Innehållsförteckning

7 Arrayer

7.1 Inledning

En array är en indexerad variabel som används för att enklare hantera data när data består av mer än ett värde av samma datatyp. Det kan t.ex. vara temperaturavläsningar, ett värde för varje dag under ett år. Array, vilket är den engelska beteckningen, heter vektor på svenska. I några läroböcker förekommer också begreppet fält som synonym för array.

I den nya standarden, C++ 2011, har man infört en ny standardklass som heter array och som förenklar hanteringen av arrayer. Jag betecknar den i fortsättningen std::array.

Sedan tidigare finns standardklassen vector som också hanterar arrayer på ett enkelt sätt.

Här i avsnitt 7 går jag igenom hur man använder traditionella arrayer, std::array och std::vector för hantera stora mängder av data, söka data och sortera data.

7.2 Arraybegreppet

Antag att man har tre variabler x_1, x_2 x_3
Dessa kan C++ hantera som en enhet:

```
int x[3];
```

Detta är en array, en lista av heltal. Siffran inom hakarna [] anger **antalet** element, **x** är arrayens namn och **int** arrayens datatyp .

Arrayen består av tre element x[0], x[1], x[2]. Man anger element med index. Första index = 0;
Varje element lagrar (i detta fall) ett heltal.

Följande bild visar arrayen x:

Index	0	1	2
Data	34	-45	789

Obanstående innehåll får arrayen om man gör tilldelningarna:

```
x[0] = 34;
x[1] = -45;
x[2] = 789;
```

Ett exempel på hur man kan använd en array. Summera talen i arrayen x:

```
int sum =0;
for(int i =0; i<3; i++)
  sum += x[i];
```

I den nya standarden finns detta alternativa skrivsätt:

```
int sum = 0;
for(auto i: x)
  sum += i;
```

Här stegas alla element arrayen x igenom. Systemet håller reda på hur stor x är. Indexet i läser värdet från det element som det pekar på för tillfället!

7.3 Arrayer

I avsnitt 7.3 beskrivs hur man använder traditionella arrayer.

7.3.1 Deklaration av arrayer

Ex

int y[100];	Skapa plats för 100 int
float skruvDiam[30];	skapa plats för 30 flyttal
const size_t SIZE = 40;	Dataypen *size_t* bör användas för index i en array
char name[SIZE];	Skapa plats för 40 char
size_t size = 25;	
int tal[size]	Är **inte** korrekt, eftersom arrayens storlek måste vara konstant. Det skapas en fast plats i minnet för arrayen

7.3.2 Använd en array för att lagra 5 heltal

I följande exempel visas hur man tilldelar heltal till en array och läser arrayens värden.

```
// array_010
// Ett första försök med arrayer
// Per Ekeroot 2013-09-03
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>   // cout
#include <iomanip>    // setw()
using namespace std;
//--------------------------------------------------------------------------
int main()
{
  // Deklarera en array
  const size_t SIZE = 15;
  int x[SIZE] = {0};

  // Fyll arrayen
  for(size_t idx=0; idx < SIZE;idx++)
    x[idx] = 3*idx - 2 ;

  // Skriv ut arrayens värden
  cout << " x= ";
  for( size_t idx=0; idx < SIZE; idx++)
    cout << setw(4) << x[idx] ;

  // Skriv ut arrayens adress (= plats i minnet)
  cout << endl <<endl;
  cout << "Adressen till x= " << x << endl << endl;
  return 0;
}
```

Kommentarer:

- Deklarera en konstant (SIZE), så blir det enklare att ändra från 5 till t ex 120 element i arrayen.
- Tilldelning av värden till arrayen måste ske elementvis.
- Utskrift av arrayen måste ske elementvis.
- cout << x; skriver arrayens adress.

- Ett allternativ som tillkommit i den nya standaren är att man för en array kan använda den föreklade for-loopen:

```
for(auto idx  : x)
 cout << setw(4) << idx;
```

Tolka den så här: för varje element i arrayen `x`, från första till sista, kopiera värdet från `x` till `idx` och skriv ut `idx`. Här behöver man alltså inte själv hålla reda på arrayens storlek

7.3.3 Kopiering av en array

Om vi har en array som är fylld med data och vill skapa ytterligare en array med samma innehålla kan kopiera från den första till den andra arrayen element för element så här:

```
for(size_t i=0; i < SIZE; i++)
    y[i] = x[i];
```

Hela programmet:

```
// array_020
// Kopiera array
// Per Ekeroot 2013-09-03
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>   // cout
#include <iomanip>    // setw()
using namespace std;
//-----------------------------------------------------------------------------
int main()
{
  // Deklarera en array
  const size_t SIZE = 15;
  int numbers[SIZE] = {};

  // Fyll arrayen
  for(size_t idx=0; idx < SIZE;idx++)
    numbers[idx] = 3*idx - 2 ;

  // Skriv ut arrayens värden
  cout << " Numbers= ";
  for(auto idx: numbers)
    cout << setw(4) << idx ;

  // Deklarera ytterligare en array
  int numbers2[SIZE] = {};

  // Kopiera numbers till numbers2
  for(size_t idx=0; idx < SIZE; idx++)
    numbers2[idx] = numbers[idx];

  // Skriv ut arrayen numbers2
  cout << endl <<endl;
  cout << " numbers2= ";
  for(auto idx: numbers2)
    cout << setw(4) << idx ;

  return 0;
}
```

Kommentarer:

- Man **måste** kopiera arrayer element för element
- Det går alltså **inte** att skriva y = x !!

7.3.4 Alternativa sätt att deklarera och initiera en array

Andra sätt att deklarera och tilldela värden till arrayer.

- `int x[5] = {2,4,6,8,10};` Initiering av värden vid deklaration
- `int x[] = {2,4,6,8,10,12}` Kompilatorn räknar antalet själv
- `double y[10] = {0}` nollställer **alla** element
- `double y[5] = {3.5, -4.2}` Sätter resp. värde på de två första elementen och 0 på resten.
- enbart int [20]; ger odefinierade värden i alla element.

Så här får man **inte** göra:
Deklarera först två arrayer:

```
int x1[4] = {20,30,40,50};
int x2[4];
x2 = x1;              // FEL!!
x2 = {2,3,4,5};       // FEL!!
```

Kommentarer:

- Arrayer måste tilldelas (kopieras) elementvis. Kommentar för x2 = x1;
- Initiering kan enbart ske vid deklaration. Kommentar för `x2 = {2,3,4,5};`

7.3.5 Använd en array för att beräkna summa och medelvärde för ett stort antal tal

Här följer ett exempel som använder en array för att beräkna summa och medelvärde för ett stort antal tal.
Programstruktur:

- Skapa en array med 100 element
- Låt användaren ange hur många tal som ska slumpas
- Slumpa talen i intervallet 1 – 1000 till en array
- Skriv ut talen med 10 tal på varje rad
- Beräkna talens summa och medelvärde
- Skriv ut talens summa och medelvärde

```
// array_030
// Använd en array för summa- och medelvärdesberäkning
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>        // cout
#include <iomanip>         // setw()
#include <ctime>           // time()
#include <random>
using namespace std;

//--------------------------------------------------------------------------
int main()
{
  // Deklarera och initiera konstanter och variabler
  const size_t SIZE = 100;
  int numbers[SIZE] = {};
  size_t num;
```

```
  // Initiera slumpning
  default_random_engine generator(static_cast< unsigned>(time(0)));
  uniform_int_distribution<int> random(1,1000);

  // Rubrik, ledtext och inmatning
  cout << "Random integers in intervall [1,1000], calculate sum and average";
  cout << " How many integers do you want to randomize (<="<< SIZE << "): " ;
  cin >> num;

  // Slumpa talen och lägg in dem i vektorn
  for(size_t idx=0; idx < num; idx++)
    numbers[idx] = random(generator);

  // Skriv talen i 10 kolumner
  for(size_t idx=0; idx < num; idx++)
  {
     cout << setw(5) << numbers[idx];
     if((idx+1)%10 == 0)            // Ny rad för var 10:e element
       cout << endl;
  }

  //Beräkna summa och medelvärde
  int sum = 0;
  float average = 0;
  for(size_t idx=0; idx < num; idx++)
    sum += numbers[idx];

  average = static_cast<float>(sum) / num;

  // Skriv summa och medelvärde
  cout << endl << endl << " Sum     = " << sum << endl;
  cout << " Average = " << average << endl << endl;

  return 0;
}
```

Kommentarer:

- OBS! Det finns ingen kontroll på att användaren anger antalet tal som ska slumpas mindre eller lika med 100 (<=100). Vad händer om man anger ett antal större än 100?
- Med for-loopen tilldelar man värden elementvis till arrayen
- Tvinga typomvandling (`average = static_cast<float>(sum)/ num;`). Eftersom sum och num är heltal så utförs heltalsdivision om typomvandling inte görs.

