variere fra 1 til 20. Skriv kun *en* sammenheng pr.linje på skjermen, og bruk for-løkke.

Oppgave 8 **(kap.3)**

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 9 **(kap.3)**

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 som hun/han 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 10 (kap.3)

Skriv et program som gjør at maskinen kan spille det gamle spillet «NIM» mot en menneskelig motstander. I dette spillet starter man med et bestemt antall pinner: 25 stk. To spillere trekker etter tur, 1-3 pinner hver gang. Den som blir tvunget til å ta den siste pinnen har tapt. Programmet skal la brukeren trekke først.

Antall pinner ved start er *en* høyere enn et multippel av 4 (25 = 1+(6*4)). Da vil vinner-strategien gå ut på å subtrahere fra 4 det antall pinner brukeren tar, og så selv ta vekk dette antall pinner. F.eks. dersom brukeren trekker 3, da skal maskinen ta 1, dersom brukeren tar 2, skal maskinen ta 2. Dette garanterer at det alltid vil være en pinne igjen når brukeren må trekke til slutt.

(Hva med å prøve dette spillet/programmet på en uvitende venn, for å vise maskinens utrolige intelligens ??!!)

Oppgave 11 (kap.3)

Utvid exercise nr.3 side 126-127 i læreboka slik at den håndterer at brukeren taster andre tegn enn lovlige siffer (0-9), og at den avslutter innlesningen om brukeren taster mer enn 9 siffer (dvs. «sprenger» nesten int-variabelen).

Oppgave 12 (kap.3)

Lag et program som skriver ut primtall, fortløpende fra 3 og oppover. Bestem selv øvre grense. (**Hint:** side 96-97 i læreboka.) Stans utskriften (og vent på et tegn/tastetrykk etterfulgt av ENTER/CR) for hver 20.linje.

Oppgave 13 (kap.3)

Lag et program som leser inn ett 8-sifret 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

Legg inn «robusthet» ved at programmet *kun* godtar sifre, og at nummeret *ikke* kan starte med '0'.

Oppgave 14 (kap.2-3)

Du skal lage en enkel kalkulator. Den skal fungere slik: Ved oppstart ligger tallet 0 i en variabel kalt «akkumulator». Du bruker kalkulatoren ved å skrive en regneoperasjon etterfulgt av et tall (på en og samme linje). Hvis du f.eks. gav operasjonen '+' og tallet 4, vil 4 bli lagt til det tallet som allerede ligger i akkumulatoren. Resultatet av regneoperasjonen legges tilbake i akkumulatoren, og til slutt skrives innholdet av akkumulatoren ut på skjermen. Dette vil da bli resultatet av regningen. Dette skjer gang etter gang, til brukeren gir stopp kommandoen 's' istedet for en regneoperasjon.

Kalkulatorprogrammet skal forstå:

- regneoperasjonene +, -, * og /
- kommandoen 't <tall>' for å lagre <tall> direkte inn i akkumulatoren
  (Dvs. tallet som evnt. er der allerede blir erstattet med <tall>.)
- kommandoen 'c' for å nullstille akkumulatoren
- kommandoen 's' for å stoppe kalkulator-programmet

Hvis noen prøver å dele med null, skal det skrives ut en feilmelding, og hvis brukeren gir noe annet enn en av de lovlige kommandoene, så skal hun/han gjøres oppmerksom på dette, og evnt. gis litt hjelp.

En kjøring av kalkulator-programmet kan f.eks. se slik ut:

```
Enkel kalkulator starter:

=   0.000

: + 4
=   4.000

: + 2.3
=   6.300

: / 3.0
=   2.100

: / 0.0
Feil!  Dele på null er tull
=   2.100

: c
=   0.000

: t 12.2
=   12.2

: - 3.100
=   9.100
```

```
: k
Feil!  Ukjent kommando gitt
=   9.100

: s
Enkel kalkulator stopper.
```

Programmet skriver start- og stoppbeskjedene, svarlinjene ('= ....') samt tegnet ':' (som betyr at brukeren skal gi en operasjon (+, -, *, /, t) og ett tall, eller en av kommandoene 'c' eller 's').

