**Mitthögskolan ITM**
Institutionen för Informationsteknologi
och medier.

Jan-Erik Jonsson 060-14 87 90


# ¾ ¾ Laborationskompendium del2 ¾ ¾

## Datakunskap NT, DTAA04
## Grundläggande datavetenskap, DTAA90

¾ ¾ ¾ ¾ ¾ ¾ ¾ ¾ ¾ ¾ **v1.3 2002-05-30** ¾ ¾ ¾ ¾ ¾ ¾ ¾ ¾ ¾ ¾ ¾

# Programmering.

I den här kursen ska du lära dig grundläggande programmering. Det är inte samma sak som att du lär dig ett programspråk. Men för att lära dig programmering så måste du programmera och för att programmera måste du kunna ett programspråk. Du måste också lära dig använda verktygen som behövs för att skriva, redigera och felsöka din programkod (källkod), och till slut göra körbara program av det hela. Det är detta som dessa laborationer handlar om.

## Om testning:

Det är du själv som så långt som möjligt ska testa ditt program, inte handledaren. Det sparar tid både för dig och för handledaren. Om du får problem ska du förstås be handledaren om hjälp.

På flera av uppgifterna finns det tips på vad du bör testa. De finns som en hjälp för att du ska komma igång med hur man gör för att testa ett program. Att lära sig testa och felsöka program är en viktig del i att lära sig programmering.

## Programutvecklingsmiljö:

I denna kurs används utvecklingsmiljön Microsoft Visual C++ 6.0, förkortat MSVC++. Det är en vanlig miljö i professionella sammanhang. Den finns även som studentlicens för under tusenlappen.

Ett alternativ till dessa, som är gratis, finns att ladda hem från nätet på http://www.delorie.com/djgpp. Det finns dessutom en gratis grafisk utvecklingsmiljö som heter Dev-C++ på http://www.bloodshed.net/devcpp.html.

## Redovisning:

Förutom det som står i varje uppgift så gäller det som står i Del 1 av laborationskompendiet.

# Laboration 4. In/ut-matning, beräkning och styrning.

## Uppgift 4a. Utmatning.

### 4a1. Att skriva, kompilera och köra program.

Skriv ett program som består av nedanstående kod. Radnumren längst till höger ska inte vara med förstås, de finns för att det ska bli enklare att beskriva programmet.

```
1: #include <iostream>
2: using namespace std;
3: void main()
4: {
5:    cout << "Hello World";
6:
7:
8: }
```

Kompilera och kör programmet. När du fått programmet att fungera så ska det skriva ut:

```
Hello WorldPress any key to continue
```

Du ska nu ändra i koden enligt nedanstående. Efter varje förändring ska du testa programmet och beskriva vad som händer.

Prova och experimentera gärna med andra ändringar. Kom bara ihåg att det är de nedanstående som ska redovisas. Det gör inget om det blir fel, det absolut värsta som kan hända är att datorn kraschar. Det är bara att starta om i så fall.

1. Ändra rad 5 till `cout << "Hello World" << endl;`
2. Ändra rad 5 till `cout << "Hello World\n";`
3. Ändra rad 5 till `cout << "Hello Earth" << " and Mars to." << endl;`
4. Ändra rad 5 till `cout << "Hello Earth"`
   Ändra rad 6 till `<< " and Mars to." << endl;`
5. Ändra rad 5 till `cout << "Hello Earth";`
   Ändra rad 6 till `cout << " and Mars to." << endl;`
6. Radera rad 6 och ändra rad 5 till `cout << endl << "Hello World" << endl;`
7. Ändra rad 5 till `cout << endl << "ääö ÅÄÖ" << endl;`
8. Ändra rad 5 till `cout << "\n\n\n\n\tEmpty\n\taround\n\n\n\n\n";`
9. Ändra rad 5 till `cout << endl << 2+3 << endl;`
10. Ändra rad 5 till `cout << endl << "2+3 = " << 2+3 << endl;`

Spara koden som den ser ut efter sista förändringen eftersom du behöver den i nästa uppgift.

#### Redovisning:

Beskriv i laborationsrapporten vad som händer (eller inte händer) efter varje förändring.