7.3.6 Sökning i en array

Om man vill veta om ett tal förekommer i en array kan man söka efter talet på följande sätt:

Antag att arrayen är deklarerad så här:

```
const size_t SIZE = 10;
int array[SIZE];
```

Använd en for-loop för att stega igenom arrayen och en if-sats för att kolla om ett element innehåller talet 7:

```
  int searchNumber = 7;
  bool found = false;
  for(auto idx : arr)
  if( idx == searchNumber)
  {
    found = true;
    break;
  }
```

Om det sökta talet finns i arrayen sätts den boolska variabeln `found` till true och sökningen avbryts med `break`. Om sökningen misslyckas har `found` värdet false.

Detta sätt att söka kallas för linjär sökning eftersom man börjar sökningen i första elementet och sedan genomsöker arrayen element för element (linjärt!). Sortering visas i nästa exempel.

```
// array_040
// Sök i en array
// Returnera index för elementet där det sökta värdet ligger
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>
#include <string>
#include <ctime>          // time()
#include <random>     // slumpning
using namespace std;

int main()
{
//--------------------------------------------------------------------------
// Deklarera en konstant och en array
//--------------------------------------------------------------------------
  const size_t SIZE = 10;
  int arr[SIZE];

  default_random_engine generator(static_cast<unsigned>(time(0)));   //
Initierar slumpningen
  uniform_int_distribution<int> random(1,10); // Slumpa tal i itervallet [1,10]

//--------------------------------------------------------------------------
// Slumpa tio tal till arrayen
//--------------------------------------------------------------------------
  for(auto &idx : arr)
    idx = random(generator);
```

```
//-------------------------------------------------------------------------------
// Skriv arrayen på skärmen
//-------------------------------------------------------------------------------
  for(auto idx : arr)
    cout << idx << "  ";

//-------------------------------------------------------------------------------
// Sök efter talet 7 i arrayen
//-------------------------------------------------------------------------------
  int searchNumber = 7;
  bool found = false;
  for(auto idx : arr)
  if( idx == searchNumber)
  {
    found = true;
    break;
  }

//-------------------------------------------------------------------------------
// Skriv resultatet av sökningen
//-------------------------------------------------------------------------------
  cout << endl << endl;
  if(found)
    cout << "The number is found!";
  else
    cout << "The number is not in the array!";
           cout << endl << endl;
  return 0;
}
```

Kommentarer:

- Talen slumpas i intervallet 1 – 10.
- Den förenklade for-satsen används här för att lägga in värden i arrayen

```
    for(auto &idx : arr)
      idx = random(generator);
```

Observera att variabeln `idx` måste referensdeklareras! Om man inte referensdeklarerar `idx` så **kopieras** elementen i `arr` till `idx` och det går inte att tilldela värden till elementen. Med referensdeklarationen **refererar** `idx` till elementen i `arr` och en tilldelning är möjlig.

7.3.7 Sortering av en array

Om man vill ordna talen i arrayen i stigande eller fallande ordning kan man göra på många olika sätt. Det finns enkla, ineffektiva sorteringsalgoritmer och det finns komplicerade, effektiva sorteringsalgoritmer. Om datamängderna inte är alltför stora (mindre är 10 000 heltal) duger de enkla ineffektiva sorteringsalgoritmerna. Några sorteringsalgoritmer:

- Bubblesort
- Selectionsort (urvalssortering)
- InsertionSort
- QuickSort

Vi väljer att visa Bubblesort. Algoritmen fungerar så här:
[6][2][1][3][4][5][8][7][0] Startarray

[2][6][1][3][4][5][8][7][0] 2 och 6 byter plats
[2][1][6][3][4][5][8][7][0] 1 och 6 byter plats
[2][1][3][6][4][5][8][7][0] 3 och 6 byter plats
[2][1][3][4][6][5][8][7][0] 4 och 6 byter plats
[2][1][3][4][5][6][8][7][0] 5 och 6 byter plats
[2][1][3][4][5][6][7][8][0] 7 och 8 byter plats
[2][1][3][4][5][6][7][0][8] 0 och 8 byter plats
[1][2][3][4][5][6][7][0][8] 1 och 2 byter plats
[1][2][3][4][5][6][0][7][8] 0 och 7 byter plats
[1][2][3][4][5][0][6][7][8] 0 och 6 byter plats
[1][2][3][4][0][5][6][7][8] 0 och 5 byter plats
[1][2][3][0][4][5][6][7][8] 0 och 4 byter plats
[1][2][0][3][4][5][6][7][8] 0 och 3 byter plats
[1][0][2][3][4][5][6][7][8] 0 och 2 byter plats
[0][1][2][3][4][5][6][7][8] 0 och 1 byter plats

Jämför elementen parvis. Börja längst till vänster. Byt plats om det till höger är mindre. Fortsätt med nästa par tills listan är slut. För att vara säker på att listan är sorterad måste man gå igenom den SIZE-1 gånger.

I C++ kan man skriva denna algoritm på följande sätt (det finns alternativ!):

```
for( int pass=0; pass < SIZE-1; pass++)  //Stega igenom arrayen SIZE-1 antal ggr
  for(int i=0; i < SIZE-1; i++)          // Jämför elementen parvis
    if(array[i] > array[i+1])          // Byt plats om fel ordning
    {
      tmp = array[i];
      array[i] = array[i+1];
      array[i+1] = tmp;
    }
```

Kommentarer:

- pass håller reda på hur många gånger arrayen ska genomlöpas
- `if(array[i] > array[i+1])` betyder: "om värdet som står till vänster är större än värdet som står till höger"
- När man vill byta plats på två värden krävs det att man har en temporär lagringsplats för ett värde. Jämför funktionen swap() i avsnittet om funktioner.

Denna sorteringsalgoritm ingår i nästa exempel:

```
// array_050
// Programmet demonstrerar sortering med bubble sort
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>
#include <iomanip>
#include <ctime>
#include <random>
using namespace std;
//-----------------------------------------------------------------------------
int main()
{
//-----------------------------------------------------------------------------
// Deklarera konstant och array, samt initiera slumpningen
//-----------------------------------------------------------------------------
  const size_t SIZE = 150;
  int arr[SIZE] ={0};

  default_random_engine generator(static_cast<unsigned>(time(0)));
  uniform_int_distribution<int> random(0,999);    //Slumpa tal i [0,999]

//-----------------------------------------------------------------------------
// Slumpa värden till arrayen  (0 - 999)
//-----------------------------------------------------------------------------
  for(auto &idx : arr)
    idx = random(generator);

//-----------------------------------------------------------------------------
// Skriv osorterad array
//-----------------------------------------------------------------------------
  cout << "Osorterad array" << endl;
  int num=1;
  for(auto idx: arr)
  {
    cout << setw(4) << idx;
    if(num % 15 == 0)                   // Skriv 15 tal på varje rad
      cout << endl;
    num++;
  }

//-----------------------------------------------------------------------------
// Sortera arrayen
// Metod: Bubble sort
//-----------------------------------------------------------------------------
  for( int pass=0; pass < SIZE-1; pass++)//Stega igenom arrayen SIZE-1 antal ggr
    for(int i=0; i < SIZE-1; i++)        // Jämför elementen parvis
      if(arr[i] > arr[i+1])              // Byt plats om fel ordning
      {
        int tmp = arr[i];
        arr[i] = arr[i+1];
        arr[i+1] = tmp;
      }
```

```
//--------------------------------------------------------------------------
// Skriv sorterad array
//--------------------------------------------------------------------------
  cout << endl << endl << "Sorterad array" << endl;
  num=1;
  for(auto idx: arr)
  {
    cout << setw(4) << idx;
    if(num % 15 == 0)                    // Skriv 15 tal på varje rad
      cout << endl;
                      num++;
  }

  return 0;
}
```

Kommentarer:

- Sortering görs i stigande ordning
- Testa med SIZE =1500, 15000

7.3.8 Leta rätt på största och minsta värde i en array

Om man vill leta rätt på största och minsta värdet i arrayen `integers[]` deklarerar man två variabler max och min för att lagra största respektive minsta värde. Sedan stegar man genom arrayen och sparar undan största respektive minsta värde:

```
  int max = integers[0];  // max initieras till det första värdet i integers
  int min = integers[0];  // min initieras till det första värdet i integers

  for(auto idx: integers)
  {
    if(idx > max)  // Leta det största värdet
      max = idx;
    if(idx < min)  // Leta det minsta värdet
      min = idx;
  }
```

Hela programmet:

```
// array_060
// Sök efter max- och minvärde i en array
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>
#include <iomanip>
#include <ctime>
#include <random>
using namespace std;
//--------------------------------------------------------------------------
int main()
{
//--------------------------------------------------------------------------
// Deklarera konstant och array samt initiera slumpningen.
//--------------------------------------------------------------------------
  const size_t SIZE = 150;
  int integers[SIZE];
  default_random_engine generator(static_cast<unsigned>(time(0)));
  uniform_int_distribution<int> random(1,1000);

//--------------------------------------------------------------------------
```

```
// Slumpa värden till arrayen i intervallet 1 - 1000
//-----------------------------------------------------------------------------
   for(auto &idx : integers)
    idx = random(generator);

//-----------------------------------------------------------------------------
// Skriv ut arrayen
//-----------------------------------------------------------------------------
  cout << "Array integers:" << endl;
  int num=1;
  for(auto idx: integers)
  {
    cout << setw(4) << idx;
    if(num % 15 == 0)                   // Skriv 15 tal på varje rad
      cout << endl;
    num++;
  }

//-----------------------------------------------------------------------------
// Leta rätt på största respektive minsta värde i arrayen
//-----------------------------------------------------------------------------
  cout << endl << endl;
  int max = integers[0]; // max initieras till det första värdet i integers
  int min = integers[0]; // min initieras till det första värdet i integers

  for(auto idx: integers)
  {
    if(idx > max)  // Leta det största värdet
      max = idx;
    if(idx < min)  // Leta det minsta värdet
      min = idx;
  }
  cout << "Maximum element in the array integers is: " << max << endl;
  cout << "Minimum element in the array integers is: " << min << endl << endl;

  return 0;
}
```

7.3.9 2D-array

Vi har hittills sett exempel på endimensionella arrayer. Det går även att ha arrayer av högre dimension. För en tvådimensionell array kan man tänka sig att varje element istället för ett heltal innehåller en ny array med heltal.

1	2	3	4	5	6	7	8	9	10	11	12

Omritat blir detta:

1	2	3	4
5	6	7	8
9	10	11	12

Fysiska exempel på tvådimensionella arrayer:

- schackbräde
- knappsatsen på en telefon eller ett tangentbord
- matriser (= ett matematiskt begrepp för siffror i ett rektangulärt mönster)

I C++ hanterar man 2D – arrayer på följande sätt. Antag att vi ska lagra en matris i en 2D-array och att matrisen ser ut så här:

Rad \ kol	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

Definition av en 2D-array med 3 rader och 4 kolumner:

```
int A[3][4];  // int A[radAntal][kolumnAntal]
```

Detta ska ses som en array med 3 element där varje element innehåller 4 element, d.v.s. 3 rader med 4 kolumner i varje rad. I varje element ligger ett heltal (int).

r0				r1				r2			
k0	k1	k2	k3	k0	k1	k2	k3	k0	k1	k2	k3

I minnet ligger talen i en rad, som i fig. ovan från vänster till höger. Initiering av värden till 2D-arrayen görs p.g.a. detta så här:

```
int A[3][4] ={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

Ett exempel följer här:

```
// array_070
// 2D-vektor: hantera matriser
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>
#include <iomanip>   // setw
using namespace std;
//--------------------------------------------------------------------------
int main()
{
//--------------------------------------------------------------------------
// Deklarera och initiera matrisen A
//--------------------------------------------------------------------------
   int A[3][4] ={{1,2,3,4},
                 {5,6,7,8},
                 {9,10,11,12}};
//--------------------------------------------------------------------------
// Skriv ut matrisen A
//--------------------------------------------------------------------------
  for(int row=0; row<3; row++)
  {
    for( int col=0; col<4; col++)
      cout << setw(3) << A[row][col];

    cout << endl;
  }
  return 0;
}
//--------------------------------------------------------------------------
/* Inmatning från tangentbordet
  for(int row=0; row<3; row++)
    for( int col=0; col<4; col++)
    {
      cout << "Input row " << row +1  << " and column " << col +1 << ": " ;
      cin >> A[row][col];
    }
*/
```

Kommentarer:

- Utskrift av 2D-arrayen (matrisen) görs enklast med två nästlade for-lopar:

```
for(int row=0; row<3; row++)
{
   for( int col=0; col<4; col++)
   cout << setw(3) << A[row][col];

    cout << endl;
}
```

  OBS! for-loopen för kolumn (col) ligger innerst och snurrar därmed snabbast. Detta innebär att 2D-arrayen skrivs ut radvis.

- Inmatning kan också göras från tangentbordet:

```
for(int row=0; row<3; row++)
  for( int col=0; col<4; col++)
  {
    cout << "Input row " << row +1 << " and column " << col +1 << ": " ;
    cin >> A[row][col];
  }
```

- Det kommer ytterligare ett 2D-arrayexempel tillsammans med strängar (string_080).

7.3.10 Array med strängar

Deklarera en array med strängar: string name[8];

Deklarera och initiera en array med strängar:

```
  const int SIZE = 5;
  string name[SIZE] = {"Ulla Andersson", "Peter Lundahl","Karin Halvarsson",
                       "Sven Karlsson", "Eva Tengdahl"};
```

Gör en sökning i arrayen

```
     found = false;
     for(int i=0; i<SIZE; i++)
       if( searchName == name[i])
         found = true;
```

Hela programmet:

```
// string_070
// En array med strängar. Sök i arrayen med strängar
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>   // cout, cin
#include <string>     // string-klssen
using namespace std;
//--------------------------------------------------------------------------
int main()
{
//--------------------------------------------------------------------------
// Deklarera konstant
// Deklarera och initiera en array av strängar
// Deklarera variabler till huvudloopen
//--------------------------------------------------------------------------
  const size_t SIZE = 5;
  string names[SIZE] = {"Ulla Andersson",
                        "Peter Lundahl",
                        "Karin Halvarsson",
                        "Sven Karlsson",
                        "Eva Tengdahl"};
   bool   found ;
   string searchName;
   char   ch;
//--------------------------------------------------------------------------
// Programmets huvudloop
//--------------------------------------------------------------------------
   do
   {
     // Mata in söknamn
     cout << endl <<  "Search for: ";
     getline(cin,searchName);

     // Leta i arrayen names efter söknamnet
     found = false;
     for(auto idx : names)
       if( searchName == idx)
         found = true;
```

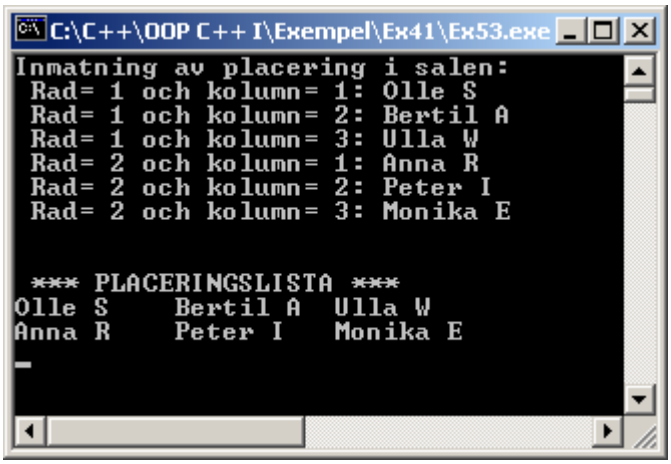
```
// Skriv sökresultatet
     cout << endl << searchName ;
     if(found)
       cout << " is in the array" << endl << endl;
     else
       cout << " is missing" << endl <<endl;

