Innehållsförteckning

5 Strängar

En sträng är en följd av tecken, t.ex. ”a2sQ”. I C++ avgränsar man en sträng med ” i början och i slutet. Vi kommer här att först behandla strängar med **string**-klassen och senare C-strängar (teckenarrayer), vilket är det äldre, traditionella sättet att hantera strängar. C-strängar tas upp i lektion 11 Teckenarrayer.

5.1 Strängbegreppet

En sträng är alltså en följd av tecken, datatypen char, som placeras efter varandra i minnet. Varje tecken är åtkomligt med ett index. Första tecknet i strängen har index = 0. En sådan konstruktion kallas för array. Arrayer behandlas längre fram i kursen.

Index	0	1	2	3
Data	’a’	’2’	’S’	’Q’

5.2 string-klassen

I nyare kompilatorer finns *klassen* **string**. Inkludera headerfilen string (se exempel string_010). OBS! Här fungerar det inte med string.h .

Med klassen string kan man t ex:

– deklarera strängvariabler

```
string s1,s2,s3;
```

Man kan alltså se string som en datatyp för hantering av strängar.

– tilldela strängvariablerna data och slå ihop dessa med en sträng (mellanslag):

```
s1 = “Olle”;
s2 = “Persson”;
s3 = s1+ “ “+s2;
```

s3 blir nu “Olle Persson”

– deklarera och initiera strängar med:

```
string namn = ”Olle”;
```

Man behöver inte ange en (max) stränglängd utan den är dynamisk, den anpassar sig till vad som behövs.

Åtkomst av ett tecken görs med s.k. index. Index för första tecknet är noll (0) och för sista tecknet antal tecken i strängen – 1!!

cout << s3[5]; ger ett P på skärmen

Index	0	1	2	3	4	5	6	7	8	9	10	11
Data	'O'	'l'	'l'	'e'	' '	'P'	'e'	'r'	's'	's'	'o'	'n'

Observera att innehållet i elementet med index = 4, är ett mellanslag (ASCII-kod = 32)

Inmatning och manipulering av strängar
Ett inledande exempel på hur string-klassen kan användas:

```
// string_010
// Använd string-klassen för att hantera strängar
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>   // cout, cin
#include <string>     // string-klssen
using namespace std;

//-----------------------------------------------------------------------------
int main()
{
//-----------------------------------------------------------------------------
//Deklarera tre strängvariabler
//-----------------------------------------------------------------------------
  string firstName,lastName,name;

//-----------------------------------------------------------------------------
// Mata in förnamn och efternamn
//-----------------------------------------------------------------------------
  cout << "Firstname: ";
  cin >> firstName;

  cout << "Lastname : ";
  cin >> lastName;

//-----------------------------------------------------------------------------
// Sätt ihop för- och efternamn till en sträng
//-----------------------------------------------------------------------------
  name = firstName + " " + lastName;

//-----------------------------------------------------------------------------
// Skriv ut strängarna
//-----------------------------------------------------------------------------
  cout << endl;
          cout << "Firstname    : " << firstName << endl;
  cout << "Lastname     : " << lastName << endl;
  cout << "Complete name: " << name << endl << endl;

  return 0;
}
```

Kommentarer:

- Inkludera #include <string> för att du ska kunna använda datatypen string
- Inmatning av strängar görs (bl.a.) med cin >> firstName;
- cin klarar inte av strängar med mellanslag. Mellanslag är ett av flera s.k. ”white spaces” (Enter, tabb, EOL, EOF). Detta problem löser vi längre fram.
- Slå ihop (konkatenera) strängar med + - operatorn
- Tilldelning av strängar görs med =. Ex: name = ”Eva”;
- Skriv ut med cout << firstName;
- Testa med förnamnet: ”Ulla Marie”!