## 4a2. God programeringsstil.

Nu ska du lägga till ett huvud i din källkod från föregående exempel. Se i exemplet nedan. Huvudet ska läggas till allra överst i programkoden. Detta påverkar inte funktionen hos själva programmet men ger upplysningar som är viktiga. Alla program du skriver hädanefter ska ha ett sådant huvud.

```
// fil-namn:        svesv01_4a_v2.cpp
// Skrivet av:      Sven Svensson
// Skapat datum:    2001-08-31
// Senast ändrat:   2001-08-31
// Kurs:            Grundläggande Datavetenskap
// Handledare:      Anders Andersson
// Beskrivning:     Programmet skriver ut . . . .

#include <iostream>
osv….
```

I god programmeringsstil ingår det förstås mycket mer än att skriva ett huvud i källkoden. Det ska t.ex. vara en riktig indentering (indrag/tabbar) och bra namn på variabler m.m. Du ska också skriva kommentarer i källkoden till sådant som kan vara svårt att förstå direkt hur det är tänkt att fungera.

### Redovisning:

Bifoga din källkodsfil tillsammans med rapporten.

## Uppgift 4b. Inmatning och utmatning, och enkla beräkningar.

### 4b1. In och utmatning.

Skriv ett program som består av nedanstående kod.

```
#include <iostream>
using namespace std;

void main()
{
   int varde;

   cout << "\nMata in ett flyttal(\"decimaltal\"): ";
   cin >> varde;

   cout << "Du matade in: "
      << varde << endl;
}
```

Kör programmet. Tänk på att det ska vara punkt i decimaltalet inte kommatecken när du matar in värden. Exempel: `10,7` är fel, `10.7` ska det vara. (Tänk på engelska, ”ten *point* seven”).

1. Programmet fungerar inte riktigt bra, varför?
2. Ändra programmet så det skriver ut rätt värde.
3. Ändra programmet så att det alltid skriver ut precis 4 decimaler.

#### Redovisning:

Besvara fråga 1 i rapporten.
Bifoga din slutgiltiga källkodsfil tillsammans med rapporten.

### 4b2. Ränta.

För att beräkna årsräntan i kronor och ören på ett belopp använder man formeln:

$$\text{ränta} = (\text{räntesats}/100) \cdot \text{belopp}$$

Gör ett program som först läser in ett belopp i hela kronor och beräknar den ränta man får på ett år. Programmet ska sedan skriva ut räntesatsen, räntan i hela kronor, och totalbeloppet i hela kronor.

Räntesatsen är en konstant som måste ändras mellan varje exekvering, det vill säga, du måste ändra i din programkod för att ändra räntesatsen. Använd räntesatsen 1,69%.

Skriv programmet så att du bara behöver ändra på ett ställe om du vill ändra räntesatsen.

#### Exempel på användardialog:

```
Mata in belopp (i kr): 1000
Rantesatsen ar: 1.69 %
Din ranta blir: 17 kr
Totalbeloppet blir: 1017 kr
Press any key to continue
```

#### Testning:

Prova att mata in olika värden. Kontrollera att programmet räknar och avrundar rätt.

#### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

### 4b3. Mer än en inmatning.

Kopiera källkoden från det förra programmet och ändra den så att man nu ska mata in två värden, både räntesatsen och beloppet.

Programmet ska inte längre bara arbeta med hela kronor.

Programmet ska skriva ut räntesatsen, räntan och slutbeloppet.

Exempel på användardialog:

```
Mata in rantesatsen (i %): 1.69
Mata in belopp (i kr): 1000
Rantesatsen ar: 1.69 %
Din ranta blir: 16.90 kr
Totalbeloppet blir: 1016.90 kr
Press any key to continue
```

Testning:
Prova att mata in olika värden. Kontrollera att programmet räknar rätt.

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Uppgift 4c. Omvandlingar.

### 4c1. Från meter till tum.

Skriv ett program som läser in ett värde i meter (inte bara hela meter!) och omvandlar det till hela fot och hela tum. Se den tidigare laborationen i Excel.

Programmet ska skriva ut antal fot och tum samt avvikelsen i meter. (Det blir en avvikelse därför att programmet arbetar med hela fot och tum.)

Testning:
Se tabellen i Excel-uppgiften för testvärden. Du ska få exakt samma svar.

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

### 4c2. Från Celsius till Farenheit.

Skapa ett program som omvandlar temperatur angivet i Celsius till Farenheit enligt nedanstående formel.

$$f = \left( \frac{9}{5} \cdot c + 32 \right)$$

Tips:
5 och 5.0 inte alltid är samma sak!