     // Ska sökningen upprepas
     cout << "New search? (y/n): ";
     cin >> ch;
     cin.get();           // Läs bort ENTER från inströmmen
     ch = toupper(ch);
   }while(ch == 'Y');

  return 0;
}
```

Kommentarer:

- Skapa och initiera en string-array med satsen
  string name[antal_element]= {"…","…", …};
- Mata in söknamn med getline(cin,serchName) pga. mellanslaget.
- En for-loop stegar igenom namn-arrayen
- Om searchName == name[i] sätts den boolska variabeln found = true.
- found används för att välja sträng för utskriften.
- OBS! Sökningen är casesensitive. Skriv en egen funktion för att konvertera en hel sträng till versaler, (för säkrare sökning, sortering) . upcase(char ch) konverterar ett tecken till versal (A-Z).

7.3.11 2D-array med strängar

Mata in namn från tangentbordet enligt bild nedan. Personerna placeras sedan i en sal med 2 rader och 3 kolumner.



Deklaration av 2D – array med string i elementen

```
  const int ROWSIZE=2, COLSIZE=3;
  string name[ROWSIZE][COLSIZE];
```

Inmatningen sker sedan i en dubbelloop

```
cout << "Input names for each place in the room: " << endl;
  for(int row=0; row<ROWSIZE; row++)
    for(int col=0; col<COLSIZE; col++)
    {
      cout <<  " Row= " << row +1
           <<  " and column= " << col+1 << ": ";
      getline(cin,name[row][col]);
    }
```

Observera att getline() används för inmatning av namn.

Utskriften sker också med en dubbelloop:

```
for(int row=0; row<ROWSIZE; row++)
  {
    for(int col=0; col<COLSIZE; col++)
      cout << left << setw(10) << name[row][col];
    cout << endl;
}
```

Hela programmet:

```
// string_080
// 2D-vektor och strängar. Namntabell
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>
#include <string>
#include <iomanip>   // setw
using namespace std;
//-----------------------------------------------------------------------------
int main()
{
  const int ROWSIZE=2, COLSIZE=3;
  string name[ROWSIZE][COLSIZE];

//-----------------------------------------------------------------------------
// Mata in namn i tabellen
//-----------------------------------------------------------------------------
  cout << "Input names for each place in the room: " << endl;
  for(int row=0; row<ROWSIZE; row++)
    for(int col=0; col<COLSIZE; col++)
    {
      cout <<  " Row= " << row +1
           <<  " and column= " << col+1 << ": ";
      getline(cin,name[row][col]);

    }
//-----------------------------------------------------------------------------
// Utskrift av placeringstabellen
//-----------------------------------------------------------------------------
  cout << endl << endl << " *** How the persons are placed in the room ***" <<
endl;
  for(int row=0; row<ROWSIZE; row++)
  {
    for(int col=0; col<COLSIZE; col++)
      cout << left << setw(10) << name[row][col];
    cout << endl;
  }

  return 0;
}
```

7.3.12 Funktioner med en array som parameter

Vi har tidigare i kursen sett hur man kan ha funktioner med parametrar av datatyperna, int, float, string. Hur gör man för att skicka med en array som parameter (argument) till en funktion? Här kommer ett exempel i vilket man skapar en funktion som multiplicerar varje element i en array med ett angivet tal och ”returnerar” den manipulerade arrayen.

10	20	30	40	50

multiplicera med 10 ger

100	200	300	400	500

Funktionsdefinition:

```
void multArray( int a[],size_t num)
{
  for(size_t i=0; i<num; i++)
    a[i] *= 10;         // Multiplicera varje element med faktorn 10;
}
```

Det nya: void multArray(int **a[]** , int num) vilket innebär att man skickar med adressen till arrayen, d.v.s. man jobbar på ”originalet”. Detta kan jämföras med **referensanrop**, men är ett **pekaranrop**!

Anrop:

```
int array[SIZE] = {10, 20, 30, 40, 50};
multArray(array,SIZE);
```

Anropet innebär att den formella parametern **a** pekar på samma utrymme som arrayen **array.**

array, a

10	20	30	40	50

Hela programmet :

```
// func_150
// Funktion med array som parameter
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>   // cout
#include <iomanip>    // setw()
using namespace std;
//--------------------------------------------------------------------------
// Funktionsprototyper
//--------------------------------------------------------------------------
void multArray(int a[], size_t num);
void showArray(int a[], size_t num);

//--------------------------------------------------------------------------
// Huvudprogram
//--------------------------------------------------------------------------
int main()
{
  //Deklarera och initiera en array
  const size_t SIZE = 5;
  int arr[SIZE] = {10, 20, 30, 40, 50};

  // Skriv arrayen
  cout <<         "Array before  manipulation:" ;
  showArray(arr, SIZE);
```

```
// Multiplicera arrayen
  multArray(arr, SIZE);

  // Skriv arrayen efter manipulation
  cout << endl << "Array after manipulation  :" ;
  showArray(arr, SIZE);

  cout << endl << endl;
  return 0;
}
//-----------------------------------------------------------------------------
// Funktionsdefinition
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// multArray
//-----------------------------------------------------------------------------
// Uppgift: Varje element i arrayen a multipliceras med faktorn 10
// Indata : a - heltalsarray
//          nr (int) - antalet element i arrayen
// Utdata : a - heltalsarray
//-----------------------------------------------------------------------------
void multArray(int a[], size_t num)
{
  for(size_t i=0; i < num; i++)
    a[i] *= 10;        // Multiplicera varje element med faktorn 10;

}
//-----------------------------------------------------------------------------
// showArray
//-----------------------------------------------------------------------------
// Uppgift: Skriv ut num element i arrayen a
// Indata : a - heltalsarray
//          num (int) - antalet element i arrayen
// Utdata : -
//-----------------------------------------------------------------------------
void showArray(int a[], size_t num)
{
  for(size_t i=0; i < num; i++)
    cout << setw(5) << a[i];
}
```

Kommentarer:

Alternativt skrivsätt till `void multArray( int a[], int num);` är

```
void multArray( int *a, int num);
```

7.3.13 const-deklarerade funktionsparametrar

Om man vill att en funktion **inte** ska kunna ändra på en array (pekare) får man använda const-deklaration: `int arraySum(const int a[], int num);`
Här ser **const** till att arrayen a inte får ändras inne i funktionen. Kompilatorn säger ifrån om man som programmerare försöker ändå.

arraySum är tänkt att summera elementen i arrayen **a**. Man vill se till att arrayen **a** inte ändras.

```
int arraySum( const int a[],size_t num)
{
  int sum = 0;
  for (size_t i=0;i<num;i++)
    sum += a[i];

  return sum;
}
```

Anrop:

```
int array[SIZE] = {10, 20, 30, 40, 50};
cout << endl << "Summan av arrayens element  = " << arraySum(array,SIZE);
```

Hela programmet:

```
// func_160
// Funktioner med const-deklarerad array som parameter
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>   // cout
#include <iomanip>    // setw()
using namespace std;

//-----------------------------------------------------------------------------
// Funktionsprototyper
//-----------------------------------------------------------------------------
int arraySum( const int a[], size_t num);
void showArray(const int a[], size_t num);

//-----------------------------------------------------------------------------
// Huvudprogram
//-----------------------------------------------------------------------------
int main()
{

  //Deklarera och initiera en array
  const size_t SIZE = 5;
  int arr[SIZE] = {10, 20, 30, 40, 50};

  // Skriv arrayen
  cout << "Array : " ;
  showArray(arr, SIZE);

  // Summera arrayen och skriv resultatet
  cout << endl << "Sum of the elements in the array = " << arraySum(arr, SIZE);

  return 0;
}
```

```
//-----------------------------------------------------------------------------
// Funktionsdefinitioner
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// arraySum
//-----------------------------------------------------------------------------
// Uppgift: Summerar elementen i arrayen a
// Indata : a - heltalsarray
//          num (int) - antal element i arrayen
// Utdata : arrayelementens summa (int)
//-----------------------------------------------------------------------------
int arraySum(const int a[], size_t num)
{
  int sum = 0;
           for (size_t i=0;i<num;i++)
    sum += a[i];
  return sum;
}
//-----------------------------------------------------------------------------
// showArray
//-----------------------------------------------------------------------------
// Uppgift: Skriv ut num element i arrayen v
// Indata : a - heltalsarray
//          num (int) - antalet element i arrayen
// Utdata : -
//-----------------------------------------------------------------------------
void showArray(const int a[], size_t num)
{
           for(size_t i=0; i < num; i++)
    cout << setw(5) << a[i];
}
```

Kommentarer:

- Lägg in följande rad i funktionen showArray() och kompilera. v[0] = 2;

7.3.14 Mata in data till en array i en funktion

Vi ska här skapa en funktion som använder sig av både returvärde och en array som parameter. Funktionens uppgift är att låta användaren mata in data till en array. Maximalt antal element bestäms vid deklaration av arrayen, men det aktuella antalet element (<= maxAntal) som matas in bestäms av användaren.

Indata: arr, maxAntalElement, aktuelltAntalElement
Utdata: arr, altuelltAntalElement

Funktionsdefinition:

```
void fillArray(double a[],int maxNum, int &curNum)
{
  if(curNum < maxNum)
  {
    cout << " Input element nr " << curNum + 1<<": ";
    cin >> a[curNum];
    curNum++;
  }
  else
    cout << "The arrayen is full";
}
```

Kommentarer:

- variabeln `curNum` räknar antalet inmatade element. Detta antal returneras också i `curNum` eftersom den är referensdeklarerad!

Anrop:

```
  const int MAX = 5;
  double arr[MAX];
  int elementNum;
  fillArray(arr,MAX,elementNum);
```

Efter detta anrop har användaren matat in *elementNum* stycken element i arrayen *arr*.

För att göra programmet helt funktionsbaserat har jag skapat funktioner för

- utskrift av arrayen: showArray()
- manipulering av arrayen (mult. varje element med en faktor): multArray()

Dessa funktioner finns förklarade i föregående exempelprogram. Se även funktionsdefinitionerna nedan.

Hela programmet:

```
// func_170
//-----------------------------------------------------------------------------
// func_170
// Funktioner med en array som parameter
// Mata in data till en array i en funktion
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>   // cout
#include <iomanip>    // setw(), fixed, setprecision()
using namespace std;

// Konstant för maximal arraystorlek
const size_t MAX = 5 ;
```

```
//-----------------------------------------------------------------------------
// Funktionsprototyper
//-----------------------------------------------------------------------------
void fillArray(double a[], size_t &curNum);
void showArray(const double a[], size_t n);
void multArray(double fact, double a[], size_t n);

//-----------------------------------------------------------------------------
// Huvudprogram
//-----------------------------------------------------------------------------
int main()
{

  // Deklarera konstanter och variabler

  double arr[MAX];
  size_t elementNum = 0;

  // Mata in data i arrayen
  fillArray(arr, elementNum);
  fillArray(arr, elementNum);
  fillArray(arr, elementNum);

  // Skriv vektorn
  showArray(arr, elementNum);

  cout << " Input more values!" << endl;
  fillArray(arr, elementNum);
  fillArray(arr, elementNum);
  fillArray(arr, elementNum);

  // Ange en faktor att multiplicera arrayen med
  double factor;
  cout << endl <<"Input factor to multiply each element with: ";
  cin >> factor;

  // Multiplicera arrayen med faktorn faktor
  multArray(factor, arr, elementNum);

  // Skriv arrayen efter manipulationen
  showArray(arr, elementNum);

  cout << endl << endl;
  return 0;
}

//-----------------------------------------------------------------------------
// Funktionsdefinitioner
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// fillArray
//-----------------------------------------------------------------------------
// Uppgift:  Mata in flyttal i arrayen
// Indata :  a - flyttalsarray
//           maxNum (int) - max antal element i arrayen
//           curNum (int) - aktuellt antal element
// Utdata :  curNum (int) - aktuellt antal element
//           a - flyttalsarray
//-----------------------------------------------------------------------------
```

```
void fillArray(double a[], size_t &curNum)
{
  if(curNum < MAX)
  {
    cout << " Input element nr " << curNum + 1<<": ";
    cin >> a[curNum];
    curNum++;
  }
  else
    cout << "The arrayen is full";
}

//--------------------------------------------------------------------------
// showArray
//--------------------------------------------------------------------------
// Uppgift: Skriv arrayens element på skärmen
// Indata : a - flyttalsarray
//          n (int) - antal element som ska skrivas ut
// Utdata : -
//--------------------------------------------------------------------------
void showArray(const double a[], size_t n)
{
  cout << endl << endl << "The content in the array:" << endl;
  cout << fixed << setprecision(2);
  for (size_t i=0; i < n; i++)
    cout << "Nr " << (i+1) << ": " << setw(6) << a[i] << endl;
}

//--------------------------------------------------------------------------
// multArray
//--------------------------------------------------------------------------
// Uppgift:  Varje element i arrayen x multipliceras med faktorn factor
// Indata : a - flyttalsarray
//          n (int) - antal element som ska multipliceras
//          factor (double) - faktorn som varje element sak multipliceras med
// Utdata : -
//--------------------------------------------------------------------------
void multArray(double factor, double a[], size_t n)
{
  for (size_t i=0; i<n; i++)
    a[i]*=factor;
}
```

Kommentarer:

- I huvudprogrammet används arrayen **arr** för att hantera arrayen och elementNum för det aktuella antalet element
- Huvudprogrammets **arr** och funktionernas formella parameter **a** delar på **samma** minnesutrymme i hela programmet.
- Olika typer av anrop:
  - **pekaranrop** (by pointer): t ex double a[] i funktionen
    `void fillArray(double a[,] size_t &curNum)`
    `(double a[]` kan ju också skrivas `double *a)`
  - **referensanrop** (by reference): int &curNum i samma funktion
  - **värdeanrop** (by value): t.ex. `size_t n` i showArray()
- Konstanten `const size_t MAX = 5` är placerad globalt för att den ska vara åtkomlig både i main() och i funktionsdefinitionerna! Det är ok, och lämpligt, att placera konstanter globalt!

7.4 Vector

Nu går vi över till den klassen vector för att utföra samma saker som vi gjort ovan med en traditionell array . Vi tjuvtittar alltså på klasser och objekt även fast dessa hör till nästa kurs. Syfte med detta är att vi ska kunna hantera arrayer, listor, på ett enklare sätt.

Klassen vector innehåller mer än elementen, data, vilket gör den enklare att användba. Det kan t.ex. nämnas att en vector själv håller reda på aktuellt antal element!

Exempel:
vector <int> y; Skapar en vector med plats för heltal
vector<float> x; Skapar en vector med plats för flyttal

OBS! Inkludera `#include <vector>`

7.4.1 Använd en vector för att lagra heltal

I följande exempel visas hur man tilldelar heltal till en vector och läser vectorns värden.

```
// vector_010
// Ett första försök med att använda vector för att hantera arrayer
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>   // cout
#include <vector>     // STL::vector
#include <iomanip>    // setw()
using namespace std;
//--------------------------------------------------------------------------
int main()
{
  const size_t SIZE = 10;

  // Deklarera en vector
  vector<int> integers1;

  // Fyll vectorn med SIZE st tal
  for( auto i=0; i < SIZE; i++)
    integers1.push_back(3*i - 2);

  // Skriv ut vectorn integers1.
  cout << "integers 1= ";
  for(auto idx : integers1)
    cout <<  setw(4) << idx;

  // Deklarera en andra vector
  vector<int> integers2;

  // Mata in godyckligt antal heltal
  cout << endl << endl;
  char ch = 'N';
  do
  {
    int num;
    cout << "Input an integer: ";
    cin >> num;
    integers2.push_back(num);
```

```
    cout << "More (y/n)? ";
    cin >> ch;
  } while('Y'== toupper(ch));

  // Skriv ut vectorn integers2.
  cout << endl << endl;
  cout << "integers 2= ";
  for( auto idx : integers2)
    cout << setw(4) << idx;

  cout << endl << endl;

  return 0;
}
```

Kommentarer:

- Deklarera en vector genom att skriva: `vector<int> integers2;`
- När man lägger in data i vectorn använder man medlemsfunktionen `push_back()`, t.ex. så här: `integers2.push_back(num);`
- Vectorn håller själv reda på hur många element man har lagt in. Antalet inlagda element fås med medlemsfunktionen *size()*, t.ex `integers2.size()`
- *size*() returnerar ett heltal av typen *size_t*, därför skriver man `for(size_t i=0; i < integers2.size(); i++)`
- Utskrift av arrayen måste ske elementvis.

7.4.2 Kopiering av en vector

Man kankopiera en vector till en annan så här:

```
y = x;
```

Hela programmet:

```
// vector_020
// Kopiera vector
// Per Ekeroot 2013-09-03
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>   // cout
#include <vector>     // std::vector
#include <iomanip>    // setw()
using namespace std;
//--------------------------------------------------------------------------
int main()
{
  // Deklarera en vector
  vector<int> integers1;

  // Fyll arrayen med SIZE heltal
  const size_t SIZE = 15;
  for( auto i=0; i < SIZE;i++)
    integers1.push_back(3*i - 2);

  // Skriv ut arrayens värden
  cout << " integers1 = ";
  for(auto idx : integers1)
    cout << setw(4) << idx;

  // Deklarera ytterligare en array
  vector<int> integers2;

  // Kopiera integers1 till intgers2
  integers2 = integers1;

// Skriv ut arrayen integers2
  cout << endl <<endl;
  cout << " integers2 = ";
  for(auto idx : integers2)
    cout << setw(4) << idx;

  cout << endl <<endl;
  return 0;
}
```

Kommentarer:

- Man kan alltså skriva `integers2 = integers1` och detta **utan** krav på att vectorerna ska vara lika stora

7.4.3 Använd en vector för att beräkna summa och medelvärde för ett stort antal tal

Här följer ett exempel som använder en vector för att beräkna summa och medelvärde för ett stort antal tal. Det här exemplet är enklare än sina motsvarigheter för array, se array_030.cpp, för att man slipper att ange maximal storlek för arrayen.

Programstruktur:

- Skapa en vector
- Låt användaren ange hur många tal som ska slumpas
- Slumpa talen i intervallet 1 – 1000 till en vector
- Skriv ut talen med 10 tal på varje rad
- Beräkna talens summa och medelvärde
- Skriv ut talens summa och medelvärde

```
// vector_030
// Använd en vector för summa- och medelvärdesberäkning
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>        // cout
#include <vector>          // std::vector
#include <iomanip>         // setw()
#include <random>          // Slumpning
#include <ctime>           // time()
using namespace std;

//-----------------------------------------------------------------------------
int main()
{
  // Deklarera en std::vector och variabler samt initiera slumpningen
  vector<int> randomIntegers;
  size_t userInputNum;
  default_random_engine generator(static_cast<unsigned>(time(0)));
  uniform_int_distribution<int> random(1,1000);//Slumpa i itervallet [1,1000]

  // Rubrik, ledtext och inmatning
  cout << " Randomize integers(1 - 1000) , calculate sum and average";
  cout <<  endl << endl <<  " How many integers do you want to randomize: " ;
  cin >> userInputNum;

  // Slumpa talen och lägg in dem i vektorn
  for(size_t i=0; i < userInputNum; i++)
    randomIntegers.push_back(random(generator));

  // Skriv talen ( 10 kolumner)
  int num=1;
  for(auto idx: randomIntegers)
  {
    cout << setw(5) << idx;
    if(num % 10 == 0)
      cout << endl;
    num++;
  }
```

```
//Beräkna summa och medelvärde
  int sum = 0;
  float average = 0;
  for(auto idx: randomIntegers)
  sum += idx;

  average = static_cast<float>(sum) / randomIntegers.size();

  // Skriv summa och medelvärde

  cout << endl << endl << " Sum     = " << sum << endl;
  cout << fixed << setprecision(2);
  cout << " Average = " << average << endl << endl;

  return 0;
}
```

7.4.4 Sökning i en vector

Sökning i en vector kan göras med en standardfunktion som heter *find()*. *find*() finns i ett bibliotek med algoritmer. Inkludera <algorithm>.

Sökningen görs så här:

```
find(vec.begin(),vec.end(),searchNumber)
```

Kommentarer:

- vec är en vector som innehåller heltal, se nedan
- vec.begin() pekar på första element i vectorn
- vec.end() pekar på ”elementet” efter det sista elementet i vectorn (bortom slutet). searchNumber är det talman söker efter i vectorn.
- find() returnerar en pekare till det element som sökt tal finns i.
- om det sökta talet inte finns i vectorn returneras en pekare = vec.end()
- Vi kommer att behandla pekare i nästa kurs. Då får du en mer detaljerad förklaring på hur pekare fungerar.

För att få reda på om det sökta talet finns i vectorn kan man deklarera en boolsk variabel och låta den få värde så här:

```
bool found = (find(vec.begin(),vec.end(),searchNumber) != vec.end() );
```

Kommentarer:

- Om find() hittar det sökta talet, searchNumber, blir **found** *true* eftersom find() då returnerar något annat än vec.end()
- Om find() inte hittar det sökta talet, searchNumber, blir **found** *false* eftersom find() då returnerar vec.end()

Hela programmet på nästa sida:

```
// vector_040
// Sök i en std::vector
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>   // cout
#include <vector>     // std::vector
#include <algorithm>  // std::find(), binary_search(), sort()
#include <random>     // slumpning
#include <string>     // string-klassen
#include <ctime>          // time()
using namespace std;
int main()
{
//--------------------------------------------------------------------------
// Deklarera en konstant och en STL::vector samt initiera slumpning
//--------------------------------------------------------------------------
  const size_t SIZE = 10;
  vector <int> vect;
  default_random_engine generator(static_cast<unsigned>(time(0)));
  uniform_int_distribution<int> random(1,10); // Slumpa i itervallet [1,10]

//--------------------------------------------------------------------------
// Slumpa tio tal till vectorn
//--------------------------------------------------------------------------
  for( size_t i=0; i < SIZE; i++)
    vect.push_back(random(generator));

//--------------------------------------------------------------------------
// Skriv vectorn på skärmen
//--------------------------------------------------------------------------
  for(auto idx : vect)
  cout << idx << "  ";
  cout << endl << endl;

//--------------------------------------------------------------------------
// Sök efter talet 7 i vectorn med find()
//--------------------------------------------------------------------------
  int searchNumber = 7;
  bool found = ( find(vect.begin(), vect.end(),searchNumber) != vect.end() );

//--------------------------------------------------------------------------
// Skriv resultatet av sökningen
//--------------------------------------------------------------------------
  if(found)
    cout << searchNumber << " is  in the vector!";
  else
    cout << searchNumber << " is  NOT in the vector!";
  cout << endl << endl;

  return 0;
}
```

7.4.5 Sortering av en vector

Sortering i en vector kan göras på samma sätt som man gör en sortering i en ”vanlig” array. Men i en vector har man även här hjälp av algoritmen sort(). En förutsättning när man använder sort() är att jämförelseoperatorn < (mindre än) måste vara definierad för den datatyp man vill sortera.

Så här enkelt blir det:

```
  sort(integers.begin(),integers.end());
```

Kommentar:

- Sorterar vectorn *integers* från första till sista element
- Sorteringen görs med < vilket innebär att sorteringen görs stigande

Denna sorteringsalgoritm ingår i nästa exempel:

```
// vector_050
// Programmet demonstrerar sortering std::sort() med vector
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>
#include <vector>          // std::vector
#include <algorithm>       // std::sort()
#include <random>
#include <iomanip>
#include <ctime>
using namespace std;
//--------------------------------------------------------------------------
int main()
{
//--------------------------------------------------------------------------
// Deklarera konstant och initiera slumpning
//--------------------------------------------------------------------------
  const size_t SIZE = 150;
  vector <int> integers;
  default_random_engine generator(static_cast<unsigned>(time(0)));
  uniform_int_distribution<int> random(0,999); // Slumpa i itervallet [0,999]

//--------------------------------------------------------------------------
// Slumpa värden till vectorn  (0 - 999)
//--------------------------------------------------------------------------
  for(size_t i=0; i<SIZE; i++)
    integers.push_back(random(generator));
//--------------------------------------------------------------------------
// Skriv osorterad vector
//--------------------------------------------------------------------------
  cout << "Not sorted vector" << endl;
  int num=1;                              // Räknare som styr antal tal per rad
  for(auto idx: integers)
  {
    cout << setw(4) << idx;
    if(num % 15 == 0)                     // Skriv 15 tal på varje rad
      cout << endl;
    num++;
  }
//--------------------------------------------------------------------------
// Sortera vectorn med den sorteringsfunktion som finns i std::algorithm
//--------------------------------------------------------------------------
  sort(integers.begin(),integers.end());
//--------------------------------------------------------------------------
```

