Filhantering ANSI C/C++

Inmatning/utmatning

…men först problemet med

```
cin.getline();
```

getline() och teckenbuffern

```
// Vad händer vid inmatning?

char rad1[6];
char rad2[6];
cout << "Mata in en sträng-->";

cin.getline(rad1,6);
cout << "En till-->";
cin.getline(rad2,6);
cout << "\nRad1:" << rad1
     << "\nRad2:" << rad2;
```

getline.cpp

Filhantering

Två sätt att spara och hämta data

- Textfiler
- Binärfiler

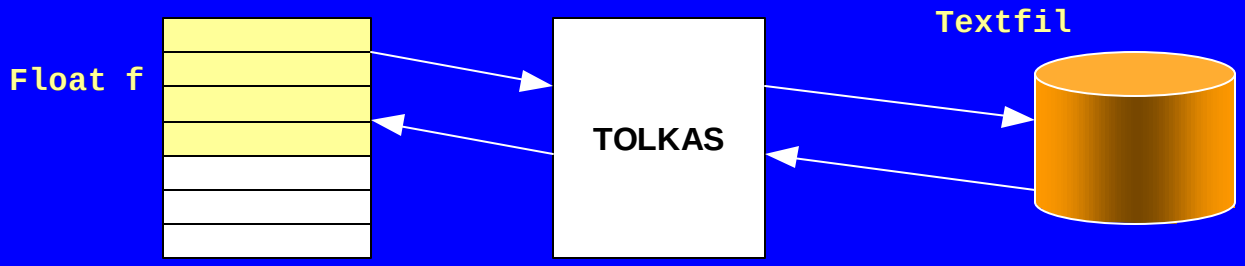
Två sätt att hantera filer

- ANSI C
  - med **`fprintf()`** och **`fscanf()`**
- ANSI C++
  - med streams **`ifstream`** och **`ofstream`**
    (**`cin`** är en **`ifstream`** , **`cout`** en **`ofstream`**)

Filtyper

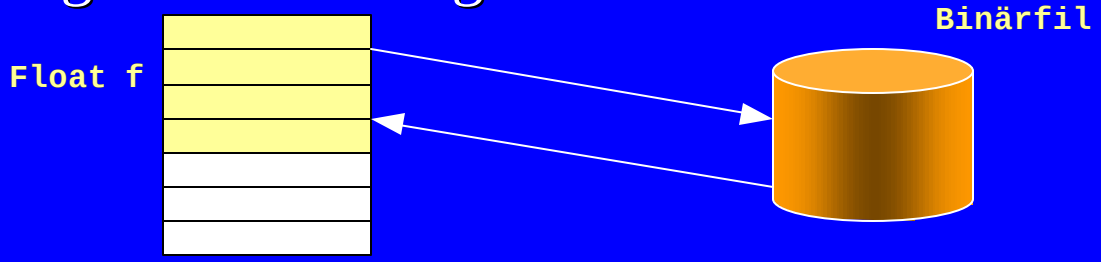
Textfiler...ASCII

Översättning (tolkning) till och från andra datatyper krävs



Binärfiler lagrar en kopia av hur data ser ut i minnet

Ingen översättningar krävs



Filhantering ANSI C

Konsolen är två textfiler

- Standard input
- Standard output

Dessa kan *omdirigeras* i operativsystemet

- från och till diskfiler / skrivare

Funktioner för konsolen:

- Man skriver till skärmen med **`printf()`**
- Man löser från tangentbodet med **`scanf()`**

Konsolen - utmatning

```
printf ("format", data, data);
```

formatsträng

argumentlista

```
int ital = 22;
float ftal = 123.78;
printf ( "Heltalet är %d och flyttalet %f", ital, ftal );
```

formatsträng

argumentlista

```
char *rad = "Detta är en teckensträng";
printf( "Skriver ut :%s", rad );
```

Formatkoder

Svarar för att översättning till ASCII sker på rätt sätt
tolkar från enkla datatyper till ASCII

Formatkoder:

```
d  Integer   signed decimal integer
i  Integer   signed decimal integer
o  Integer   unsigned octal integer
u  Integer   unsigned decimal integer
x  Integer   unsigned hexadecimal int (with a, b, c, d, e, f)
X  Integer   unsigned hexadecimal int (with A, B, C, D, E, F)
f  Floating point  signed value of the form [-]dddd.dddd.
e  Floating point  of the form [-]d.dddd or e[+/-]ddd
g  Floating point  either e or f form, based on given value
   and precision.                zeros and decimal point are
   printed if necessary.
E  Floating point  Same as e; with E for exponent.
G  Floating point  Same as g; with E for exponent if e format
```