Testning:
Några lämpliga testvärden:

| | | | |
|---|---|---|---|
| | -32°C | = | -25,6°F |
| | 20°C | = | 68°F |

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Uppgift 4d. Modulus, andra operatorer och styrsatser.

### 4d1. Värden och variabler.

Skriv ett program som består av följande kod.

```
#include <iostream>
using namespace std;

void main()
{
   short int a=33, b=330, c=3300, d=33000;

   cout << "a= " << a << endl;
   cout << "b= " << b << endl;
   cout << "c= " << c << endl;
   cout << "d= " << d << endl;
}
```

1. Vad skriver programmet ut?
2. Varför blir det så?

Redovisning:
Besvara frågorna i rapporten.

## 4d2. Udda eller jämt, styrsatser.

Skriv ett program som läser in ett heltal och som kontrollerar om talet är udda eller jämt.

### Exempel på användardialog:

```
Skriv in ett heltal: 13
13 ar ett udda tal.
Press any key to continue
```

### Testning:

Prova att mata in olika värden, prova även med 0 och negativa värden.

### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## 4d3. Heltalsdivision.

Skriv ett program som läser in två heltal och som utför en heltalsdivision. Det första talet ska vara täljare och det andra talet nämnare.

Programmet ska skriva ut resultatet av heltalsdivisionen och resten(modulus).

Programmet ska klara av inmatning av alla sorters heltal utan att krascha. Det behöver inte klara av hur stora tal som helst.

### Exempel på användardialog:

```
Det har programmet utfor heltalsdivision.
Skriv in taljaren: 8
Skriv in namnaren: 3
8/3 = 2 med rasten 2.
Press any key to continue
```

### Testning:

Prova att mata in olika värden, prova även med 0 och negativa värden.

### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

### 4d4. Sedlar och mynt.

Du ska nu skriva ett lite längre program än de som du skrivit förut. Det är inget nytt i jämförelse med de tidigare uppgifterna, men du ska nu kombinera flera av dina erfarenheter i ett och samma program.

Skriv ett program som beräknar hur mycket växel man skall få tillbaka när man har handlat. Det ska sedan skriva ut hur stor växeln blir och i vilka sedlar och mynt växeln ska betalas ut.

Programmet ska läsa in hur mycket man handlat för och hur mycket man betalat. Det ska också kontrollera att man betalat tillräckligt.

Alla belopp är i hela kronor.

Följande sedelvalörer finns: 1000, 500, 100, 50 och 20 kr.
Föjande mynt finns: 10, 5 och 1 krona.

#### Exempel på användardialog:

```
Hur mycket har du handlat for (i hela kr): 273
Hur mycket har du betalat (i hela kr): 2000
Vaxeln blir: 1727 kr
tusenlappar: 1 st
femhundralappar: 1 st
hundralappar: 2 st
tjugolappar: 1 st
femkronor: 1 st
enkronor: 2 st
Press any key to continue
```

#### Testning:

Prova att mata in olika värden. Kontrollera att programmet räknar rätt.

#### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Laborationsrapport 4.

Förutom rapporten ska du skicka dina källkodsfiler till handledaren. Du ska inte skicka med några andra filer.

#### Namn på källkodsfilerna:

Källkodsfilerna som du lämnar till handledaren ska ha filnamn som ser ut så här:
`användarnamn_uppgiftsnummer_xx.cpp`
Där `användarnamn` är ditt användarnamn och `xx` är ett versionnummer. `Uppgiftsnummer` är t.ex. 4a, 5b2 och så vidare.

Exempel: `svesv01_4b2_v1.cpp`

I övrigt gäller samma som i föregående laboration.

# Laboration 5. Mera styrsatser och iterationer.

I den här laborationen ska du lära dig mer om att styra programflödet. Att styra i vilken ordning och hur många gånger olika saker ska utföras.

## Uppgift 5a. Rabatter.

### 5a1. Enkel rabatt.

Gör ett program som beräknar rabatten på ett inköpspris. Rabattsatsen är 5 % om inköpspriset är mindre än 100 kr. Från och med 100 kr är den 20%.

Programmet ska inte bara arbeta med hela kr.

Testning:
Testa med olika värden. Fungerar programmet även vid precis 100 kr?