Det skal *ikke* legges vekt på robusthet. Dvs. vi regner med at brukeren er så vennlig at tall virkelig skrives som numeriske verdier (og ikke f.eks. tekst eller andre spesialtegn).

Oppgave 15 (kap.4)

Lag en struct som inneholder et ansattnummer, alder og vekt. Definer variable, les inn fra tastaturet og lagre disse opplysningene om tre personer. Beregn og skriv ut total alder og gjennomsnittlig vekt. Skriv også ut ansattnummeret til den som er eldst.

Oppgave 17 (kap.2-4)

(Denne oppgaven er en direkte avskrift av en tidligere eksamensoppgave.)

Vi skal i denne eksamensoppgaven lage et program som hjelper skøytekommentatorene til å raskt kunne få siste rundetid for en løper. Din oppgave er å lage det som står i punktene 1-9 nedenunder. Alt etter dette er kun ment som oppklarende tekst og hint.

Du skal lage et program som oppfyller følgende beskrivelse:

1. Brukeren spørres om en distanse. Lovlige er: 500,1000,1500,3000,5000 & 10000.

2. Programmet finner ut hvor mange ganger løperne vil passere målstreken på denne distansen. (Sålenge en runde er 400 m, så vil løperen på en 500 m passere målstreken to ganger, på en 1000 m passere tre ganger, ..., på en 10000 m passere 25 ganger.)

3. Sålenge ikke alle passeringstider for begge løperne er tastet inn, så går programmet og spør om:
   - løperens nummer (1 eller 2).
   - Passeringstid i minutter, sekunder og hundredel (alle er av typen int). (I det startskuddet for et par går, så starter en klokke. Det er dennes tre verdier ved passering at brukeren taster inn til programmet.)

4. For hver ny tid som tastes inn (under pkt.3), så skriver programmet ut tiden siden siste passering (i sekunder, og med hundredelene som desimaler).

5. Når en løper har fått tastet inn sin siste passeringstid (dvs. ved målpassering), så skriver programmet ut «MÅLPASSERING».

NB: Husk altså at pkt. 3-4 utføres helt til begge løperne har fått utskriften i pkt.5.

6. Når begge er i mål, så skriver programmet ut hvilken løper som vant paret (evnt. «Likt i mål», om tidene skulle være identiske), og hva denne tiden ble, på formatet: «mm:ss:hh».

7. Til sist spørres brukeren om han vil kjøre programmet en gang til for et nytt par. Dersom «ja», så gjentas alt f.o.m.pkt.3. (Husk å resette alle nødvendige variable.)

8. Programmet skal ikke sjekke etter om tall virkelig er numeriske verdier, men det skal sikre at:
   - distansen er en av de lovlige (pkt.1).
   - løperens nummer kun er 1 eller 2 (pkt.3).
   - min.er i intervallet 0-20, sekund.i 0-59 og hundred.i 0-99 (pkt.3).
   - passeringstidspunktet for en runde er minst 8 sekunder senere enn forrige passering (pkt.3).
   - det ikke tastes inn nye passeringsverdier for en løper som allerede er i mål, selvom hennes/hans parmakker ikke er kommet i mål ennå (pkt.3-5).

9. Programmet skal basere seg på:
   - endel enkeltvariable (som du selv må finne og definere)
   - struct Loper {

```
        int  nr;              // Løperens nummer:  1 eller 2
        int  antPasseringer;  // Antall ganger som løperen har passert mål.
                              //   F.eks. på 500 m vil denne gå fra 0 til 2.
                              //   På en 1500 m vil den gå fra 0 til 4.
        float  tidHittil;     // Antall sek. (inkl.hundredel) som løperen
    };                        //   hittil har brukt. Når «antPasseringer»
                              //   har blitt telt makimalt opp, så vil  denne
                              //   inneholde sluttiden.
```

