

6

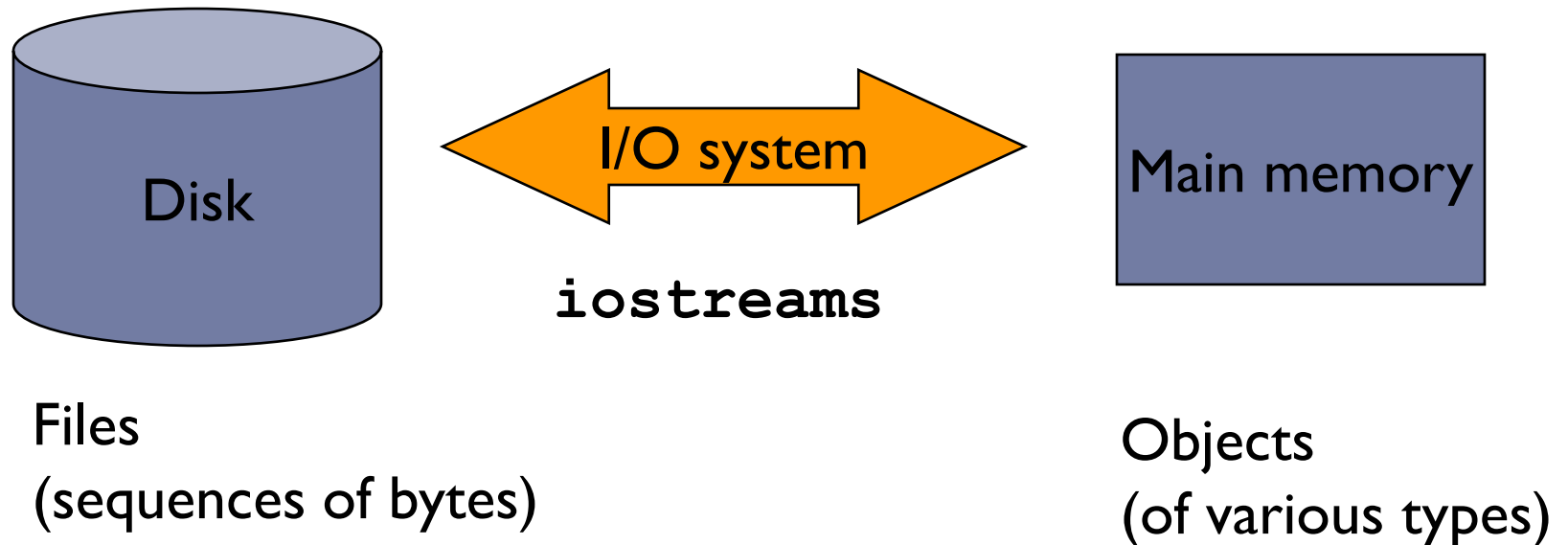
Files & Streams

Nur Nabilah Abu Mangshor

Objectives

- ▶ Learn about file operation in c++
- ▶ Discover basic operation in c++
- ▶ Learn how to open, read, write and close a file

General Model (Files)



Introduction

▶ File

- ▶ A file is a **collection of information**, usually **stored on a computer's disk**. Information can be saved to files and then later reused

▶ File Name

- ▶ All files are assigned a name that is used for **identification purposes** by the operating system and the user

Why we use file?

- ▶ **Convenient** way to deal large quantities of data.
- ▶ Store data **permanently** (until file is deleted).
- ▶ Avoid typing data into program multiple times.
- ▶ Share data between programs.
- ▶ We need to know:
 - ▶ how to "connect" file to program
 - ▶ how to tell the program to read data
 - ▶ how to tell the program to write data
 - ▶ error checking and handling EOF

Introduction (cont)

▶ Stream

- ▶ A transfer of information in the form of a sequence of bytes

▶ I/O Operations:

- ▶ **Input** :A stream that flows from an input device (i.e keyboard, disk drive, network connection) to main memory
- ▶ **Output** :A stream that flows from main memory to an output device (i.e screen, printer, disk drive)