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

### 5a2. Flera rabatter.

Kopiera och modifiera tidigare program så det blir lite mer realistiskt.

Följande rabattsatser gäller:
0% upp till 250 kr, 15% från och med 250 kr upp till 2 500 kr och 25% från och med 2500 kr.

Testning:
Testa med olika värden. Fungerar programmet även vid precis 250 och 2500 kr?

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Uppgift 5b. Iterationer (loopar, slingor).

En av IBMs högsta chefer på 50-talet trodde att världen bara behövde tre datorer för beräkning av tabeller. Han hade lite fel, men vi kan låta vår dator skapa några tabeller.

### 5b1. Multiplikationstabell.

Gör ett program som skapar och skriver ut multiplikationstabellerna mellan 1 och 9.

Exempel på utskrift:

```
1*1=1  1*2=2   1*3=3   1*4=4   1*5=5   1*6=6   1*7=7   1*8=8   1*9=9
2*1=2  2*2=4   2*3=6   2*4=8   2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*1=3  3*2=6   3*3=9   3*4=12  3*5=15  3*6=18  3*7=21  3*8=24  3*9=27
4*1=4  4*2=8   4*3=12  4*4=16  4*5=20  4*6=24  4*7=28  4*8=32  4*9=36
5*1=5  5*2=10  5*3=15  5*4=20  5*5=25  5*6=30  5*7=35  5*8=40  5*9=45
6*1=6  6*2=12  6*3=18  6*4=24  6*5=30  6*6=36  6*7=42  6*8=48  6*9=54
7*1=7  7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49  7*8=56  7*9=63
8*1=8  8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64  8*9=72
9*1=9  9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
Press any key to continue
```

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Uppgift 5c. Beräkna n!

Skriv ett program som beräknar fakulteten. Fakultet kan definieras på olika sätt, här är en definition:

$$n! = \begin{cases} 1 & \text{då } n=0 \\ 1\,2\,3 \cdots n & \text{då } n>0 \end{cases}$$

Några exempel:

0! = 1, 1! = 1, 5! = 1 2 3 4 5 = 120
13! = 1 2 3 4 5 6 7 8 9 10 11 12 13 = 6 227 020 800

Programmet ska klara av att beräkna fakulteten upp till 13, så välj rätt datatyp.

### Tips:

Fakulteten för 0 är ett specialfall.

### Exempel på användardialog:

```
Detta program beräknar fakulteten.
Ange ett tal: 5
Fakulteten för 5 = 120
Press any key to continue
```

### Testning:

Kontrollera med olika värden om programmet räknar rätt. Klarar det av även 0, 1 och 13?

### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Uppgift 5d. Iterationer och fält.

### 5d1. Samla alla negativa tal.

Gör ett program som först läser 10 värden från tangentbordet.

Sedan när alla värden är inlästa ska programmet skriva ut de som är negativa.

Testning:
0 är inte ett negativt tal.

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

### 5d2. Styrdata bland indata.

Gör ett program som läser in godtyckligt antal positiva värden från tangentbord och skriver ut summan och medelvärdet. Godtyckligt innebär att det inte ska finnas någon begränsning på hur många värden som ska kunna matas in.

Inläsningen ska avslutas när programmet läst in ett negativt värde (dvs. negativt tal fungerar som stoppmarkering).

Det negativa värdet ska inte tas med beräkningen av summan och medelvärdet.

Testning:
Programmet ska klara att läsa in hur många värden som helst. 0 är inget negativt tal.

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Laborationsrapport 5.

Förutom rapporten ska du skicka dina källkodsfiler till handledaren. Du ska inte skicka med några andra filer.

I övrigt gäller samma som i föregående laboration.

# Laboration 6. Uppdelning av program, funktioner.

## 6a1. Medelvärde.

Skriv ett program som består av följande kod.

```
1:  #include <iostream>
2:  using namespace std;
3:
4:  // Beräknar medelvärdet av två tal
5:  double medel(double, double);
6:
7:  void main()
8:  {
9:     double a, b, c;
10:    cout << "Detta program beräknar medelvärdet av två tal" << endl;
11:
12:    cout << "Mata in tal 1: "; cin >> a;
13:    cout << "Mata in tal 2: "; cin >> b;
14:
15:    c = medel(a, b);
16:
17:    cout << "Medelvärdet av " << a << " och " << b << " = " << c << endl;
18: }
19:
20: // Funktionen beräknar medelvärdet av de två inparametrarna
21: // och returnerar resultatet.
22: double medel(double in1, double in2)
23: {
24:    double svar;
25:    svar = (in1 + in2)/2;
26:    return svar;
27: }
28:
```

I detta program finns två funktioner, den ena heter `medel`, och den andra heter `main`. Du har alltså redan skrivit flera program med en funktion, nämligen `main`. I denna laboration ska du nu skriva program som innehåller fler än en funktion, men först ska du ändra lite i ovanstående program. Om du gör rätt så ska programmet fortsätta att fungera som förut.