Eksempel på dialog (brukersvar er understreket):

```
Distanse (500, 1000, 1500, 3000, 5000, 10000):     500
Passering av løper nummer (1-2):                   1
Passeringstid (mm ss hh):                          0 10 39
        Siste rundetid:                            10.39
Passering av løper nummer (1-2):                   2
Passeringstid (mm ss hh):                          0 10 42
        Siste rundetid:                            10.42
Passering av løper nummer (1-2):                   1
Passeringstid (mm ss hh):                          0 37 52
        Siste rundetid:                            27.13      MÅLPASSERING !!
```

Passering av løper nummer (1-2): 2
Passeringstid (mm ss hh): 0 38 15
Siste rundetid: 27.73 MÅLPASSERING !!
Løper nr.1 ble parvinner, med tiden: 0:37:52

Flere par (J/n): n

Klargjørende(?):
Du skal altså skrive et fullstendig og enkeltstående program, dvs. at det inneholder alt for å være et fungerende og riktig program. Det aller meste av den koden du skriver skal befinne seg mellom krøll-parantesene i «main».

I programmet skal:
- det ikke holdes orden på mer enn to løpere ad gangen.
- løperne «navngis» vha. tallene 1 og 2 (dvs. ikke-noe tekstlig navn).
- ingen data skal lagres (skrives til fil).
- ingen data skal leses fra fil.

Under pkt.3 skal det kun behandles *en* løper pr. runde i loopen. Brukeren skal altså ikke bli tvunget til å taste inn tiden for begge løperne pr. runde i loopen. (Dette har sin naturlige forklaring i at på lengre løp (f.eks. 10000 m) så kan det hende at en løper tar igjen en annen med en runde, og dermed vil passere først over målstreken to ganger på rad. Dermed må den innhentede løperens rundepasseringer tastes inn to eller flere ganger til slutt.)

Oppgave 18 (kap.2-3)

(Denne oppgaven er en direkte avskrift av en tidligere kontinuasjonseksamensoppgave.)

Du skal lage et program som «simulerer» et kassaapparat i en butikk. Programmet skal ta imot ulike kommandoer, evnt. kombinert med tall-input, gjøre noen enkle beregninger og skrive ut resultater.

Din oppgave er å lage det som står i pkt.1-4 nedenunder. Alt etter dette er kun ment som oppklarende tekst og hint. «Brukeren» betyr «den-som-betjener-kassaapparatet».

Du skal lage et program som oppfyller følgende beskrivelse:

1. Brukeren skal først kunne gi kommandoene 'V', 'A', 'E' eller 'S'.
   Når brukeren taster disse skal følgende skje:

   V(arekjøp): Programmet spør om og leser inn varens pris. Dette tallet adderes til totalsummen som allerede er registrert for nåværende kunde.

A(ngre): Siste inntastede varepris trekkes fra på totalsummen for nåværende kunde.

E(ndre): Programmet spør om og leser inn et beløp. Dersom dette tallet er mellom 0 og totalsummen for nåværende kunde, så trekkes det fra på totalsummen for kunden, ellers skrives en «feilmelding».

S(lutt): Varekjøpene for nåværende kunde (dvs. løkka i pkt.1) avsluttes.

2. Etter at pkt.1 er ferdig, så skriver programmet automatisk ut hva vedkommende kunde totalt har handlet for (totalsummen). Denne summen adderes til den summen det totalt er handlet for av alle kundene siden programmet ble startet.

3. Brukeren får deretter følgende menyvalg: 'T', 'R', 'N', 'Q'.
Når disse tastes, så skjer følgende:

T(ilbake): Programmet spør om et beløp større enn kundens totalsum.
(Dette er den pengesummen som kunden leverer for å betale sine varer.)
Dersom dette altså er større, så skriver programmet ut hvor mye penger kunden skal ha tilbake, ellers kommer en «feilmelding».

R(esultat): Programmet skriver hvor mye det totalt er handlet for siden programmet ble startet, dvs. summen av alle kundenes kjøp.