5.3 Inmatning av hel rad

I nästa exempel löser vi problemet med mellanslag i strängen:

```
// string_020
// Använd getline() för att mata in en rad
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>   // cout, cin
#include <string>     // string-klassen
using namespace std;
//--------------------------------------------------------------------------
int main()
{
//--------------------------------------------------------------------------
//Deklarera en strängvariabel
//--------------------------------------------------------------------------
  string name;

//--------------------------------------------------------------------------
// Mata in förnamn och efternamn
//--------------------------------------------------------------------------
  cout << "Name         : ";
  getline(cin,name);

//--------------------------------------------------------------------------
// Skriv ut strängen
  cout << "Complete name: " << name << endl << endl;

  return 0;
}
```

Kommentarer:

- getline(cin, name) läser en hel rad. fram till enter, radslut (EOL)
- Syntax: `getline(inmatningsström, strängvariabel);`

5.4 Blandad inmatning av int och string

Ibland vill man blanda inmatning av heltal och strängar. Gör man på följande sätt får man problem:

```
int tal;
string s;

cin >> tal;
getline(cin,s);
```

Detta beror på att vid inmatning av heltalet med cin, lämnas ENTER kvar i inmatningsströmmen och när man kommer till nästa inmatningssats ligger detta ENTER kvar i inmatningsströmmen och getline() tror att ett ENTER är allt som ska in till strängen. Den blir alltså tom. Detta problem löses genom att man läser bort ENTER-tecknet från inmatningsströmmen med `cin.get();`, se nästa exempel.

```
// string_030
// Blandad inmatning av heltal och strängar med cin och getline()
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------------
#include <iostream>   // cout, cin
#include <string>     // string-klassen
using namespace std;
//--------------------------------------------------------------------------------
int main()
{
//--------------------------------------------------------------------------------
// Deklarera variabler
//--------------------------------------------------------------------------------
  string name;
  int shoeSize =0;

//--------------------------------------------------------------------------------
// Mata in skostorlek och namn
//--------------------------------------------------------------------------------
  cout << "Shoe size: " ;
  cin >> shoeSize;

  cin.get();     // Läser "bort" radslutstecknet (ENTER) från inmatningsströmmen

  cout << "Ange ditt namn: " ;
  getline(cin,name);

//--------------------------------------------------------------------------------
// Skriv ut skonummer och namn
//--------------------------------------------------------------------------------
  cout << endl << name << " har skonummer " << shoeSize << endl << endl;

  return 0;
}
```

Kommentarer:

- Kommentera bort cin.get() för att se vad som händer

5.5 Manipulera strängar I

I string-klassen finns det ”inbyggd” funktionalitet.

- Jämförelseoperator (<, >, <=, >=, ==, !=) kan användas för jämförelse av hela strängar
- Åtkomst av enstaka tecken görs med array-notation, t.ex. name[0] . Första tecken har position =0;
- Stränglängd kan läsas med name.size(). Strängens längd i bytes returneras.
- Strängar kan manipuleras. T ex insättning i sträng: name.insert(3,”-”); sätter in ett – i position 3 i name.
- En sammanfattning av string-klassens funktionalitet finns i "Lathund för strängar" sist i detta dokument.

Ett belysande exempel:

```
// string_040
// Visa funktionalitet i string-klassen
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>   // cout, cin
#include <string>     // string-klassen
using namespace std;

//--------------------------------------------------------------------------
int main()
{
  // Deklarera två stringvariabler
  string name1, name2;

  // Mata in förnamn och efternamn för två personer
  cout << "Name 1: ";
  getline(cin,name1);

  cout << "Name 2: ";
  getline(cin,name2);

  //Plocka ut tecken ur strängen
  cout << endl << "First character in name 1 : " << name1[0] << endl << endl;

  // Jämför likhet för strängarna name1 och name2
  if(name1==name2)
    cout << "Name 1 equals Name 2!" << endl << endl;
  else
    cout << "The strings are different!" << endl<< endl;

// Jämför strängarna name1 och name2 med operatorn mindre än (<)
  if(name1 < name2)
    cout << name1 << " is less than " << name2 << endl << endl;
  else
    cout << name1 << " is bigger or equal to "<< name2  << endl<< endl;

  // Sätt in ett "mellannamn"
  string::size_type spacePos = name1.find(" ");
  if(spacePos != string::npos)          // Kolla om det finns ett " " i namnet
    name1.insert(spacePos," \"Bullit\"");

  spacePos = name2.find(" ");
  if(spacePos != string::npos)
    name2.insert(spacePos," \"Rocket\"");
```

```
  cout << "Name 1 : " << name1 << endl;
  cout << "Name 2 : " << name2 << endl;;

  return 0;
}
```