1. Testkör först programmet precis som det står.
2. Ta bort variabeln `c` på rad 9.
   Ta bort rad 15.
   Byt ut variabeln `c` på rad 17 mot `medel(a, b)`.
3. Testkör programmet. Fortsätt inte på nästa punkt förrän det fungerar.
4. Ta bort rad 24 och 25.
   byt ur variabeln `svar` på rad 26 till `((in1 + in2)/2)`.
5. Testkör programmet.

Du har nu skrivit om programmet, men det fungerar likadant. Lösningen är varken bättre eller sämre än den ursprungliga. Den kan vara svårare att förstå om du är ovan att läsa programkod men när du blir mer van kommer du istället att tycka att den är lättare att läsa. En annan fördel med den nya koden är att den sparar (lite) minne.

### Redovisning:

Beskriv i rapporten programflödet i det ursprungliga programmet (dvs *utan* ändringarna).
Bifoga din källkodsfil (den *med* ändringarna) tillsammans med rapporten.

## 6a2. Funktioner.

Generellt kan alla program delas upp i tre delar: inläsning, bearbetning, presentation. Du ska nu skriva ett program där varje del läggs i var sin funktion.

Gör ett program som innehåller funktionerna `LasIn`, `Addera` och `SkrivUt`.

| | |
|---|---|
| `LasIn` | Läser in ett tal från tangentbordet och returnerar det.<br>Lägg in en lämplig användardialog för inläsningen i funktionen.<br>Funktionen ska inte ha några inparametrar, och returtypen ska vara av typen `int`. |
| `Addera` | I den här funktionen ska två tal adderas.<br>Funktionen ska ha två inparametrar av typen `int` och returtypen ska var av typen `int`.<br>De två inparametrarnas värden adderas till varandra och resultatet returneras. |
| `SkrivUt` | Skriver ut resultatet, dvs. värdet i inparametern, tillsammans med en lämplig text.<br>Funktionen ska en inparameter av typen `int` och returtypen ska vara av typen `void`. |

Skriv ett huvudprogram (dvs en `main`-funktion) som med hjälp av de tre funktionerna läser in två heltal från tangentbordet, adderar dessa och skriver ut resultatet.

### Testning:

Prova att programmet räknar rätt.

### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Uppgift 6b. Kontroll av indata.

Det är ofta lämpligt att kontrollera vad man får för data (värden, tecken m.m.) *samtidigt* som man läser in data. Ett vanligt sätt att lösa detta är att ha en funktion som läser in data till en referensparameter och som returnerar värdet 0 om allt gick enligt planerna. Så fort ett felaktigt värde blir inläst returneras ett värde skilt ifrån 0. På det viset kan en felinmatning hanteras där funktionen anropas. Om man kontrollerar returvärdet där man anropar funktionen kan man avgöra om något felmeddelande ska skrivas ut och om funktionen ska anropas igen.

### 6b1. Kontroll av tecken.

Gör en funktion som kontrollerar om *ett inläst tecken* är en siffra. Om tecknet är en siffra returneras heltalet noll, annars returneras tecknets ASCII-kod.

Döp gärna funktionen till `teckenkoll`. Du kan förstås döpa den till något annat. Namnet är valt bara för att det stämmer med exemplet nedan.

Exempel på en `main`-funktion som använder `teckenkoll`-funktionen:

```
.
.
void main()
{
   char tecken;

   cout << "Skriv in en siffra: ";    cin >> tecken;

   while ( teckenkoll(tecken)!=0 )
   {
      cout << "Du skrev inte en siffra, försök igen:";
      cin >> tecken;
   }

   cout << "Nu skrev du en siffra" << endl;
}
.
.
```

I exemplet ovan anropas `teckenkoll`-funktionen i `if`-satsen. På det sättet kan kontrollen göras på samma ställe som man bestämmer om ett nytt värde ska läsas in.