N(y kunde): Pkt.3 (løkka) avsluttes, og programutførelsen går tilbake til (løkka i) pkt.1.

Q(uit): Pkt.3 (løkka) og programutførelsen går til pkt.4.

4. Resultatet av alle kundenes kjøp skrives automatisk til skjermen.
Programmet venter på et tastetrykk og avsluttes deretter.

DIVERSE:

- Du skal altså skrive et fullstendig og enkeltstående program som virker.
Det aller meste av den koden du skriver skal befinne seg mellom krøll-parantesene i «main». Passende variable må du selv finne og definere.

- Husk å:
  - nullstille totalsummen for alle kundene ved programmets oppstart.
  - nullstille kundens totalsum og sist inntastede varebeløp for hver gang Det startes på pkt.1.

- Robusthet: programmet skal ikke sjekke at tall virkelig er numeriske.

- Gjør dine egne forutsetninger dersom du finner oppgaveteksten upresis eller ufullstendig. Gjør i så fall rede for disse forutsetningene.

Oppgave 20 (kap.5)

Lag et program som spør brukeren om en dato på formen: dd mm åååå (dd=dag, mm=måned, åååå=år). Dersom dette er en lovlig dato, så skriver programmet ut hvilket dagnummer i året dette er. F.eks. 06 03 1994 er dag nr. 65.

Hint:

1) Lag en funksjon som har tre heltallsvariable som parametre. Inn i disse leses verdiene for dag, måned og år. Endringene som skjer med de formelle parametrene skal også skje med de medsendte aktuelle parametrene (referanse-overføring).

2) Lag en boolsk funksjon som sjekker om et visst årstall (parameter inn) er et skuddår. Funksjonen returnerer 'true' dersom det er et skuddår, 'false' ellers. Skuddår er det i år som er delelig med 400, eller i år som er delelig med 4, men ikke med 100.
Dvs. skuddår i: 1896, 1600, 1972 og 2000, ikke skuddår i: 1800, 1833, 1934 og 1900.

3) Lag en funksjon som, ut fra inn-parametrene ”maaned” og ”aar”, returnerer antall (int) dager i denne måneden det angitte året. For å bestemme antall dager i februar (det aktuelle året) må funksjonen benytte seg av funksjonen laget i 2).

4) Lag en funksjon som tar de tre innleste verdiene fra 1) som input, og som sjekker at disse er av et lovlig format (dd = 1 - 28-31, mm = 1-12, åååå = 1600-2100). Husk at antall dager i måneden også må stemme. For å sjekke dette siste bruker du funksjonen laget i 3). Funksjonen returnerer datoens dagnummer i året (eller '0' om datoen er ugyldig).

5) Lag et hovedprogram som tester det du har laget. Dvs. tilkaller først 1) med relevante parametre, tilkaller deretter 4), og skriver ut dagnummeret (dersom dette lot seg gjøre å beregne). Programmet skal loope inntil brukeren slår inn en ulovlig dato.

Oppgave 21 (kap.5 og 7)

Lag et program som leser inn tegn inntil brukeren slår et spesielt avslutningstegn (f.eks. ‘!’). Programmet skal telle opp forekomsten av antall A'er, B'er, C'er, ..., Z'er (vi ignorerer Æ, Ø og Å). Store og små bokstaver skal telle likt.

Til slutt skriver programmet ut et histogram over forekomstene.
F.eks (en stjerne pr. forekomst):

A: ********
B: **********
...
Z: *******

Hint:

1) Bruk en int-array (26 lang) til å lagre antall forekomster.
2) Lag en funksjon som nullstiller arrayens elementer.
3) Lag en funksjon som teller opp forekomstene.
4) Lag en funksjon som skriver histogrammet.
5) Lag et hovedprogram som tester funksjonene i 2)-4).

Funksjonene i 2)-4) *skal* ha arrayen som input-parameter.

