Datorövningar

Grunderna i C/C++

övning 1

Innehåll: Editering, kompilering, länkning och exekvering av C++-program. Något om funktioner/procedurer.

1. Ett program är ett antal rader text som beskriver lösningen till ett problem. Om problemet är att på datorns skärm skriva ut texten ”Hello, world!”, så ser programmet ut så här (i programspråket C++):

```
#include <iostream.h>
void main()
{
   cout << ”Hello World!”;
}
```

2. När du skall skriva ett C++-program måste du ha en *texteditor*. I princip går det bra att använda vilken enkel editor som helst, exv dtpad, emacs eller vi. Den C++-kompilator som vi använder på kursen är gnu c plus plus (g++). Den innehåller alla verktyg som behövs för att skriva fungerande C++-program.

3. Logga in på ditt användarkonto.

4. Skapa en mappstruktur på din hårddisk för de filer som du skapar under kursen. Skapa exv mappen *c_prog*. I denna kan du sedan efterhand skapa lämpliga undermappar för datorlaborationer etc. Det gör det lättare för dig att hålla ordning på dina filer.

5. Du kan behöva installera gcc och tillhörande dokumentation, den återfinner du på www.sunfreeware.com. Följ instruktionerna i så fall, testa dock först genom att skriva gcc och g++, kommer det upp ett meddelande av typen “no input files” , är gcc redan installerat.

6. Skriv in programmet i 1 ovan. Ta för vana att alltid spara källkoden innan du börjar kompilera. Spara

7. För att kompilera ett enkelt program skriver du g++ <programnamn.cpp> Eventuella fel vid kompileringen visas därefter.

   Vid kompileringen skapas en utfil a.out som innehåller det översatta programmet. Vid kompilering i flera steg kan man notera att en .out fil också finna. Du måste sedan göra denna a.out fil körbar med chmod u+x a.out. Den färdiga a.out-filen kan sedan exekveras med ./a.out. *Notera att ett otal kompilatordirektiv finns.*

8.
```
#include <iostream.h>
void main()
{
   int tal1, tal2;
   cout << “Ange två tal (åtskilda med mellanslag): “;
   cin >> tal1 >> tal2;
   clrscr();
   cout << “Produkten av “ << tal1 << “ och “ << tal2 << “ är: “;
```

```
    cout << (tal1*tal2);
}
```

Skriv in och provkör programmet ovan!

6) 9. När du arbetar med vanlig text på bildskärmen fungerar den som ett rutnät med normalt 25 x 80 tecken. Raderna (y-koordinaten) är då numrerade 1 t o m 25 med den första raden överst. Kolumnerna (x-koordinaten) är numrerade 1 t o m 80 med kolumn 1 längst till vänster.

Modifiera programmet i 8 så att inmatningstexten för det första talet skrivs ut på rad 5 och det andra på rad 6. Resultatet skall sedan skrivas ut på rad 10.

övning 2

Innehåll: Uttryck , satser och program flöden. Slumptal

Skriv ett program först läser in antalet värden och som sedan läser värdena ett och ett, därefter beräknar och skriver ut medelvärdet och standardavvikelsen för värdena. Skärmen skall rensas före inmatning av varje nytt värde.

1. I biblioteket *stdlib* finns funktionerna *int random(int max)* och *void randomize()*. Med funktionen *random* kan du själv bestämma inom vilket intervall slumptalen skall hamna. Parametern *max* anger den övre gränsen vilken inte får användas. *random(5)* ger således slumpmässigt ett av värdena 0,1,2,3 och 4. Funktionerna är inte heller dessa standardiserade men finns i de flesta system.

   Funktionen *randomize()* används för att ge slumptalsgeneratorn ett slumpmässigt startvärde ("seed"), annars erhålles samma slumptalsserie varje gång och det är ju inte meningen. Innan du i programmet börjar generera slumptal, skall du således anropa *randomize()*.

   Fˆljande program ("Gissa tal") fˆrekom under Java-kursen:

```
import dna.util.SimpleRandom;
import dna.util.SimpleInput;

class GissaTal {
      public static void main(String[] args) {
             int nbrComp, nbrMe, nbrGuess;
             nbrGuess = 0;
             nbrMe = 0;
             nbrComp = SimpleRandom.randInt(1,1000);
             System.out.println("VILKET TAL TÄNKER JAG PÅ?");
             while (nbrComp != nbrMe) {
                    nbrMe = SimpleInput.readInt();
             nbrGuess += 1;
             if (nbrMe > nbrComp)
                    System.out.println("FEL! FÖR HÖGT!");
                    if (nbrMe < nbrComp)
                           System.out.println("FEL! FÖR LÅGT!");
             }
             System.out.println("RÄTT! DU BEHÖVDE "+nbrGuess+"
GISSNINGAR!");
      }
}
```

   Skriv om programmet i C++ ! "Hyfsa" dessutom till utskriften lite så att du får en programexekvering

liknande den nedan:

2. Skriv ett program som bestämmer största respektive minsta värde bland ett godtyckligt antal inmatade tal. Antalet tal som skall matas in skall anges av användaren först då programmet körs.
3. Skriv ett program som skriver ut siffrorna 0-9 (slumpmässigt) i slumpmässigt genererade punkter (x,y), där x=1,2,….80 och y=1,2,….25, på skärmen tills användaren trycker på en tangent. (Tips: lägg in en fördröjning med en *for*-loop någonstans i programmet annars går det lite väl fort!)