Testning:
Använd ovanstående `main`-funktion eller skriv en egen och testkör programmet.

Tips:
Tänk på att en siffra också kan ses som ett tecken. Och precis som bokstäver, &-tecken och andra tecken så har då varje siffra en ASCII-kod.

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten. Källkoden ska även innehålla en `main`-funktion så att din funktion kan testas.

## 6b2. Sifferkontroll av sträng.

Skriv en funktion som kontrollerar att en sträng består av bara siffror. Om det är så ska funktionen returnera en nolla, annars ska den returnera ett tal skilt ifrån noll.

Eventuell inläsning av strängen ska inte göras i funktionen. Det ska i så fall göras före anrop av funktionen.

### Tips:

Det *kan* vara enklare att kontrollera om *något* tecken *inte* är en siffra än att kontrollera om *alla* är siffror.

Du kan använda din funktion från föregående uppgift. Det går utmärkt att anropa en funktion i en annan funktion. Det har du ju gjort tidigare när du har gjort anrop ifrån `main`-funktionen.

### Testning:

| | | | | |
|---|---|---|---|---|
| Några testdata: | ”1” | ”a” | ”qw1” | ”1qw” |
| | ”q1w” | ”1q2” | ”123” | ”qwe” |

### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten. Källkoden ska även innehålla en `main`-funktion så att din funktion kan testas.

## 6b3. Inläsning med gränsvärden.

Skapa en funktion som läser in ett tal (hel eller flyttal). Funktionen ska kontrollera om det inlästa värdet är mellan ett min och ett max-värde (hel eller flyttal).

Funktionen ska minst ta två inparametrar där man kan ange min- och max-värde.

Om talet är mellan (eller lika med) min och max ska funktionen returnera `true`. Om det är utanför gränserna ska funktionen returnera `false`.

Det inlästa värdet ska alltid överföras via en referensparameter till den *anropande* funktionen.

### Testning:

Skriv en `main`-funktion så att du kan testa din funktion.

### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten. Källkoden ska även innehålla en `main`-funktion så att din funktion kan testas.

## Uppgift 6c. Menyprogram.

Här ska du skriva om några program som du redan gjort i tidigare uppgifter till att fungera som funktioner istället. Till slut ska du samla ihop dem till ett menyprogram.

### 6c1. n! igen.

Skriv en funktion som beräknar fakulteten.

All inmatning och utskrift ska göras i funktionen. Det vill säga att både beräkningen och hela användardialogen ska göras i funktionen. Om du döper funktionen till `fakultet` så ska den kunna användas som i exemplet nedan.

Funktionen ska inte ha några inparametrar och inte returnera något.

#### Exempel på en `main`-funktion som använder `fakultet`-funktionen

```
.
.
void main()
{
   fakultet();
}
.
.
```

#### Tips:

Titta på uppgift 5c.

#### Testning:

Titta på uppgift 5c.

#### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit din funktion.

### 6c2. Temperaturen igen.

Skriv en funktion som omvandlar en temperatur angiven i Celsius till Farenheit.

På samma sätt som i 6c1 ska all inmatning och utskrift göras i funktionen. Det vill säga att både beräkningen och hela användardialogen ska göras i funktionen.

Funktionen ska inte ha några inparametrar och inte returnera något.

#### Tips:

Om du vill kan du fortsätta att skriva dessa funktioner i samma program som funktionerna i de föregående uppgifterna. Du ska ändå lägga in alla i samma program i nästa uppgift.

Titta även på 4c2.

#### Testning:

Titta på 4c2.

#### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit din funktion.

### 6c3. Temperaturen ännu en gång.

Skriv en funktion som omvandlar en temperatur angiven i Farenheit till Celsius. I övrigt gäller samma som i 6c2.

#### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit din funktion.

## 6c4. Ett menyprogram.

Skapa ett menyprogram. Användaren ska kunna välja vilken av funktionerna, som du skapat tidigare i den här laborationen, som ska köras.

Användaren ska erbjudas en meny där det går att välja funktion.

I menyn ska det finnas ett alternativ för att avsluta. Så länge som användaren *inte* väljer avsluta ska programmet åter erbjuda menyn.

