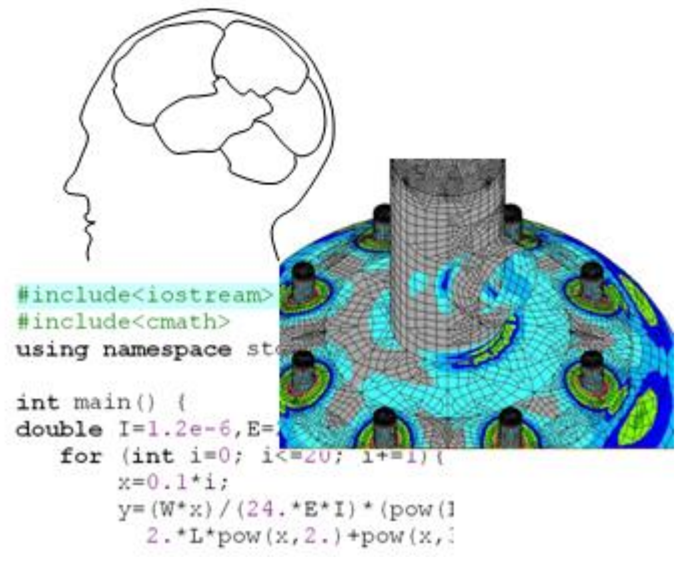
ME 110 Computation for Mechanical Engineering



```
#include<iostream>
#include<cmath>
using namespace st

int main() {
double I=1.2e-6,E=
    for (int i=0; i<=20; i+=1){
        x=0.1*i;
        y=(W*x)/(24.*E*I)*(pow(1
          2.*L*pow(x,2.)+pow(x,
```

Formatted I/O

File processing

Content of this Week

Extracted from `http://cpp.gantep.edu.tr`

This week we will study on:

- Overview of Streams in C++
- Formatted Input/Output
- Input/Output with Files

What we have done upto now?

Until now, we have used

- `cout` to write data to the screen and
- `cin` to read data from the keyboard without specifying any
format.

In this week, we deal with formatted input/output (formatted I/O);

this allows the programmer to specify how numbers/characters are displayed or read.

we will also look at how to read from and write to files (file I/O).

Overview of Streams in C++

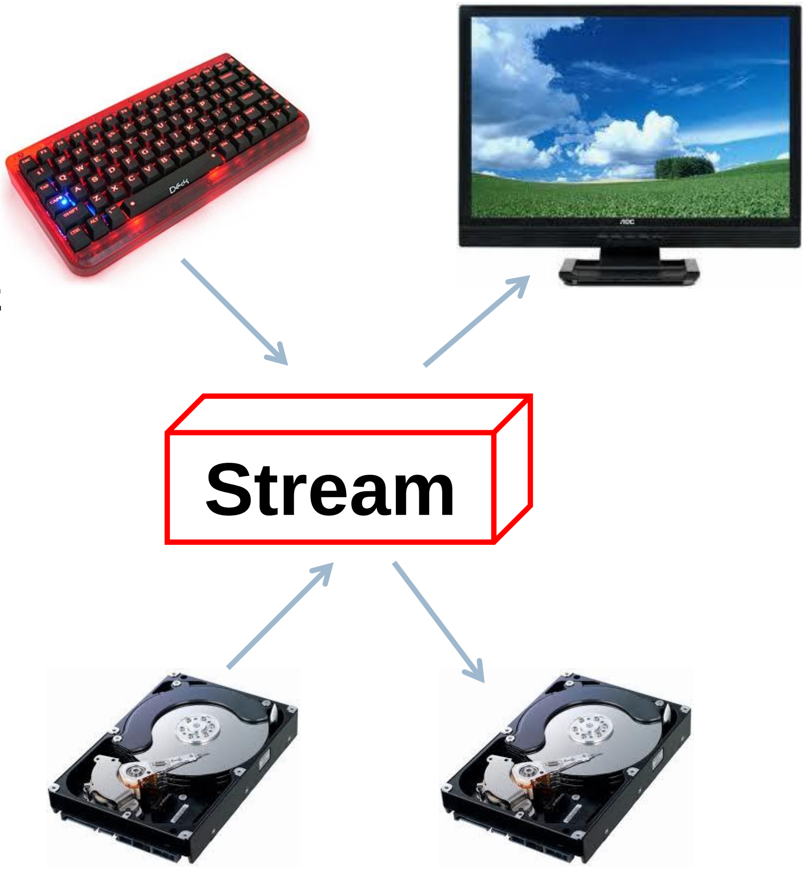
The standard C++ library provides the following classes to perform I/O operations:

`iostream` Stream class to facilitate basic I/O.

`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.



Formatted Input/Output

In C++, the I/O formatting can be performed by
- either ***manipulators***
- using the methods of ***I/O classes***
- using ***format-state flags***.

We will use manipulators in this lecture.

A ***manipulator*** is a function that can alter the characteristics of the output (and input) stream. For example,

For **`cout`**, we have seen the **`endl`** manipulator which ends a line.

Using `cout` to Format Output

The following manipulators, defined in the header file `<iomanip>`, are the most commonly used for formatting output.

