



EP578 Computing for Physicists

Topic 6 File Management

*Department of
Engineering Physics
University of Gaziantep*



Course web page
www.gantep.edu.tr/~bingul/ep578

Nov 2011

Sayfa 1

Content of this Week

Extracted from <http://cpp.gantep.edu.tr>

This week we will study the basics of:

- Overview of Streams in C++
- Format Manipulators
- Formatted Output
- File Output
- File Input

Sayfa 2

Overview of Streams in C++

The standard C++ library provides the following classes to perform I/O operations:

- `iostream` Stream class for basic I/O (without files).
- `ofstream` Stream class to output to files
- `ifstream` Stream class to input from files
- `fstream` Stream class to both read and write from/to files.



Sayfa 3

Format Manipulators

In C++ there are different ways to format input and output.

In this course we will only use format *manipulators*.

These are objects that are placed in the data stream to change the characteristics of the input or output.

For example:

```
double x = 123.4567890123;  
cout << x << endl;  
cout << fixed << setprecision(9) << x << endl;  
cout << scientific << setprecision(7) << x << endl;
```

outputs:

123.457	Default format.
123.456789012	Fixed format, with 9 decimal places.
1.2345679e+02	Scientific format with 7 decimal places.

Note: to use manipulators you need to include the `<iomanip>` header.

Sayfa 4

Some commonly used manipulators.

Requires `#include <iomanip>`

* <code>setw(n)</code>	Sets the width of the next input/output to <code>n</code> .
<code>setfill(c)</code>	Fills spaces in a number with the character <code>c</code> .
* <code>setprecision(n)</code>	Display the number with <code>n</code> decimal places.
* <code>fixed</code>	Display values in fixed-point notation.
* <code>scientific</code>	Display values in scientific notation.
<code>left</code>	Left-justify.
<code>right</code>	Right-justify.
<code>internal</code>	Left-justify the sign, right-justify the value.
* <code>dec</code>	Display integer values in decimal format.
* <code>oct</code>	Display integer values in octal (base 8) format.
* <code>hex</code>	Display integer values in hexadecimal (base 16).
<code>boolalpha</code>	Display boolean as "true" or "false".
<code>noboolalpha</code>	Display boolean as 1 or 0 (default).

Sayfa 5

Formatted Output

Manipulators are injected into the output stream to control the format of following streams of data.

In general we *combine* manipulators to obtain the desired results:

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

int main() {
    double x = 123.456;
    cout << setw(10) << fixed << setprecision(2) << x;
    cout << setw(15) << scientific << setprecision(3) << x;
}
```

Output

123.46	1.2349e+002
--------	-------------

Column 1234567890123456789012345

Sayfa 6

Earlier in the course (in “repetitive structures”) we wrote a program to output the sines and cosines of angles. We can now format that output:

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

int main() {

    cout << " x sin(x)   cos(x)" << endl;
    cout << << fixed << setprecision(6);
    for (int deg=0; deg<=90; deg+=10)
    {
        cout << setw(2) << deg << " "
             << setw(9) << sin(deg*_M_PI/180.)
             << setw(9) << cos(deg*_M_PI/180.)
             << endl;
    }
}
```

Output

x	sin(x)	cos(x)
0	0.000000	1.000000
10	0.173648	0.984808
20	0.342020	0.939693
30	0.500000	0.866025
40	0.642788	0.766044
50	0.766044	0.642788
60	0.866025	0.500000
70	0.939693	0.342020
80	0.984808	0.173648
90	1.000000	0.000000

Sayfa 7

left, right, and setfill()

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

int main() {
    int i = 1453;
    string s = "University of Gaziantep";
    cout << setw(30) << left          << s << endl;
    cout << setw(30) << right         << s << endl;
    cout << setw(10) << setfill('0') << i << endl;
}
```

Output

University of Gaziantep
University of Gaziantep
0000001453

Column 123456789012345678901234567890

30 columns

Sayfa 8

File Input/Output

Reading from and writing to files is performed using file stream classes.

`ifstream` is used for reading (inputting) data from a file.

`ofstream` is used for writing (outputting) data to a file.

These two class require the `<fstream>` header to be included.

File processing requires some basic actions:

- Stating the name of the file.
- Opening an existing file, or created a new one.
- Optionally stating the read/write mode.
- Optionally determining if a file is successfully open.
- Detecting the end of a file (when reading).
- Closing a file after is has been used.

These actions are provided by `fstream` functions...

Sayfa 9

Some functions of `fstream`

Function	Description
<code>open(filename, mode)</code>	<code>filename</code> is the name (and path) of the file to open. <code>mode</code> is an optional parameter and can have the following flags: <code>ios::in</code> open for input operations (default for <code>ifstream</code>). <code>ios::out</code> open for output operations (default for <code>ofstream</code>). <code>ios::binary</code> open in binary mode (default is text mode). <code>ios::ate</code> set the initial position at the end of the file. (default is the beginning of the file). <code>ios::app</code> append the content to the current content of the file. <code>ios::trunc</code> delete the previous content and replace with new.
<code>.is_open()</code>	Returns <code>true</code> if a file is successfully opened.
<code>.eof()</code>	Returns <code>true</code> if a file open (for reading) has reached the end.
<code>.close()</code>	Closes the file.

Sayfa 10

Writing data to a file called `numbers.txt`

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

int main() {

    int i;
    ofstream dosya("numbers.txt");

    for (i=1; i<=10; i++){
        dosya << i << " " << i*i << endl;
    }

    dosya.close();
}
```

The file `numbers.txt` contains:

```
1 1
2 4
3 9
4 16
5 25
6 36
7 49
8 64
9 81
10 100
```