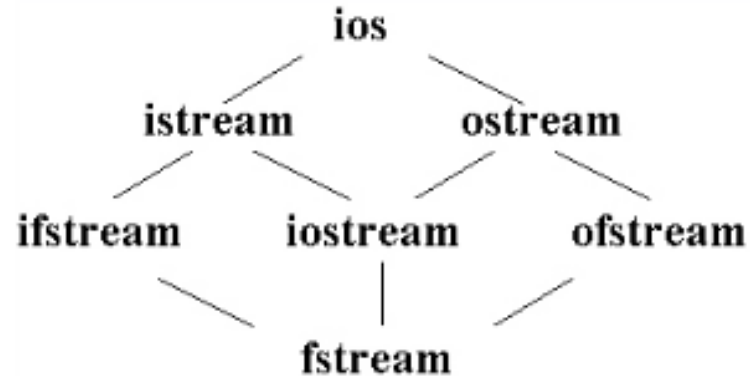
Using Input/Output Files

- ▶ **Stream** – A sequence of characters
 - ▶ interactive **<iostream>**
 - ▶ **cin** – input stream associated with keyboard
 - ▶ **cout** – output stream associated with display
 - ▶ file **<fstream>**
 - ▶ **ifstream** – defines new **input stream** (normally associated with a file)
 - ▶ **ofstream** – defines new **output stream** (normally associated with a file)

Stream I/O Library Header Files

- ▶ Note: There is no “.h” on standard header files :
<fstream>
- ▶ **<fstream>** : contains information for performing file I/O operations
- ▶ **<iostream>** : contains basic information required for all stream I/O operations
- ▶ **iomanip** : contains information useful for performing formatted I/O with parameterized stream manipulators
- ▶ **sstream** : contains information for performing in-memory I/O operations (into or from string memory)

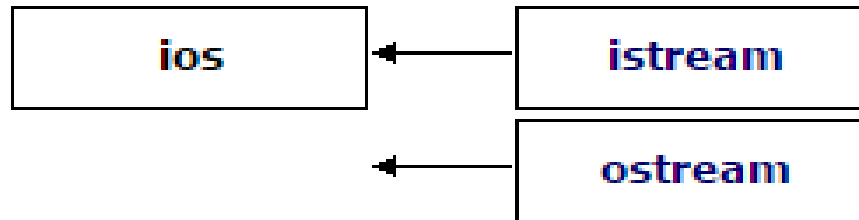
Classes for I/O Stream



- ▶ `ios` is the base class
- ▶ `iostream` and `ostream` inherit from `ios`
- ▶ `ifstream` inherits from `istream` (and `ios`)
- ▶ `ofstream` inherits from `ostream` (and `ios`)
- ▶ `iostream` inherits from `istream` and `ostream` (and `ios`)
- ▶ `fstream` inherits from `ifstream`, `iostream` and `ofstream`

Classes for I/O Stream (cont)

- ▶ **ios** is a base class if all stream classes



- ▶ **istream** is header providing the standard input and combined input/output stream classes
- ▶ **ostream** a header providing the standard output class

Operations on File

General File I/O Steps

- ▶ Declare a file name variable
- ▶ Associate the file name variable with the disk file name
- ▶ Open the file
- ▶ Use the file (write / read)
- ▶ Close the file

Opening a File

- ▶ A file must be opened before we can read or write on it
- ▶ Use the member function **open()**
- ▶ The syntax:

```
objectName.open(fileName);
```

- ▶ Where *objectName* is either object for **reading** a file or object for **writing** a file
- ▶ *fileName* is a string holds the name of the file

File I/O Example : Opening

```
#include <conio>
#include <fstream>

int main ()
{
    ofstream myfile;
    myfile.open ("example.txt");

    getch();
    return 0;
}
```

- The statement `myFile.open ("example.txt");` opens a text file name "example.txt"

Reading a File

- ▶ Once a file has successfully opened, you can read from it in the same way as you would read with `cin` or write to it in the same way as `cout`
- ▶ `ifstream` object is used to open file only for reading

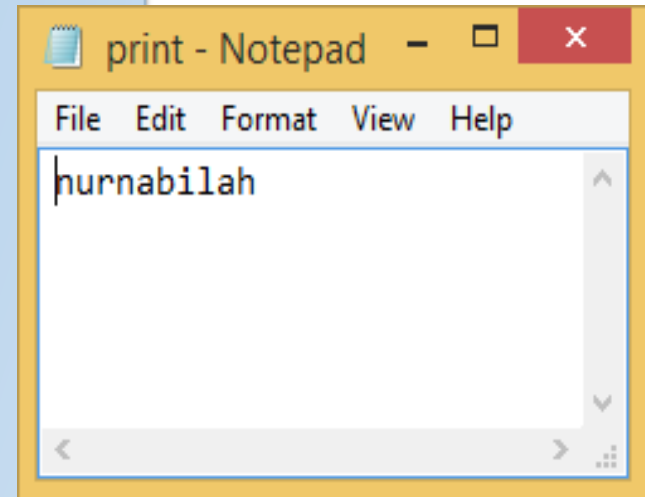
File I/O Example : Reading

```
#include <conio>
#include <fstream>

int main ()
{
    // open a file in read mode.
    ifstream infile;
    infile.open("print.txt");

    cout << "Reading from the file" <<
        endl;
    infile >> name;

    // write the data at the screen.
    cout << name << endl;  getch();
    return 0;
}
```