Kommentarer :

- Mata in strängar med `getline()`
- Plocka tecken med []
- Kolla likhet med ==
- Jämför storlek med <. Strängarna jämförs tecken för tecken. Ett tecken representeras av dess ASCII-värde. Detta innebär att A (ASCII = 65) ligger före a (ASCII = 97).
- Leta mellanslag med `find(" ")`
- `find()` returnerar ett heltal med den inbyggda datatypen `string::size_type`
- Om sökning med find misslyckas returneras `string::npos`
- Sätt in sträng i sträng med `insert(pos, string)`.
- Skriv in ett namn utan mellanslag (EvaAndersson) och se vad som händer
- Istället för `string::size_type spacePos = name1.find(" ");` kan man i den nya standarden använda typspcificeraren auto. Då låter man kompilatron avgöra vilken typ variabeln ska ha. I det här fallet kan man skriva: `auto spacePos = name1.find(" ");`

Syntax:
s och t är strängar

`s.find(t);`	Söker efter *t* i *s*. Returnerar *pos* i *s* om *t* finns, annars string::npos
`s.insert(k,t);`	Skjuter in *t* i *s* med början i position *k*

5.6 Manipulera strängar II

Vi ska här visa ett sätt att konvertera siffror i en sträng till tal genom att koppla ihop strängar med strömmar.

I exemplet

```
int year, month, day;
string date = "2006 02 03";
istringstream iss(date);   // Skapa ett iss-objekt.
iss >> year>> month>> day;
```

skapas först en sträng med ett datum där årtal, månad och dag är åtskiljda med ett mellanslag (=whitespace). Sedan skapas ”iss” vilket är ett stringstream-objekt. 'iss' initieras med strängen 'date'. iss läser tecknen fram till första mellanslaget och omvandlar dessa till ett tal, fortsätter till nästa mellanslag och omvandlar, fortsäter till nästa och omvandlar.

```
iss >> year>> month>> day;
```

iss ovan skickar dessa omvandlade tal i tur och ordning till heltalsvariablerna year, month respektive day. Ovanstående finns med i exempel string_050, se nedan.

En något fylligare genomgång av hur man kopplar ihop strängar och strängströmmar visas i exempel string_060 nedan.

```
// string_050
// Ver 9
// Omvandling av string till tal med stringstream
// Datumkontroll
// Per Ekeroot 2013-09-01
//-----------------------------------------------------------------------------
#include <iostream>
#include <string>        // string, substring()
#include <sstream>       //istringstream
using namespace std;
int main()
{
//-----------------------------------------------------------------------------
// Mata in datum
//-----------------------------------------------------------------------------
  string date, yearStr, monthStr, dayStr;
  int year, month, day;

  cout << " *** CHECk DATE ***" << endl << endl;
  cout << "Input date (YYYY-MM-DD): ";
  getline(cin,date);

//-----------------------------------------------------------------------------
// Omvandla från datumsträng till tre heltal för år, månda resp. dag
//-----------------------------------------------------------------------------
  string::size_type idx = 0;
  while((idx = date.find("-")) != string::npos) // Skriv över '-' med mellanslag
  {
    if(idx != string::npos)
      date[idx] = ' ';
  }

  istringstream iss(date);   // Skapa ett"istringstream"-objekt och tilldela
                             // strängen date till det.
  iss >> year>> month>> day; // Omvandla strängen date till heltalen
                             // year, month resp day
                             // iss läser fram till ett whitespace, i detta fall
                             // ett mellanslag (space)
//-----------------------------------------------------------------------------
// Kolla att år, månad och dag har rimliga värden
//-----------------------------------------------------------------------------
  bool dateOK = true;
  if( year < 1000 || year > 3000)
    dateOK = false;

  if( month < 1 || month > 12)
    dateOK = false;

  if( day < 1 || day > 31)
    dateOK = false;

//-----------------------------------------------------------------------------
// Skriv resultat av datumkollen
//-----------------------------------------------------------------------------
  cout << endl;
  cout << "Year  = " << year << endl;
  cout << "Month = " << month << endl;
  cout << "Day   = " << day << endl;
  cout << "Is ";
  if(dateOK)
```

```
    cout << "a proper date!!";
  else
    cout << "not a proper date!!";

  return 0;
}
```

Kommentarer:

- Använd en boolsk variabel för att hålla reda på om datumet är OK.
- Man kan förfina programmet genom att använda en separat boolsk variabel för år, månad resp. dag (yearOK, monthOK, dayOK) för att kunna skriva bättre felmeddelanden
- Med hjälp av `while((idx = date.find("-")) != string::npos)` söker man i strängen date efter – och ersätter dem med ' '. Observera den "extra parentesen kring `idx = date.find("-"))`!! Den behövs av prioritetsskäl!