övning 3

1. Skriv ett program med en main-funktion och en funktion *int dice()*, vilken ger antalet prickar vid kast med en tärning. Simulera 100 000 kast med tärningen och beräkna sedan sannolikheten för att få sex prickar. Vad bör svaret bli?

2. Skriv ett program med en main-funktion och tre andra funktioner med följande namn och parametrar:

```
float box(float langd, float bredd, float hojd)
float kon(float diameter, float hojd)
2  float cylinder(float diameter, float hojd)
```

I var och en av funktionerna skall begränsningsytan för "produkten" i fråga beräknas. I main-funktionen skall användaren, genom att mata in exv 1=Box, 2=Kon eller 3=Cylinder, välja någon av produkterna och tillsammans med ett inmatat materialpris (kr/m2) få besked om materialkostnaden för att tillverka produkten i fråga.

Som framgår av utskriften ovan skall programmet fråga användaren efter en beräkning, om ny beräkning skall utföras. I biblioteket *conio* finns exempelvis funktionen *getch()*, som tar emot ett tecken utan att eka det tillbaka på skärmen. Vilken tangent/tecken som trycktes ner kan testas exempelvis i en *while* eller *if*-sats.

Ex.

```
char tkn;
tkn=getch();
if (tkn=='J' || tkn=='j')
   …..
```

3. Skriv en main-funktion som låter dig ställa klockan och sedan med hjälp av klassen *Clock* visar en digital
klocka som kontinuerligt visar tiden.

övning 5

Innehåll: Mera om klasser. Vektorer

1. Vi har tidigare i ett antal exempel simulerat olika företeelser med hjälp av slumptal. Metodiken har då varit s k "dragning med återläggning", dvs ett och samma slumptal har kunnat dras flera gånger, exv vid simulering av kast med en tärning. Men hur skall man förfara om ett slumptal bara får komma upp en gång? Ett exempel på detta utgör dragningen i det svenska Lotto-spelet, där ju sju nummer plus två tilläggsnummer mellan 1 och 35 skall dras varje vecka. Om exv talet 12 dras, så skall ju detta tal inte kunna komma upp som något av nästföljande tal osv.

   Hur skall man lösa detta om man skall låta datorn generera slumptalen mellan 1 och 35? Ett sätt är att använda vektorer och låta programmet hålla reda på vilka nummer som gått. Använd exempelvis en heltalsvektor, *kontroll* med plats för 35 element (0-34) som samtliga sättes lika med *0* innan själva dragningen startar. För varje nytt nummer som datorn sedan slumpar fram, skall elementet med motsvarande plats i vektorn som det dragna numret kontrolleras. Om elementet har värdet *0* har numret inte tidigare gått och kan användas, varvid programmet också måste sätta elementets värde lika med *1* för att markera att det använts. Är elementet däremot redan lika med *1*, har numret redan gått och ett nytt värde måste slumpas fram.

   Ex. Datorn "drar" talet 23. Programmet skall då kontrollera värdet i elementet *kontroll[22]*. Om elementets värde är *1*, måste ett nytt nummer dras. Är det däremot lika med *0*, har siffran 23 inte tidigare gått och kan därför användas. Elementet *kontroll[22]* sätts samtidigt lika med *1*, för att markera att numret nu är använt.

2. Följande main-funktion visar metodiken:

```
#include <iostream.h>
#include <stdlib.h>
#include <conio.h>
void main()
{
    int dragetNr;
    int kontroll[35];
    for (int k=0; k<=34; k++)
          kontroll[k]=0;
    randomize();
    cout << "LOTTODRAGNING\n";
    cout << "=============\n";
    for (k=1; k<=9; k++)
    {
           do
           {
                 dragetNr = random(35)+1;
           } while (kontroll[dragetNr-1]==1);
```

```
                kontroll[dragetNr-1] = 1;
                if (k<=7)
                     cout << "Nummer         " << k << ": " << dragetNr << endl;
                else
                     cout << "Tilläggsnummer " << (k-7) << ": " << dragetNr <<
endl;

                // fördröjning i programmet för att dragningen inte skall gå
                // så snabbt
                for (long i=1;i<=20000000;i++);
        }
    }
```

Programmet som heter *lotto* finns färdigskrivet. Hämta in och provkör!