Formatkoder

```
c  Character Single character
s  String pointer  Prints characters until a null-terminator
   is reached.
```

Inställning av fältvidd

Ett heltal placerat mellan % och formatkoden

```
”%10f”  -> (minst) 10 positioner
|    12.678|
 ^^^^^^^^^^
```

Inställning av antal decimaler

En decimalpunkt + antal decimaler mellan % och formatkod

```
”%.5f”  -> 5 decimaler
”%12.4f -> 4 decimaler i (minst) 12 positioner
”%-12.4 ->       -  ”  - ,vänsterjusterat
```

Konsolen - inmatning

```
scanf ("format", &data, &data);
```

formatsträng

argumentlista

Formatsträngen som för printf() - vissa ting tillkommer
OBS! *Adress* till data i argumenten!

```
int ital;
float ftal;
printf( ”Skriv in ett heltal och ett flyttal-->”
);

scanf( "%d%f", &ital, &ftal );
printf( "Du skrev in %d och %f", ital, ftal );
```

scanfochprintf.cpp

Textfiler i ANSI C

Deklarera en filpekare	`FILE *fp;`
Öppna filen	`fopen()`
Skriv eller läs data	`fprintf(), fscanf()`
Stäng filen	`fclose()`

```
FILE *utfil;
utfil = fopen("utdata.dat", "w");
for (int i = 0; i < 10; i++)
  fprintf("Rad %d: i kvadrat är %d", i, i*i);
fclose(utfil);
```

Öppna textfilen

`fopen ( filnamn, mode );`

`filnamn` är en teckensträng, t.ex. ”DATA.TXT”

`mode` anger text- eller binärfil, läsning eller skrivning

`fopen` returnerar en filpekare eller **`NULL`**

```
Textfil                 "t" (default)
Binärfil                "b"
För läsning             "r"
För skrivning           "w"
Append                  "a"
Öppna för skriv/läs     "r+"
Skapa för skriv/läs     "w+"
```

Öppna fil för...

```
FILE *utfil;
utfil = fopen("dbase.dat", ”rb");
   // Binärfil för läsning
utfil = fopen("dbase.dat", ”wb");
    // Binärfil för skrivning
utfil = fopen("dbase.dat", ”wb");
    // Binärfil för tillägg i slutet
utfil = fopen("dbase.txt", ”w+");
    // textfil för skrivning och läsning

// Testa om det gick bra (Kolla för NULL)
if((utfile = fopen("dbase.dat", "wb")) == NULL){
   printf("Kunde inte öppna filen");
   exit(1);
}
```

Läsning och skrivning

Att skriva och läsa textfiler i ANSI C

`fprintf()`, och `fscanf()` fungerar som `printf()` och `scanf()`

Filen måste anges

```
fprintf( utfil, format, data, data );
fscanf ( infil, format, &data, &data );
```

För typen char finns också

```
fgets() // för teckenarray (sträng)
fgetc() // för enstaka tecken
```

Skriva till textfil

```
if ( ( utFil = fopen( filnamn, "wt" ) ) == NULL )
   {
      printf( "\nFilen finns inte!" );
  // cout << "\nFilen finns inte!";
      exit(1);
   }
float a;
  for(int i = 0; i < 10; i++){
     a = (float)i;
     fprintf( utFil, "%f\t%f\n", a, a*a );
  }
fclose( utFil );
```

skrivtext.cpp

Läsa från textfil

Läsning med fgets()

```
if ( ( inFil = fopen( filnamn, "rt" ) ) == NULL ){
   printf( "\nFilen finns inte!" );
   exit(1);
}
printf( "\nHär är filens innehåll:\n\n" );
while ( fgets( rad, 81, inFil ) != NULL )
    printf( "%s", rad );
fclose( inFil );
```