Sayfa 11

Writing data to a file called `inverse.txt`

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

int main() {

    ofstream dosya("inverse.txt");

    dosya << setprecision(2) << scientific;
    for (int i=1; i<=10; i++){
        double x = 1.0/i;
        dosya << setw(3) << i
            << setw(10) << x << endl;
    }

    dosya.close();
}
```

The file `inverse.txt` contains:

```
1 1.00e+00
2 5.00e-01
3 3.33e-01
4 2.50e-01
5 2.00e-01
6 1.67e-01
7 1.43e-01
8 1.25e-01
9 1.11e-01
10 1.00e-01
```

Sayfa 12

Writing data to a file called `data.txt`

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

int main() {
    double a[4] = {8.4, 3.6, 9.1, 4.7};
    ofstream myFile("data.txt");

    for (int i=0; i<4; i++)
        myFile << a[i] << endl;

    myFile.close();
}
```

The file `data.txt` contains:

```
8.4
3.6
9.1
4.7
```

Sayfa 13

Writing data to a file called `data.txt`

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

int main() {
    double a[4] = {8.4, 3.6, 9.1, 4.7};
    ofstream myFile("data.txt");
    if ( myFile.is_open() ) {
        for (int i=0; i<4; i++)
            myFile << a[i] << endl;
        myFile.close();
    } else
        cout << "Unable to open file!";
}
```

The file `data.txt` contains:

```
8.4
3.6
9.1
4.7
```

Sayfa 14

Reading data from a file called `data.txt`

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

int main() {
    double a[4];
    ifstream myFile("data.txt");

    for (int i=0; i<4; i++) {
        myFile >> a[i];
        cout << "a[" << i << "]="
              << a[i] << endl;
    }

    myFile.close();
}
```

Given that
the file
`data.txt`
contains:

```
8.4
3.6
9.1
4.7
```

The output is

```
a[0]=8.4
a[1]=3.6
a[2]=9.1
a[3]=4.7
```

Sayfa 15

Reading data from a file called `data.txt`

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

int main() {
    double a[4];
    ifstream myFile("data.txt");

    if ( myFile.is_open() ) {

        for (int i=0; i<4; i++) {
            myFile >> a[i];
            cout << "a[" << i << "]="
                  << a[i] << endl;
        }

        myFile.close();
    } else
        cout << "Unable to open file!";
}
```

Given that
the file
`data.txt`
contains:

```
8.4
3.6
9.1
4.7
```

The output is

```
a[0]=8.4
a[1]=3.6
a[2]=9.1
a[3]=4.7
```

Sayfa 16

Reading data from a file called `student.dat`

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

int main() {
    const int n = 10;
    int     score[n];
    string  name[n];
    double  sum = 0.0;
    ifstream myFile("student.dat");

    for (int i=0; i<n; i++) {
        myFile >> name[i] >> score[i];
        sum = sum + score[i];
    }
    myFile.close();

    cout << "mean of the class = "
         << sum / n << endl;
}
```

Given that the file `student.dat` contains:

UGUR	67
KADIR	72
SEMIH	00
GOZDE	73
FIRAT	30
EDA	51
FULYA	41
GUL	67
EMRE	30
AHMET	76

The output is

```
mean of the class = 50.7
```

Sayfa 17

Example: reading periodic table data from a file

`atoms.txt`

```
2 He Helium 4.002602 NobleGas
3 Li Lithium 6.941 AlkaliMetal
4 Be Beryllium 9.012182 AlkalineEarth
5 B Boron 10.811 Metaloid
6 C Carbon 12.0107 NonMetal
7 N Nitrogen 14.00674 NonMetal
8 O Oxygen 15.9994 NonMetal
9 F Fluorine 18.998404 Halogen
11 Na Sodium 22.9898 AlkaliMetal
12 Mg Megnesium 24.305 AlkalineEarth
```

For each atom, each property is separated by a *single space*. We call this a space-delimited list of values, and reading such lists is simple in C++ (see next page).

Sayfa 18

Example: reading periodic table data from a file

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

int main () {

    const int n = 10; // read 10 atom data
    int    number[n];
    string symbol[n], name[n], type[n];
    double mass[n];

    ifstream atoms("atoms.txt");

    for (int i=0; i<n; i++)
        atoms >> number[i] >> symbol[i]
            >> name[i] >> mass[i]
            >> type[i];

    atoms.close();
    .
    . Now use the arrays
```

The format of the data is:

```
number(int)
symbol(string)
name (string)
mass (double)
type (string)
```

Here we are assuming that we have exactly **n=10** atoms in the file!

Sayfa 19

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

int main () {

    vector<int>    number;
    vector<string> symbol, name, type;
    vector<double> mass;

    ifstream atoms("atoms.txt");

    int z; string s, n, t; double m;
    while(true) {

        atoms >> z >> s >> n >> m >> t;
        if ( atoms.eof() ) break;
        number.push_back(z); symbol.push_back(s);
        name.push_back(n); mass.push_back(m);
        type.push_back(t);
    }
    atoms.close();
    .
    . Now use the vector arrays
```

If we don't know how many atoms there are in the file, then we can use **vectors** with the `.push_back` method, and the `.eof` method to detect the end of the file.

Note that we need scalars **z, s, n, m** and **t** to read values from the file and insert them into the vectors.

The file `atoms.txt` can now be expanded to incorporate more atoms without changing the program that reads the data.

Sayfa 20

Homework

Solve the following problems. You have to prepare a pdf document and sent it to me until next lecture.

E-mail: bingul[at]gantep.edu.tr (*replace [at] with @*)

Please see:

<http://www1.gantep.edu.tr/~bingul/ep578/hw-files.html>