`setw(n)` sets the minimum width of the next output. The length of the width is `n`.
`setfill(c)` fills leading spaces in a number with a given character `c`.
`setprecision(n)` sets the maximum number of digits (given by `n`) that are displayed for a number.
`fixed` allows inserting floating-point values in fixed format
`scientific` allows inserting floating-point values in scientific format
`left` left-justify
`right` right-justify
`dec` insert or extract integer values in decimal format
`oct` insert or extract values in octal (base 8) format
`hex` insert or extract integer values in hexadecimal (base 16) format

setw(n) with default format

This program diplays an integer and a float with `setw(n)`.

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
int i = 1299;
float f = 314.15926;
cout << "numbers:"
     << setw(10) << i
     << setw(15) << f << endl;
}
```

It shows only 3 digits because default format of the compiler is used

They are right justified

Output

Shows the column number

```
numbers:      1299         314.159
123456789012345678901234567890123
         1         2         3
```

setw(n) with left, right and setfill

This program diplays an integer, a float and a string with `setw(n)`.

```
#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

Shows the column number

```
University of Gaziantep
       University of Gaziantep
0000001453
123456789012345678901234567890123
         1         2         3
```

setw(n) with fixed and scientific formats

This program diplays a float with **fixed** with **scientific** formats

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
float f = 314.1593;
cout<<setw(15)<<fixed<<setprecision(4)<<f<<endl;
cout<<setw(15)<<fixed<<setprecision(2)<<f<<endl;
cout<<setw(15)<<scientific<< setprecision(3)<<f<<endl;
cout<<setw(15)<<scientific<< setprecision(5)<<f<<endl;
}
```

Output

```
       314.1593
         314.16
    3.142e+002
    3.14159e+002
123456789012345678901234567890123
         1         2         3
```

Shows the column number

Using cin to Format Input

cin has some member functions to manipulate input data.
The famous one is get() that is used to get a single character, including a white space, or a string for input.

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
char a, b, c;
cout << "Enter three letters: ";
cin.get(a).get(b).get(c);
cout << a << " "
     << b << " "
     << c << endl;
}
```

Output

```
Enter three letters: gaziantep
g a z
```

Using `getline()` to Format Input

Another useful function is `getline()` that is a standard library function that is used to read a string or a line from an input stream. Getline is a part of <stream> library.

```
#include <iostream>
#include<string>
using namespace std;
int main(){
string s;
cout << "Enter a line: ";
getline(cin,s);
cout<< "You entered:" << s <<endl;
}
```

Output

```
Enter a line: gaziantep universitesi
You entered:gaziantep universitesi
```

Input/Output with Files

Reading from and writing to files is performed using data streams.

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

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

These two class require the `fstream` to be included in the program.

There are several methods associated with `ifstream` and `ofstream`.

Some functions of these classes are given in the following table.

Input/Output with `fstream`

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

Using `ofstream`

```cpp
#include <iostream>
#include <fstream>
using namespace std;
int main ()
{
ofstream myFile("try.txt");

if (myFile.is_open())
{
  myFile << "We are in the try.txt file.\n";
  myFile << "And this is a sample text.\n";
  myFile.close();
}
else
  cout << "Unable to open file try.txt";
return 0;
}
```

The output is written to the file `try.txt`.

```
We are in the try.txt file.
And this is a sample text.
```

Using ifstream and ofstream

```
#include <iostream>
#include <fstream>
using namespace std;
int main ()
{
    double a,b;
    ifstream kutuk1("dosya1.txt");
    ofstream kutuk2("dosya2.txt");

    kutuk1>>a>>b;

    kutuk2<<"sum is: "<<a+b<<endl;
    kutuk2<<"mul is: "<<a*b<<endl;
    kutuk2<<"sub is: "<<a-b<<endl;
    kutuk2<<"div is: "<<a/b<<endl;
system("pause");
return 0;
}
```

The program first reads a and b from `dosya1.txt`.

Then it writes their summation, multiplication, subtruction and division to the output file of `dosya2.txt`

Using ofstream with format manipulators

```
#include <iostream>
#include <cmath>
#include <fstream>
#include <iomanip>
using namespace std;
int main ()
{
    ofstream an("angles.txt");
    an<<setw(15)<<"angle  sin()  cos()"<<endl;
    an<<setw(15)<<"-------------------"<<endl;
    for(int ang=0;ang<=90;ang+=5){
         an<<" "<<setw(2)<<setfill('0')<<ang
         <<setw(8)<<setfill(' ')<<fixed<<setprecision(3)
         <<sin(ang*M_PI/180.)<<setw(8)<<setfill(' ')
         <<fixed<<setprecision(3)<<cos(ang*M_PI/180.)<<endl;
    }
system("pause");
return 0;
}
```

Reading and writing in the same file

```
#include <iostream>
#include <fstream>
using namespace std;
int main ()
{
    int a,b;
    fstream i("inout.txt");
    i >> a >> b;
    i.close();
    ofstream o("inout.txt",ios::app);
    o << a*a << endl;
    o << b*b << endl;
system("pause");
return 0;
}
```

The program first reads a and b from `inout.txt`.

Then it writes their squares to the same file.