```
// Skriv sorterad vector
//-------------------------------------------------------------------------
  cout << endl << endl << "Sorted vector" << endl;
  num=1;                      // Ettställ räknaren som styr antal tal per rad
  for(auto idx: integers)
  {
    cout << setw(4) << idx;
    if(num % 15 == 0)                   // Skriv 15 tal på varje rad
      cout << endl;
    num++;
  }

  return 0;
}
```

7.4.6 Leta rätt på största och minsta värde i en vector

Vi ska nu använda funktionerna max_element() och min_element() för att söka max- respektive minvärde i en vector:

```
*(max_element(integers.begin(), integers.end()))
*(min_element(integers.begin(), integers.end()))
```

- max_element() och min_element() letar i arrayen *integers* från begin() fram till end()
- max_element() och min_element() returnerar en pekare till det element där respektive värde finns. För att få reda på vilket värdet skriver man * framför pekaren.
- Repetition av ”kort förtydligande” av pekare
    - en pekarvaribel (pekare), *pek*, innehåller adressen till en minnescell där data ligger
    - för att komma åt värdet som finns i cellen som pek pekar på skriver man **pek*

Hela programmet:

```
// vector_060
// Sök efter max- och minvärde i en vector
// Per Ekeroot 2013-09-01
// Ver 9
//-------------------------------------------------------------------------
#include <iostream>
#include <vector>          // std::vector
#include <algorithm>       // std::sort()
#include <random>          // Slumpning
#include <iomanip>
#include <ctime>
using namespace std;
//-------------------------------------------------------------------------
int main()
{
//-------------------------------------------------------------------------
// Deklarera konstant och vector samt initiera slumpningen.
//-------------------------------------------------------------------------
  const size_t SIZE = 150;
  vector <int> integers;
  default_random_engine generator(static_cast<unsigned>(time(0)));
  uniform_int_distribution<int> random(0,999); // Slumpa i itervallet [0,999]
//-------------------------------------------------------------------------
// Slumpa värden till vectorn  (0 - 999)
//-------------------------------------------------------------------------
  for(size_t i=0; i<SIZE; i++)
    integers.push_back(random(generator));
```

```
//-------------------------------------------------------------------------------
// Skriv ut vectorn
//-------------------------------------------------------------------------------
  cout << "Vector integers:" << endl;
  int num=1;
  for(auto idx: integers)
  {
    cout << setw(4) << idx;
    if(num % 15 == 0)                    // Skriv 15 tal på varje rad
      cout << endl;
    num++;
  }

  cout << endl << endl;
  cout << "Maximum element in the vector is: " <<
    *(max_element(integers.begin(), integers.end())) << endl;
  cout << "Minimum element in the vector is: " <<
    *(min_element(integers.begin(), integers.end())) << endl << endl;

  return 0;
}
```

7.4.7 En vector med strängar

Sök i en vector med strängar och sortera en vector med strängar.

Deklarera en vector med strängar och lägg in några namn:

```
vector <string> names
names.push_back("Ulla Andersson");
names.push_back("Peter Lundahl");
names.push_back("Karin Halvarsson");
names.push_back("Sven Karlsson");
names.push_back("Eva Tengdahl");
names.push_back("Bo Tengdahl");
```

Gör en sökning i arrayen

```
found =  (find(names.begin(), names.end(),searchName) != names.end() );
```

Kommentar:

- Jämförelse görs tecken för tecken mellan searchName och de namn som ligger i vectorn
- Jämförelsen är case sensitive

Sortera arrayen

```
sort(names.begin(), names.end());
```

Kommentar:

- Namnen som ligger i arrayen är av datatypen (klassen) string
- < är definierad för string-klassen
- Jämförelse av två namn görs tecken för tecken. Det är tecknens ASCII-värden som jämförs
- Jämförelsen är case sensitive
- Eftersom namnen är inlagda på formatet *förnamn efternamn* (OBS mellanslag mellan namnen!) sorteras namnen på förnamn och om de är lika sorteras de på efternamn

Vi ska nu sortera namnen på efternamn och om de är lika på förnamn. För att fixa detta skriver vi en funktion som byter plats på förnamn och efternamn:

```
string lastName_firstName(string name)
{
  size_t spacePos = name.find(" ");
  string fn = name.substr(0 ,spacePos);  // Plocka ut förnamnet
  string ln = name.substr(spacePos+1, name.size()-spacePos-1); //Plocka ut enamn
  return ln + " " + fn;
}
```

Kommentarer:

- `spacePos` anger index för mellanslaget i namnsträngen
- strängvariabeln fn (firstName) tilldelas en delsträng ur *name*, tecknen från 0 till `spacepos`
- strängvariabeln ln (lastName) tilldelas en delsträng ur *name*, tecknen från `spacepos+1` till `name.size()-spacePos-1`
- till sist returneras strängen `ln + ” ” + fn`
- Observera mellanslaget mellan ln och fn. Det är viktigt för att sorteringen ska göras rätt!

Sedan använder lastName_firstName() i en funktion som jämför två namn och rturnerar true om namn 1 är mindre än namn2. Tack vare lastName_firstName() jämförs efternamnen och om de är lika förnamnen.

```
bool nameLT(const string & n1, const string & n2)
{
  return lastName_firstName(n1) < lastName_firstName(n2);
}
```

Kommentar:

- Funktionen förutsätter att namnen n1 och n2 skrivs på formatet *förnamn efternamn*

Till slut använder vi en variant av sorteringsfunktionen sort() som ser ut så här:

```
sort(names.begin(), names.end(),nameLT);
```

Kommentarer:

- Den här varianten av sort() tar en parameter till, jämförelsefunktionen nameLT()
- Den talar om för sort() hur jämförelsen ska göras!!
- Observera att man enbart skriver funktionens namn, inga parametrar, inga parenteser!!
- Att använda jämförelsefunktioner på detta sätt är **mycket** användbart!

Hela programmet:

```
// string_070_vector
// En namnlista i form av en vector med strängar.
// Sök i namnlistan
// Sortera namnlistan
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>   // cout, cin
#include <string>     // string
#include <vector>     // std::vector
#include <algorithm>  // std::find(), std::sort()
using namespace std;
//--------------------------------------------------------------------------
// Funktionsprototyper
//--------------------------------------------------------------------------
string lastName_firstName(string name);
bool nameLT(const string & n1, const string & n2);
```

```
int main()
{
//-----------------------------------------------------------------------------
// Deklarera och initiera en vector med strängar
//-----------------------------------------------------------------------------
  vector <string> names;
  names.push_back("Ulla Andersson");
  names.push_back("Peter Lundahl");
  names.push_back("Karin Halvarsson");
  names.push_back("Sven Karlsson");
  names.push_back("Eva Tengdahl");
  names.push_back("Bo Tengdahl");

//-----------------------------------------------------------------------------
// Skriv de inlagda namnen på skärmen
//-----------------------------------------------------------------------------
  for(auto idx: names)
    cout << idx << endl;

//-----------------------------------------------------------------------------
// Sök efter namn som användaren matar in
//-----------------------------------------------------------------------------
  bool   found ;
  string searchName;
  char   ch;

  do
  {
    // Mata in söknamn
    cout << endl <<  "Search for: ";
    getline(cin,searchName);

    // Leta i arrayen name efter söknamnet
    found =  ( find(names.begin(), names.end(),searchName) != names.end() );

    // Skriv sökresultatet
    cout << endl << searchName ;
    if(found)
      cout << " is in the vector" << endl << endl;
    else
      cout << " is NOT in the vector" << endl <<endl;

    // Ska sökningen upprepas
    cout << "New search? (y/n): ";
    cin >> ch;
    cin.get();           // Läs bort ENTER från inströmmen
    ch = toupper(ch);
  }while(ch == 'Y');

// Lägg till ett  namn
  names.push_back("Anders Karlsson");

//-----------------------------------------------------------------------------
// Sortera namnen på förnamn
//-----------------------------------------------------------------------------
  sort(names.begin(), names.end());
```

```
//-------------------------------------------------------------------------------
// Skriv de sorterade namnen
//-------------------------------------------------------------------------------
  cout << endl << "Sorted on firstname: " << endl;
  for(auto idx: names)
    cout << idx << endl;


//-------------------------------------------------------------------------------
// Sortera namnen på efternamn
//-------------------------------------------------------------------------------
  sort(names.begin(), names.end(),nameLT);