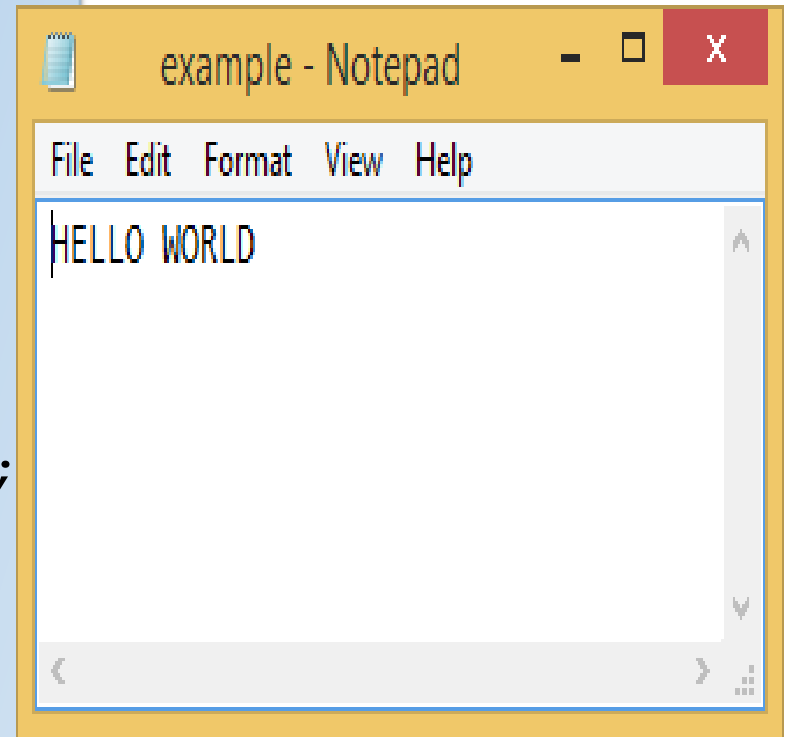
Writing a File

- ▶ A file must be opened before we can read or write on it
- ▶ write to it in the same way as `cout`
- ▶ `ofstream` or `fstream` object may be used to open a file for writing

File I/O Example : Writing

```
#include <iostream>
#include <conio>
#include <fstream>

int main ()
{
    ofstream myfile;
    myfile.open ("example.txt");
    myfile << "HELLO WORLD";
    myfile.close();
    getch();
    return 0;
}
```



Standard Input

- ▶ Standard Input is where things come from when you use `cin`. For example:

```
cin >> val;
```

- ▶ Standard input associated with the keyboard

Standard Output

- ▶ Standard output is exactly where things go when you use `cout`. For example :

```
cout << "Value = " << val << endl;
```

- ▶ Standard output associated with the screen

Closing a File

- ▶ A file shall be closed once we finished our input and output operation on it
- ▶ This member function `close()` takes flushes the associated buffers and closes the file

```
myfile.close();
```

- ▶ A statement above closes a file name myfile

Example

```
int main ()
{
    char name[10];
    // open a file in write
mode.
    ofstream outfile;
    outfile.open("print.txt");

    cout << "Writing to the
file" << endl;
    cout << "Enter your name: ";
    cin.getline(name, 100);

    // write inputted data into
the file.
    outfile << name << endl;
```