5.7 Talomvandling med stringstream

Med hjälp av klassen istringstream kan man omvandla strängar till tal och med ostringstream-klassen kan man omvandla tal till strängar.

Vi ska först omvandla en sträng till tal:

```
string numStr = "12 17.3";
istringstream iss(numStr);

int intNum;
double floatNum;
iss >> intNum >> floatNum;
```

- I strängen numStr finns ett heltal och ett flyttal åtskiljt av ett mellanslag. Man måste skilja talen med ett "whitespace" om de ska fungera med mer än ett tal i strängen.
- Skapa ett istringstream-objekt, *iss*, och initiera det med strängen som ska omvandlas.
- *iss* läser fram till första mellanslaget och omvandlar dessa tecken till ett tal, fortsätter sedan att läsa till strängslutet och omvandlar dessa tecken till ett tal.
- Med satsen `iss >> intNum >> floatNum;` strömmas de omvandlade talen till variablerna intNum resp. floatNum.

Sedan ska vi hantera en sträng som innehåller både ett tal och en text:

```
string mixedStr= "25 Grader Celsius", outStr ;
iss.clear();
iss.str(mixedStr);
iss >> intNum ;
getline(iss,outStr);
```

- I strängen mixedStr finns först ett heltal och sedan en text med bl a mellanslag
- iss.clear() tömmer iss-objektet. Detta måste göras eftersom objektet har använts en gång redan.
- iss.str(mixed) laddar objektet med den nya strängen.
- iss>>intNum "plockar ut" heltalet och lägger det i intNum
- getline(iss, outStr); läser all tecken till strängens slut och lägger dessa i outStr

Till slut ska vi omvandla tal till en sträng

```
int n = 234;
double x = 5.1795;
ostringstream oss;
oss << "Heltalet n= " << n << " och flyttalet x = " << x;
outStr = oss.str();
```

- Först skapas och initieras en heltals- och en flyttalsvariabel.

- Sedan skapas ett ostringstream-objekt, oss.
- I satsen `oss << "Heltalet n= " << n << " och flyttalet x = " << x;` skapas en enda sträng av de två strängarna, heltalet och flyttalet
- Denna sträng tilldelas strängen outStr med outStr = oss.str() .

Här kommer allt i ett exempel:

```
// string_060
// Omvandling från sträng till tal och från tal till sträng med sstream-klassen
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>
#include <sstream>
#include <string>
#include <iomanip>
using namespace std;

int main()
{
//-----------------------------------------------------------------------------
// Konvertera från sträng till heltal och flyttal
//-----------------------------------------------------------------------------
  string numStr = "12 17.3";
  istringstream iss(numStr);              // Skapa ett istringstream-objekt och
                                          // initiera det med numStr
  int intNum;
  double floatNum;

  iss >> intNum >> floatNum;              // Konvertera från sträng till tal
  cout << "Integer = " << intNum << "  float = " << floatNum << endl;

  string mixedStr= "25 degrees Celsius", outStr ;
  iss.clear();                            // Rensa iss-objektet
  iss.str(mixedStr);                      // Ladda iss med strängen mixedStr
  iss >> intNum ;                         // Konvertera från sträng till tal
  getline(iss,outStr);                    // Kopiera från iss-objekt till sträng
                                          // Använd getline() så att läsningen
                                          // inte stannar vid mellanslaget
  cout << "Integer = " << intNum << " String =" << outStr << endl;

//-----------------------------------------------------------------------------
// Konvertera från tal till sträng
//-----------------------------------------------------------------------------
  int n = 234;
  double x = 5.1795;

  ostringstream oss;                      // Skapa ett oss-objekt

  oss << "Integer n= " << n               // Konvertera från tal till sträng
      << " och float x = " << x;

  outStr = oss.str();                     // Kopiera oss-bufferten till outStr

  cout << outStr << endl;                // Skriv ut outStr - strängen
  return 0;
}
//-----------------------------------------------------------------------------

//  sätt i setw() och fixed << setprecision(2) i oss-strömmen! Vad händer?
```

5.8 Lathund för strängar

s och s2 är string-variabler. x är en textsträng eller en teckenarray. c är av typen char.
t är en stringvariabel, en textsträng eller en teckenarray.