3. En klass som beskriver en bingobricka av ”modell” Bingolotto har följande deklaration:

```
class BingoBricka
{
   public:
                /*  skapa en BingoLotto-bricka med övre vänstra hörnet i
                punkten ixStart, iyStart med 15x5 rader som ritas i fönstret
iw
                och som i första kolumnen har siffrorna 1-15 slumpmässigt
                fördelade, i andra kolumnen siffrorna 16-30 osv. */
                BingoBricka(int ixStart, int iyStart);  // konstruktor
                ~BingoBricka();                         // destruktor

                /* rita upp den fyllda brickan i ritfönstret */
                void ritaBricka();

                /*  markera draget nummer på brickan i ritfönstret genom att
                ringa in siffran */
                void markNbr(int dragetNr);

                /* gå igenom raderna på brickan och kontrollera om bingo
                erhållits. 1=bingo, 0=ej bingo */
                int bingo();

                /* tag reda på radnumret vid Bingo */
                int getRowNbr();
   private:
                int bricka[15][5];               // en bricka
                int check[75];                  // kontrollvektor med 75
element
```

```
                int nbr, nbrsTrue, radNr, xStart, yStart,isBingo;
                char str[10];
};
```

Klassen finns färdigskriven under namnet *bingobr*.

4. Skriv ett program *bingo* med en main-funktion som låter datorn genomföra en omgång Bingolotto med *tre* brickor.

   Själva dragningen går till så att datorn får generera heltalsslumptal i intervallet 1 till 75 tills någon av brickorna får bingo, dvs har fem dragna nummer på samma rad. Då bingo har erhållits skall texten längst upp på skärmen nedan skrivas ut, dvs vilken bricka bingo har erhållits på samt vilken rad, jämte totalt antal dragna nummer.

övning 6

Innehåll: Enkel filhantering. Stränghantering och tecken-vektorer.

1. På en fabrik som tillverkar elektronikkomponenter tillverkas bl a kondensatorer. Just nu tillverkar man en stor batch med kondensatorer märkta med kapacitansen 2.5 pikoFarad. För att kontrollera produktkvalitén tar man varje dag slumpmässigt ut ett antal kondensatorer som kontrollmäts med avseende på kapacitansen. Driftingenjören matar in de uppmätta värdena i en binär datafil märkt med dagens datum.

   Som produktchef vill du naturligtvis datorisera bearbetningen av materialet och bestämmer dig för att skriva ett C++-program som hjälper dig utvärdera materialet. Programmet skall göra följande:

   - läsa in samtliga uppmätta värden lagrade i “dagens” fil. (Testfil: 990328)
   - beräkna medelvärde, standardavvikelse, största och minsta värde i stickprovet
   - skriva ut stickprovets storlek, dvs antal inlästa värden

2. En sträng är en följd av tecken som deklareras som ett fält av typen *char*, ex:

   ```
   char nameArray[50];
   ```

   De enskilda tecknen i strängen kan du få fram på vanligt sätt då det gäller vektorer genom att ange *index* för det/de element (tecken) du är intresserad av, exv *nameArray[0]* för det första tecknet, *nameArray[3]* för det fjärde osv. Prova att skriva in följande program:

   ```
   #include <iostream.h>
   void main()
   {
      char nameArray[50];
      cout << “Ange ditt namn: “;
      cin >> nameArray;
   ```

```
   cout << “Jaså, du heter “ << nameArray << endl;
   cout << nameArray[1];
}
```

Om du matar in exv namnet *Larsson* kommer du att få utskriften:

```
Jaså, du heter Larsson
a
```

3. Kör ovanstående program och undersök vad som händer för inmatningar som inleds med mellanslag och som innehåller mellanslag.

4. Som du ser gäller vid inläsning av strängar med *cin* att:

   - inledande mellanslag hoppas över
   - läsningen sker fram till nästa blanktecken

   Detta innebär att du inte kan mata in ett helt namn som innehåller mellanslag eller löpande text.
   Om du ändrar programmet i 2 ovan och byter ut raden med *cin* mot:

   ```
   cin.getline(nameArray,50);
   ```

   kommer du att kunna läsa in en sträng, i det här fallet med mindre än 50 tecken som *får* innehålla mellanslag. Prova detta!

5. För att få fram längden på en sträng , dvs det faktiska antalet tecken i strängen just nu, kan du använda funktionen *strlen()* i biblioteket *string*.

   ```
   lengd=strlen(nameArray);
   ```

   ger variabeln *lengd* värdet på det antal tecken som finns i strängen *nameArray*. Komplettera programmet ovan med en utskrift även av längden på strängen som du matar in. (Glöm inte att inkludera <string.h> när du använder funktionen.)

6. I datafilen *secret* finns en text lagrad på max 500 tecken som dessutom är kodad baserad på följande kodnyckel:

   ```
   Alfabetet:       ABCDEFGHIJKLMNOPQRSTUVwXYZÅÄÖ.,?!

   Kodnyckel:       FXIPVGOÅZWKRAECHLÄYUBDÖJMQSTN.,?!
   ```

   dvs ett “kodat” P motsvarar i verkligheten bokstaven D, ett C motsvarar bokstaven O osv.

   Skriv ett program med en main-funktion, som läser in den kodade texten i filen *secret*, dechiffrerar den och skriver ut den i klartext på skärmen. Läs in den kodade texten exv med hjälp av följande

instruktionssekvens till strängen *theCode[500]*:

```
.
.
ifstream fin("secret", ios::binary);
fin.read((char *)&theCode, sizeof(theCode));
fin.close();
.
.
```