Om användaren skriver in ett alternativ som inte finns, ska ett felmeddelande skrivas ut och programmet ska åter erbjuda menyn.

### Exempel på användardialog:

```
I detta program kan du valja olika beräkningar:
   1 Berakna fakultet.
   2 Omvandling fran Celsius till Farenheit.
   3 Omvandling fran Farenheit till Celsius.
   4 Avsluta programmet.
Valj ett alternativ mellan 1 och 4: 2

Skriv in det antal grader Celsius du vill omvandla: -32
-32.0 grader Celsius = -25.6 grader Farenheit

   1 Berakna fakultet.
   2 Omvandling fran Celsius till Farenheit.
   3 Omvandling fran Farenheit till Celsius.
   4 Avsluta programmet.
Valj ett alternativ mellan 1 och 4:
```

### Testning:

Kontrollera att alla alternativen fungerar och räknar rätt.

### Observera:

Det är *inte* tillåtet att anropa `main` eller en eventuell meny-funktion (som i sin tur anropar de andra funktionerna igen!!) ifrån de andra ingående funktionerna. Om du gör det kommer det hela tiden att startas nya kopior av funktionerna men aldrig avslutas några. Programmet kommer då att ta upp mer och mer plats i minnet för varje menyval.

### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Uppgift 6d. Slumpen.

Du ska nu använda några standardfunktioner som finns för att generera slumptal. Tänk på att ge slumpen ett frö.

### 6d1. Simulera en vanlig tärning.

Gör ett program som använder funktionen rand() eller random() för att simulera kast med sexsidig en tärning.

Användaren ska kunna skriva in antalet kast.

Programmet ska skriva ut antalet 1:or, 2:or, etc. och den procentuella andelen för respektive siffra.

Testning:
Blir den sammanlagda procenten för alla siffrorna 100%?
Blir fördelningen jämn mellan siffrorna?

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

### 6d2. Skapa en lottorad.

Gör ett program som skapar en slumpmässig lottorad. En lottorad består av 7 ordinarie nummer och 4 tilläggsnummer. Talen ska ligga i intervallet 1-35 och samma nummer får inte förekomma mer än en gång per fullständig rad.

Testning:
Blir det några upprepningar i siffrorna?
Blir raderna olika om du kör programmet flera gånger?

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

### 6d3. Skapa en hel speltalong med lottorader.

Förbättra föregående program genom att skapa en hel speltalong med lottorader.

Användaren ska kunna välja antalet lottorader i talongen.

Testning:
Blir varje rad olika i talongerna? Blir talongerna olika om du kör programmet flera gånger?

Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Laborationsrapport 6.

Förutom rapporten ska du skicka dina källkodsfiler till handledaren. Du ska inte skicka med några andra filer.

I övrigt gäller samma som i föregående laboration.

# Laboration 7. Blandade uppgifter.

## Uppgift 7a. Beräkna primtal.

Ett primtal är ett positivt heltal, större än 1, som *bara* är jämnt delbart med sig själv och 1.
En hjälp är att ett tal är ett primtal om det *inte* är jämnt delbart med *något* mindre primtal.

Skriv en funktion som beräknar de första 50 primtalen och lägger dem i en tabell (vektor, fält).

Skriv en till funktion som skriver ut vektorn med primtalen. Du ska förstås också skriva en `main`-funktion. Du får naturligtvis dela upp programmet i ännu fler funktioner.

### Tips:
Det första primtalet är 2. Är nästa jämt delbart med 2?

### Testning:
De första 3 primtalen är 2, 3, 5. Det 50:e primtalet är 229.

### Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Uppgift 7b. Ett värde styr ett annat.

Skriv ett program som räknar ut summan av de priser(flyttal) vars artikelnummer(heltal) slutar på 0.

Inläsningen avslutas när ett negativt artikelnummer matas in.

### Exempel på användardialog:

```
Summerar priset for varor, vilkas artikelnummer slutar pa noll.
Inmatningen avslutas genom att skriva ett negativt artikelnummer.

Ange varans artikelnummer: 10
Varans pris? 10
Ange varans artikelnummer: 20
Varans pris? 10
Ange varans artikelnummer: 23
Varans pris? 10
Ange varans artikelnummer: 1
Varans pris? 10
Ange varans artikelnummer: 0
Varans pris? 10
Ange varans artikelnummer: -15

Summan blir 30.00 kr
Press any key to continue
```

### Tips:
Tal som slutar på 0 är jämt delbara med…

### Testning:
Programmet ska klara av att läsa in hur många värden som helst. 0 är inget negativt tal.

### Redovisning:
Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

## Uppgift 7c. Personnummer.

Så här beräknar du kontrollsiffran (sista siffran) i ett personnummer.