Oppgave 22 (kap.5-7)

Denne oppgaven går ut på å lage et program som holder orden på stilkarakterer, hopplengde, poengsum og navn for N hoppere. Programmet skal kun fungere for *en* omgang (dvs. *alle* hopperne hopper *kun en* gang).

Lag et program som:

- Inneholder en array med N hopper-objekter.
- For hver hopper lagrer: navn, poengsum, lengde og fem stilkarakterer.
- Leser inn navnet på disse N hopperne.
- For hver av hopperne leser inn hans lengde og fem stilkarakterer.
- Beregner poengsum for hver hopper. Dette gjøres ved:

  poengsum = lengdepoeng + stilpoeng

  der lengdepoeng = 70 + ((hopperensLengde - 80) * 1.4)

  der stilpoeng = summen av de tre karakterene som er igjen, når den minste OG største er tatt vekk.

  Hint: EKSEMPEL \ EKS_16.CPP
- Skriver ut alle hoppernes navn og poengsummer
- Sørger for at lengden ligger i intervallet 20-130, og at stil-karakterene er i intervallet 5-20.

Ekstra1:

Før utskriving av hoppernes poengsummer så sorteres arrayen (objektene) etter fallende poengsum (den med høyest poengsum kommer først).

Hint: EKS_21.CPP (til våren) inneholder en sorteringsfunksjon som er basert på for-løkkene i ”bsort” side 449 og ”order” side 186 i læreboka.

Ekstra2:

Sørg ikke bare for at tall som leses inn er i riktig intervall, men også at brukeren virkelig slår inn en *numerisk* verdi, og ikke tekst eller andre spesialtegn.

Ekstra 3:

Sørg for at hopperens navn *kun* inneholder bokstaver (A-Z, a-z) og blanke (space).

Ekstra 4:

Hoppernes navn leses inn fra fil i stedet for fra tastaturet.
Hoppernes navn og poengsummer skrives også til fil (ikke bare til skjermen).

Oppgave 23 (kap.5 og 7)

Lag et program som leser inn et personnummer (som tekst). Denne teksten sendes til en funksjon, som sjekker om teksten *virkelig* er et *lovlig* personnummer. Funksjonen returnerer med '1' dersom OK, og '0' ellers.

Det er et lovlig personnummer dersom:

a) Teksten er 11 lang (**hint:** strlen)
b) Teksten består av kun *siffer/tall* (**hint:** isdigit)
c) Datoen (de seks første sifre) finnes (**hint:** Kopier og bruk funksjoner fra OPPG_20.CPP.)
d) Sifrene 7-9 utgjør tilsammen et tall i intervallet 100-499.
e) Det 10. og 11. siffer, som er kontrollsifre, er korrekte. Beregning av kontrollsifre foregår slik: Anta at de 11 sifrene i personnummeret er p1, p2, p3, ..., p10, p11. Da skal følgende *alltid* være oppfylt:

```
(3*p1 + 7*p2 + 6*p3 + 1*p4 + 8*p5 +  9*p6 + 4*p7 + 5*p8 + 2*p9 + 1*p10)
        %  11  == 0  og
(5*p1 + 4*p2 + 3*p3 + 2*p4 + 7*p5 +  6*p6 + 5*p7 + 4*p8 + 3*p9 + 2*p10
        + 1*p11)  %  11  == 0
```

Kommentar til pkt.d): Dette er sant for mennesker født i det 20.århundre. Disse tildeles forløpende fra 499 og *nedover*. 500-749 brukes for de født på 1800-tallet. 750-999 brukes i særtilfeller som: innvandrere, adopsjon o.l. Det 9.sifret forteller kjønnet: kvinne dersom partall, mann dersom oddetall.

Oppgave 24 (kap.7)

Lag et program som inneholder:

a) En to-dimensjonal array (kalt «tall») som er 8x10.

b) To en-dimensjonale arrayer (kalt «snittLinje» og «snittKolonne») som er henholdsvis 8 og 10 lange.

c) En funksjon som tar den to-dimensjonale arrayen som input, og fyller alle dens elementer med tilfeldige tall mellom 10 og 20.

d) En funksjon som tar alle de tre arrayene som input, og som regner ut snittet av alle tallene i henholdsvis hver linje og kolonne. Resultatene lagres i de to aktuelle arrayene.

e) En funksjon som skriver ut resultatet (alle de tre arrayenes innhold). Layout'en bestemmer du selv.

f) Et hovedprogram som tester ut alt det ovenfor skrevne.