lastext.cpp

fscanfochfprintf..cpp

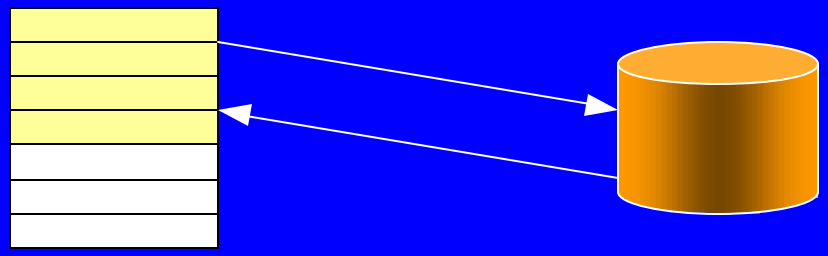
Skriva och läsa binärfiler

Kopierar block av data direkt till och från minne

Ingen omvandling

`fread()`

`fwrite()`



Läsning med

```
fread(void *buf, int antal_bytes,int antal_block, FILE *fp )
```

skrivning med

```
fwrite(void *buf, int antal_bytes,int antal_block, FILE *fp )
```

Bufferstorlek **`antal_bytes`**

bestäms ofta med **`sizeof`**-operatorn

Skriva till binärfil

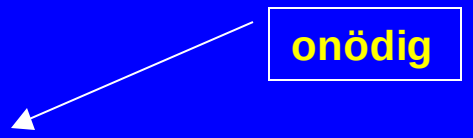
```
void main(void){
   FILE *fp;
   float a = 234.56, b = 4711.0;

   if ((fp = fopen( "test.dat", "wb” )) == NULL){
      printf("Filen kunde inte skapas\n");
      exit(1);
   }
   fwrite( &a, sizeof(float), 1, fp );
   fwrite( &b, sizeof(float), 1, fp );
   fclose( fp );
}
```

binout.cpp

Skriva struct till binärfil

```
struct Komplex{
   float re;
   float im;
};
void sparaPaFil( Komplex* vektor, FILE *fp ){
   if ((fp = fopen("strukt.dat", "wb")) == NULL){
      printf("Filen kunde inte skapas\n");
      exit(1);
   }
   printf("Saving to file");
     fwrite(vektor, sizeof(Komplex), ANTAL, fp);
}
```



struc2bin.cpp

Läsa från binärfil

```
void main(void){
   FILE *fp;
   float lista[2];

   if ((fp = fopen("test.dat", "wr")) == NULL){
      printf("Filen kunde inte skapas\n");
      exit(1);
   }
   fread( lista, sizeof(float), 2, fp );
   fclose( fp );
   printf("Tal 1 = %f\nTal 2 = %f\n",
                  lista[0], lista[1]);
}
```

binin.cpp

Filer i C++

Deklareras i **fstream.h**

```
ifstream     // för läsning från filer
ofstream     // för skrivning till filer
fstream      // för skrivning och läsning
```

Alla funktioner som finns i cin och cout
finns också i egendefinierade dataströmmar (filer)

Deklarera och öppna filer

Deklaration av filer

```
ifstream fin;
ofstream fout;
```

Öppna filer

```
fin.open(filnamn); // öppna för läsning
// eller
fstream fil;
fil.open(filnamn, mode);
```

Öppna i vilket läge (mode)

Filer kan öppnas för
- läsning eller skrivning,
- text eller binärt (default är textläge)

```
ios::binary
ios::in
ios::out
```

```
fstream dbase;
dbase.open(”dbase.dat”, ios::binary | ios::out);
dbase.seekp(0L, ios::end); // gå tilll slutet
dbase.close(); // stäng filen
```

Filens användning

Användning bestäms vid öppningen

```
ios::in          // för läsning
ios::out         // för skrivning
ios::app         // lägg till i slutet
ios::binary      // binärfil
```

Lägena kan kombineras med | (bitvis OR)

```
ios::in| ios::out // läsning och skrivning
```

Sökning i fil kan göras med **`seekp()`**

```
dbase.seekp(0L, ios::end); // gå tilll slutet
dbase.seekp(0L, ios::beg); // gå tilll slutet
```

Skriv en post på fil

```
struct PostTyp {    // Typdefinition
       char enamn[20];
       char fnamn[20];
       char tel[15];
     };
void savepost( PostTyp &post ){
   fstream dbase(filnamn, ios::binary |ios::in
                          |ios::out);
   PostTyp *p = &post;
   dbase.seekp(0L,ios::end);
// Skriv ut post
   dbase.write((char*)p,sizeof(PostTyp));
}
```

dbase.cpp