Deklaration

string s;	deklaration, s innehåller en tom text (med längden noll)
string s=t;	deklaration med initiering, s blir kopia av t
string s(t)	deklaration med initiering, s blir kopia av t
string s(s2, p, n);	deklaration, initierar s med n tecken från position p i s2
string s(x, n);	deklaration, initierar s med n första tecknen från x
string s(n, c);	deklaration med initiering, s tilldelas n st c:n

Tilldelning

s=t; s=c;	tilldelning
s.assign(t);	tilldelar t till s
s.assign(s2, p, n);	tilldelar till s n tecken från position i s2
s.assign(x, n);	tilldelar till s de n första tecknen från x
s.assign(n, c);	tilldelar n st c:n till s

Åtkomst av element

s[k]	indexering utan indexkontroll
s.at(k);	indexering med indexkontroll (ger s[k])
s.substr(k, n);	ger del av s, med början i position k och med längden n

Utmatning och inmatning

cout << s;	skriver ut s
cin >> s;	läser in till s, slutar vid vitt tecken
getline(cin, s);	läser in en hel rad till s, ger true om inläsningen gick bra

Radera

s.erase(k, n);	tar bort n st tecken ur s med början i position k
s.clear();	tar bort alla tecken ur s

Storlek

s.size();	ger längden av s
s.resize(n);	ändrar s:s längd till n. Fyller ut med nollor om n > den gamla längden, kapar slutet annars
s.resize(n);	som ovan, men fyller med c istället för med nollor
s.capacity();	ger den interna arrayens storlek
s.reserve(n);	anger att man kommer att behöva plats för n st tecken
s.c_str()	ger en pekare till en teckenarray med variabelns text

Jämförelser

s < t s == t s > t	jämförelser, ej alfabetiskt korrekta
s + t s + c	ger en ny string med sammanslagning av texterna
s += t s += c	lägger t eller c sist i s

Lägg till

s.append(t);	lägger t sist i s
s.append(s2, p, n);	lägger n st tecken från position p i s2 till s
s.append(x, n);	lägger de n första tecknen från x till s
s.append(n, c);	lägger till n st c:n till s

Skjut in

s.insert(k, t);	skjuter in t i position k i s
s.insert(k, s2, p, n);	skjuter in n tecken från position p i s2 till position k i s
s.insert(k, x, n);	skjuter in de n första tecknen från x i position k i s
s.insert(k, n, c);	skjuter in n st c:n i position k i s

Ersätt

s.replace(k, m, t);	ersätter tecken nr k till k+m-1 i s med t
s.replace(k, m, s2, p, n);	ersätter tecken nr k till k+m-1 i s med tecken nr p till p+n-1 i s2
s.replace(k, m, x, n);	ersätter tecken nr k till k+m-1 i s med de n första tecknen från x
s.replace(k, m, n, c);	ersätter tecken nr k till k+m-1 i s med n st c:n

Sök

s.find(t);	söker i s efter texten t, ger positionen i s om texten finns annars ges värdet string::pos
s.find(t, k);	som ovan men börjar sökningen i position k
s.find(c, n);	söker i s efter text med n st c:n. Resultat som ovan
s.rfind	som find men söker bakifrån
s.find_first_of(t);	söker i s efter första förekomsten av *något* av de tecken som finns i texten t. Resultat som find
s.find_first_of(t, k);	som ovan men börjar sökningen i position k
s.find_last_of(t);	som find_first_of, men söker bakifrån
s.find_first_not_of(t);	söker i s efter första förekomsten av *något* av de tecken som *inte* finns i texten t. Resultat som find
s.find_first_not_of(t, k);	som ovan men börjar sökningen i position k
s.find_last_not_of(t, k);	som find_first_not_of men söker bakifrån