Oppgave 25 (kap.7)

Lag et program som holder orden på et husregnskap. Programmet skal gå gjennom alle månedene i året, og for hver måned spørre brukeren om hvor mye hun/han har brukt til ulike formål/poster (f.eks. husleie, mat, forsikringer, lån, kommunale avgifter, strøm, aviser/tidsskrifter, abbonement/medlemskap, ferie, forlystelser, o.s.v).

Programmet skal også oppsummere hvor mye som er brukt pr. måned og pr.post.
Output fra programmet kan f.eks. se slik ut:

	Jan	Feb	Mar	Apr	Mai	Jun	Jul	Aug	Sep	Okt	Nov	Des	Sum(post)
Husleie	2000	2000	2000	2000	2000	2000	2200	2200	2200	2200	2200	2200	25200
Mat	1500	1200	1300	1500	1400	1000	600	800	1400	1200	1300	900	14100
.													
.													
Forlyst.	300	300	250	500	800	600	300	100	200	300	600	800	5050
Sum(mnd)	3800	3500	3550	4000	4200	3600	3100	3100	3800	3700	4100	3900	44350

Hint:
- Representer alle tallene i en to-dimensjonal tabell/array.
- La arayen være: (antallposter+1) * (antall måneder+1). Der siste kolonne og linje er summen av alle de foregående tallene i vedkommende kolonne/ linje (Sum-mnd og sum-post).
- La alle tekstene ligge i to en-dimensjonale string-arrayer.
- Benytt for-løkker og arrayene (tekst og tall) til å lese inn verdier og skrive ut tabellen.

Oppgave 26 (kap.5-7)

Du skal lage et program som holder orden på et varelager. Om hver vare skal det lagres dets navn (streng/tekst), pris (float) og antall (int) igjen på lager. Dere skal lage ulike funksjoner (menyvalg) som opererer på disse dataene. I utgangspunktet lar vi varelageret maksimum inneholde 100 varer.

Gjør følgende:

1) Opprett datastrukturen. Dvs. lag en klasse for å representere en vare, og en array av objekter til å holde orden på hele varelageret. Dette medfører at element nr.'i' i arrayen, inneholder data om vare nr.'i' (dvs. vi ignorerer element nr.0).

2) Lag et hovedprogram (main), som presenterer de ulike menyvalgene (se pkt. 3), leser kommando (ønsket alternativ) fra brukeren (jfr. pkt. 4), og går i loop til brukeren velger Q (= quit/avslutt). Inni loopen skal ønsket valg (pkt.5-8) bli utført vha. switch-setning.

3) Lag en funksjon som viser brukeren menyen.

4) Lag en funksjon som ber om og leser en kommando (ett tegn) fra brukeren.

5) Lag *funksjonene* som registrerer en ny vare (N = registrer ny vare). Om hver vare skal dets navn, pris og antall leses inn. Varen legges *alltid* til slutt i arrayen. Dvs. når 'N' tastes første gang, så tas objekt nr.1 i bruk. Når 'N' tastes neste gang, så tas objekt nr.2 i bruk, osv……

6) Lag *funksjonene* som viser en totaloversikt (O = oversikt over varelageret) over varelageret (dvs. *alle* varenes nummer, navn, pris og antall skal skrives ut).
Få utskriften til å stanse for hver 20.vare, og brukeren må skrive *ett* tegn for at den skal fortsette. Til slutt skal totalverdien av hele varelagret skrives ut. (Verdien for *en* vare er antall ganger pris.)