//-------------------------------------------------------------------------------
// Skriv de sorterade namnen på skärmen
//-------------------------------------------------------------------------------
  cout << endl << "Sorted on lastname: " << endl;
  for(auto idx: names)
    cout << idx << endl;

return 0;
}

//-------------------------------------------------------------------------------
// lastName_firstName
// Uppgift: Byt plats på förnamn och efternamn
// Indata : en sträng med ett namn på formatet 'förnamn efternamn'
//          Observera mellanslaget mellan förnamn och efternamn!
// Utdata : en sträng på formatet 'efternamn förnamn'
//          Observera mellanslaget mellan efternamn och förnamn!
//-------------------------------------------------------------------------------
string lastName_firstName(string name)
{
  size_t spacePos = name.find(" ");
  string fn = name.substr(0 ,spacePos);  // Plocka ut förnamnet
  string ln = name.substr(spacePos+1, name.size()-spacePos-1); //Plocka ut enamn
  return ln + " " + fn;   //Skapa en sträng med efternamn + ' ' + förnamn
}
//-------------------------------------------------------------------------------
// nameLT (LT = Less Than)
// Uppgift : Jämför namnen n1 och n2
// Indata  : Namnen n1 och n2
// Utdata  : True om n1 < n2 .
//          Först jämförs efternamnen om de är lika jämförs förnamnen.
//          Funktionen lastName_firstName() byter plats på förnamn och efternamn
//-------------------------------------------------------------------------------
bool nameLT(const string & n1, const string & n2)
{
  return lastName_firstName(n1) < lastName_firstName(n2);
}
```

7.4.8 Funktioner med en vector som parameter

Vi har tidigare i kursen sett hur man kan ha funktioner med parametrar av datatyperna, int, float, string. Hur gör man för att skicka med en vector som parameter (argument) till en funktion? Här kommer ett exempel i vilket man skapar en funktion som multiplicerar varje element i en vector med ett angivet tal och returnerar den manipulerade vectorn.

Funktionsdefinition:

```
vector <int> multVector(vector <int> v)
{
  vector <int> res;
  for(auto idx: v)              // Multiplicera varje element i v med faktorn 10;
    res.push_back(idx * 10);    // Lägg resultatet i vectorn res och returnera

  return res;
}
```

Kommentar:

- Vi behöver inte ha någon parameter som anger hur många element som ska multipliceras eftersom vector själv håller reda på sin storlek.

Man kan behandla en vectorpå samma sätt som en variabel, t.ex. med datatypen int, när man skickar den till och från funktioner. Den manipulerade vectorn returneras i det här fallet från funktionen i funktionsnamnet. Anrop:

```
vector <int> integers;
//. . . ge värden till integers
integers = multVector(integers);
```

Hela programmet :

```
// func_150_vector
// Funktion med std::vector som parameter
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>   // cout
#include <string>
#include <vector>          // std::vector
#include <iomanip>    // setw()
using namespace std;

//-----------------------------------------------------------------------------
// Funktionsprototyper
//-----------------------------------------------------------------------------
  void showVector(vector <int> v);
  vector <int> multVector(vector <int> v);

// Huvudprogram på nästa sida:
```

```
int main()
{
  //Deklarera och ge värden till en vector
  vector <int> integers;
  integers.push_back(10);
  integers.push_back(20);
  integers.push_back(30);
  integers.push_back(40);
  integers.push_back(50);
  integers.push_back(75);


  // Skriv vectorn
  cout << "Vector before  manipulation:" ;
  showVector(integers);

  // Multiplicera vectorn
  integers = multVector(integers);

  // Skriv vectorn efter manipulation
  cout << endl << "Vector after manipulation  :" ;
  showVector(integers);

  cout << endl << endl;
  return 0;
}
//-----------------------------------------------------------------------------
// Funktionsdefinitioner
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// showVector
//-----------------------------------------------------------------------------
// Uppgift: Skriv ut alla element i vectorn v
// Indata : v - heltalsvector
// Utdata : -
//-----------------------------------------------------------------------------
void showVector(vector <int> v)
{
  for(auto idx: v)
    cout << setw(5) << idx;
}
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// multVector
//-----------------------------------------------------------------------------
// Uppgift: Varje element i vectorn v multipliceras med faktorn 10
// Indata : v - heltalsvector
// Utdata : heltalsvectorn
//-----------------------------------------------------------------------------
vector <int> multVector(vector <int> v)
{
  vector <int> res;
  for(auto idx: v)             // Multiplicera varje element i v med faktorn 10;
    res.push_back(idx * 10);    // Lägg resultatet i vectorn res och returnera

  return res;
}
```

7.4.9 Mata in data till en vector i en funktion

Vi ska här skapa en funktion som vars uppgift är att låta användaren mata in data till en vector. Hanteringen förenklas när vi använder vector!

Indata: vec,
Utdata:vectorn v

Funktionsdefinition:

```
vector <double> fillVector(vector <double> v)
{
  double num;
  cout << " Input a float number: ";
  cin >> num;
  v.push_back(num);

  return v;
}
```

Kommentarer:

- Data läggs in med push_back()!
- vector håller själv reda på sin storlek

Anrop:

```
vector <double> floatNumbers;
floatNumbers = fillVector(floatNumbers);
```

För att göra programmet helt funktionsbaserat har jag skapat funktioner för

- utskrift av arrayen: showVector()
- manipulering av arrayen (multiplicera varje element med en faktor): multVector()

Dessa funktioner finns förklarade i föregående exempelprogram för array. Se även funktionsdefinitionerna nedan.

Hela programmet:

```
// func_170_vector
// Funktioner med en vector som parameter
// Mata in data till en vector i en funktion
// Per Ekeroot 2013-09-01
// Ver 9
//-------------------------------------------------------------------------
#include <iostream>   // cout
#include <vector>     // std::vector
#include <iomanip>    // setw(), fixed, setprecision()
using namespace std;

//-------------------------------------------------------------------------
// Funktionsprototyper
//-------------------------------------------------------------------------
vector <double> fillVector(vector <double> v);
void showVector(vector <double> v);
vector <double> multVector(vector <double> v, double f);

// Huvudprogram på nästa sida:
```

```
int main()
{

  // Deklarera konstanter och variabler
  vector <double> floatNumbers;

  // Mata in data i arrayen
  floatNumbers = fillVector(floatNumbers);
  floatNumbers = fillVector(floatNumbers);
  floatNumbers = fillVector(floatNumbers);

  // Skriv vectorn
  showVector(floatNumbers);

  cout << " Input more values!" << endl;
  floatNumbers = fillVector(floatNumbers);
  floatNumbers = fillVector(floatNumbers);

  // Ange en faktor att multiplicera arrayen med
  double factor;
  cout << endl <<"Input factor to multiply each element with: ";
  cin >> factor;

  // Multiplicera arrayen med faktorn faktor
  floatNumbers = multVector(floatNumbers, factor);

  // Skriv arrayen efter manipulationen
  showVector(floatNumbers);

  return 0;
}

//--------------------------------------------------------------------------------
// Funktionsdefinitioner
//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
// fillVector
//--------------------------------------------------------------------------------
// Uppgift:  Mata in flyttal i arrayen
// Indata :  v - vector med flyttal
// Utdata :  v - vector med flyttal
//--------------------------------------------------------------------------------
vector <double> fillVector(vector <double> v)
{
  double num;
  cout << " Input a float number: ";
  cin >> num;
  v.push_back(num);

  return v;
}
```

```
//-------------------------------------------------------------------------------
// showVector
//-------------------------------------------------------------------------------
// Uppgift: Skriv arrayens element på skärmen
// Indata : a - flyttalsarray
//          n (int) - antal element som ska skrivas ut
// Utdata : -
//-------------------------------------------------------------------------------
void showVector(vector <double> v)
{
  cout << endl << endl << "The content in the vector:" << endl;
  cout << fixed << setprecision(2);
  int i=1;
  for(auto idx : v)
    cout << "Nr " << i++ << ": " << setw(6) << idx << endl;
}

//-------------------------------------------------------------------------------
// multVector
//-------------------------------------------------------------------------------
// Uppgift:  Varje element i vectorn x multipliceras med faktorn factor
// Indata : v - flyttalsvector
//          factor (double) - faktorn som varje element sak multipliceras med
// Utdata : v - flyttalsvector
//-------------------------------------------------------------------------------
vector <double> multVector(vector <double> v, double f)
{
  vector <double> res;
  for(auto idx : v)
    res.push_back(idx*f);

  return res;
}
```