- Multiplicera siffrorna i alla udda positioner med 2.
- Multiplicera siffrorna i alla jämna positioner med 1.
- Summera alla enskilda siffror i delresultaten.
  Det betyder att om ett delresultat är tvåsiffrigt ska var och en av dessa summeras som om de vore ental (se exemplet).
- Ta resten(modulus) av heltalsdivisionen ( summa/10 ).
  (Dvs. summa%10 på ”C++språket”)

Om du då har ett korrekt personnummer:

1.) Om du tar med kontrollsiffran i beräkningen ska resten bli 0.

2.) Om du inte tar med kontrollsiffran i beräkningen ska
( (10 – resten)%10 ) ge kontrollsiffran.

### Exempel:

```
                8   9   0   8   2   1  -  5   8   7   7

Multipliceras:  *2  *1  *2  *1  *2  *1    *2  *1  *2  *1
Delresultat:   =16  =9  =0  =8  =4  =1   =10  =8 =14  =7

Adderas 1.):    1+6 +9  +0  +8  +4  +1   +1+0 +8 +1+4 +7 = 50
modulus10:     = 0          Kontrollsiffran är rätt.

Adderas 2.):    1+6 +9  +0  +8  +4  +1   +1+0 +8 +1+4    = 43
modulus10:     = 3
Subtraheras:   10-3 = 7
modulus10:     = 7          Kontrollsiffran är 7.
```

Titta speciellt på hur man gör när ett delresultat blir tvåsiffrigt.

### 7c1. Kontroll av personnummer.

Gör ett program som läser in ett personnummer (som teckensträng) från tangentbordet och därefter kontrollerar kontrollsiffran.

7:e tecknet i personnumret skall vara ett bindestreck `'-'` eller ett mellanslag `' '`.

Exempel: ”123456-7890”

Programmet skall skriva ut antingen `Kontrollsiffran rätt` eller `Kontrollsiffran felaktig, rätt siffra är X`.

### Tips:

Efter att du läst in teckensträngen i programmet bör du börja med att ta ut tecknen i den inmatade teckensträngen, omvandla vart och ett till motsvarande heltal, och lagra dem i ett heltalsfält. Det blir då mycket enklare att göra beräkningarna.

### Testning:

Testa både med riktiga och felaktiga personnummer. Kontrollera även ett riktigt personnummer med kontrollsiffra 0. Prova om programmet föreslår rätt kontrollsiffra om du skriver fel kontrollsiffra.

### Redovisning:

Beskriv i rapporten hur du har tänkt när du skrivit ditt program.
Bifoga din källkodsfil tillsammans med rapporten.

### 7c1. Mer kontroller av personnummer. (frivillig uppgift)

I föregående uppgift görs ingen rimlighetskontroll av personnumret innan kontrollsiffran beräknas. T.ex. görs ingen koll på att 7:e tecknet verkligen är ett bindestreck eller mellanslag. Inte heller om det är inskrivet orimliga datum som t.ex. ”761342-1234”.

Utöka föregående program med kontroller av 7:e tecknet och datumkontroll. Du väljer själv hur avancerad datumkontroll du vill göra. Den enklaste kollen är att se att månaden ligger mellan 1 och 12 och dagen mellan 1 och 31. En mer avancerad kontroll kan ta hänsyn till vilken månad det är vid kontroll av dagen. Ännu mer avancerat blir det förstås om man tar hänsyn till skottår!

#### Tips:

Med hjälp av tecknen som bildar årtalet kan du skapa ett heltalsvärde som kan lagras i en variabel. Då kan du arbeta med t.ex. heltalet `76` istället för de två tecknen `'7'` och `'6'`. Samma sak gäller för månader och dagar. Det blir då mycket enklare att göra beräkningarna.

#### Redovisning:

Behöver inte redovisas.

## Laborationsrapport 7.

Förutom rapporten ska du skicka dina källkodsfiler till handledaren. Du ska inte skicka med några andra filer.

I övrigt gäller samma som i föregående laboration.