7) Lag *funksjonene* (S = selg vare) som selger «x» stk. av en vare. Dvs. den ber om et varenummer, og om denne finnes, så ber den om et lovlig antall (fra 0 til så mange som det er av varen på lageret), som trekkes fra på varens antall.

8) Lag *en* funksjon (F = fjern vare) som fjerner en vare. Dvs. den ber om et varenummer, og fjerner denne (om den da finnes).

9) I pkt.7) og 8) er det aktuelt å be brukeren om et varenummer (fra 0 til antall varer registrert til nå). I pkt.5 skal brukeren oppgi en pris, og i pkt. 5) og 7) skal hun/han angi et antall. Lag en funksjon som har følgende parametre:
   - en ledetekst («Varenummer», «Pris» eller «Antall»)
   - tallet som er nedre godtatte verdi (MIN)
   - tallet som er øvre godtatte verdi (MAX)

   Funksjonen returnerer med en tallverdi som ligger mellom MAX og MIN.
   La funksjonen også godta verdien '0' i alle de tre tilfellene (dvs. MIN = 0 i alle tilfellene).

Oppgave 29 (kap.12)

Lag et program som skriver tilfeldige bokstaver (A-Z) til filen «BOKSTAV.DTA».
Filen *skal* inneholde 80 tegn pr.linje og 30 linjer skal lages.

Oppgave 31 (kap.12)

Filen «METROLOG.DTA» har følgende format på hver linje:
<dagnr> <min.temp> <max.temp> <nedbør>
Lag et program som leser disse dataene inn fra filen. Programmet skal finne ut:
- gjennomsnittlig minimumstemperatur
- gjennomsnittlig maksimumstemperatur
- gjennomsnittlig nedbør
- dagen med størst temperaturforskjell
- dagen med mest nedbør

Oppgave 32 (kap.12)

Hver linje i filen «BILER.DTA» består av ulike felter. Disse feltene har en fast startposisjon og dermed en maksimumslengde. Alt dette er gitt ved:

Felt:	<årstall>	<reg.nr>	<biltype>	<navn>	<gate>	<poststed>
Pos:	0	6	16	38	60	82
Lengde:	4	8	20	20	20	20
	alltid	alltid	max	max	max	max

Dvs. det er alltid to blanke mellom hvert av feltene.

Lag et lite program som henter en og en linje, og deretter leser hvert enkelt felt inn i enkelt-variable. Skriv ut disse variablenes innhold, før du leser en ny linje.

Hint:
- Bruk ”getline” til å hente hver enkelt linje.
- Bruk ”strncpy” (cstring) til å hente ut de enkelte feltene.
- Bruk ”atoi”» (cstdlib) til å hente ut heltall.
- Dersom hele den innleste linjen ligger i en variabel som kalles ”buffer”, så vil f.eks. ”navn”-feltet (og resten av linja) være en tekststreng som starter i ”buffer+38”.

Oppgave 48 (kap.6)

Lag et program med de to klassene:

1. ”Posisjon”- som inneholder:
   - datamedlemmene: int’ene ”grad” (0-90 eller 0-180), ”minutt” (0-59) og ”sekund” (0-59), samt char’en ”retning” (’N’/’S’ eller ’E’/’W’).
   - en privat funksjon (”int les(int min, int max)”) som leser inn en et tall, og sikrer at denne ligger mellom ”min” og ”max”, før den returnerer dets godtatte verdi.
   - en constructor som initierer/nullstiller alle datamedlemmene.
   - en constructor som setter alle datamedlemmene lik de fire verdiene som kommer inn som parametre.
   - funksjonen (”void les (int gr, char r1, char r2)”) som sørger for at alle data blir lest inn fra brukeren, og sikrer at de tilordnes korrekte verdier (”grad” er 0-’gr’, ”minutt” og ”sekund” er begge 0-59, ”retning” er ’r1’ eller ’r2’).
   - funksjonen (” void skriv()”) som skriver ut datamedlemmene på eksempelvis formen: 76º23’12” N
   .
2. ”Skip” – som inneholder:
   - datamedlemmene: ”breddegrad” og ”lengdegrad” (begge av typen ”Posisjon”), samt int’en ”nr”.
   - en constructor som setter ”nr” lik en parameter som kommer inn.
   - funksjonen (” void les()”) som leser inn skipets breddegrad (max. 90, ’N’ og ’S’) og lengdegrad (max. 180, ’E’ og ’W’) vha. andre ”les”-funksjonen i ”Posisjon”.
   - funksjonen (” void skriv()”) som sørger for at alle skipets data skrives ut. Denne bruker bl.a. ”skriv”-funksjonen i ”Posisjon”.

Lag til slutt et hovedprogram som oppretter tre ulike Skip-objekter, og som sørger for at alle deres data både blir lest inn og skrevet ut igjen til skjermen.

Oppgave 49 (kap.3)

Lag et program som skriver ut *alle* tallene fra 1 til 45. Dersom tallet er heltallig delelig med 3, så skrives *i stedet* ”Pompel”. Er det delelig med 5 skrives *i stedet* ”Pilt”, mens er det delelig med begge tallene skrives *i stedet* ”Pompel & Pilt”.

Oppgave 50 (kap.3)

Lag et program som går fra 24 og ned til 1. Ut fra hva tallet er, så skriver det følgende:

24 brusflasker i kassen, 24 flasker med brus.
Ta en og drikk den opp, 23 flasker igjen i kassen.
23 brusflasker i kassen, 23 flasker med brus.
Ta en og drikk den opp, 22 flasker igjen i kassen.
< Gjentar seg ned t.o.m. 3. På slutten skrives:>
2 brusflasker i kassen, 2 flasker med brus.
Ta en og drikk den opp, 1 flaske igjen i kassen.
1 brusflaske i kassen, 1 flaske med brus.
Gå til butikken og kjøp mer, 24 flasker igjen i kassen.

Legg spesielt merke til de to siste utskriftenes (for 2 og 1) ulikhet ift. alle de foregående linjene, med hensyn til bruken av ordet ”flaske(r)” den *aller* siste utskriftslinjen.

Oppgave 51 (kap.7)

Lag et program som foretar Cæsar-kryptering på en innlest tekst fra brukeren. Algoritmen for denne krypteringsformen er meget enkel: hvert enkelt tegn i hele teksten erstattes med et annet tegn som kommer et fast antall tegn etter (evt. før) vedkommende tegn i alfabetet.
F.eks: om teksten er ”ARSENAL” og alle tegn erstattes med tegnet som kommer tre plasser lengre ut i alfabetet, så blir den krypterte teksten ”DUVHQDO”.
Husk på at om tegnet som skal krypteres, ved å få det aktuelle tillegget, kan havne utenfor/etter alfabetet. Da må det ”wrappes” om, og starte på begynnelsen igjen av alfabetet.
F.eks: Teksten ”STUDENT” med et tillegg på syv gir ”ZABKLUA”.
Programmet du skriver skal altså:

- Lese inn en tekst (max. 80 lang, kun inneholdende bokstavene ’A’-’Z’ (dette siste trenger du evt. *ikke* å sjekke).
- Gjøre hele teksten om til store bokstaver.
- Lese Cæsar-keyen – et tall mellom 0 og 25 som teksten skal krypteres med (dvs. det ”tillegget” hver enkelt bokstav skal få for å bli kryptert).
- Foreta Cæsar-krypteringen på hele den innleste teksten.
  Hint: Ved ”wrapping” bør det brukes modulus (%)-operatoren.
- Skrive ut på skjermen den krypterte teksten/meldingen.

Oppgave 52 (kap.3)

Lag et program som fortløpende skriver 10 og 10 tall fra 1 og oppover, til brukeren ikke ønsker dette lengre. Det *skal* brukes en ”evig” for-løkke til dette. Det enkleste er nok å løse/kode dette bl.a. ved bruk av ’%’ (modulus) og ’break’. Lag først en slik versjon. Lag deretter kode (også primært vha. for-løkke) der dette *ikke* brukes.