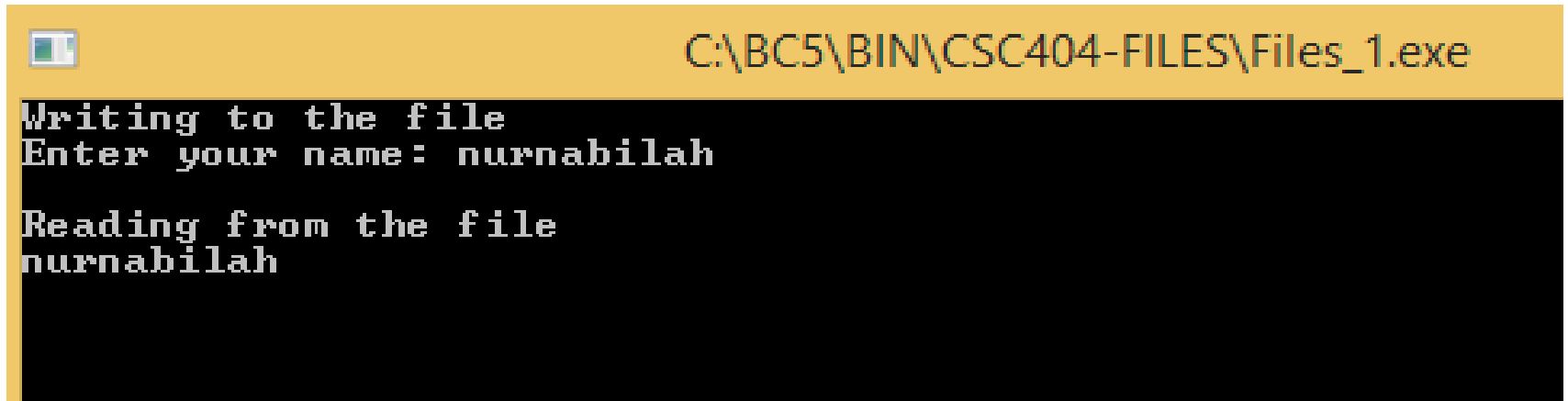
```
// open a file in read mode.
ifstream infile;
infile.open("print.txt");

    cout << "\nReading from the
file" << endl;
    infile >> name;

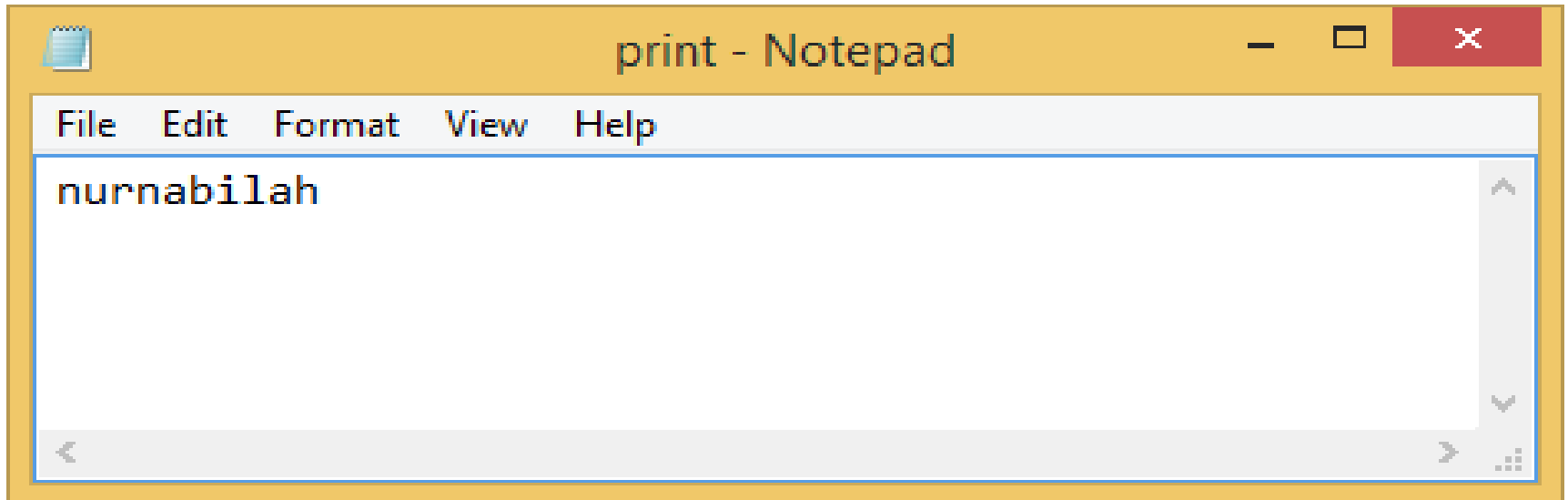
    // write the data at the
screen.
    cout << name << endl;

    getch();
    return 0;
}
```

Example (cont)



```
C:\BC5\BIN\CSC404-FILES\Files_1.exe  
Writing to the file  
Enter your name: nurnabilah  
Reading from the file  
nurnabilah
```



```
print - Notepad  
File Edit Format View Help  
nurnabilah
```

Summary of Input File-Related Functions

```
#include <fstream>
ifstream fsIn;
```

- ▶ **fsIn.open (const char[] fname)**
 - ▶ Connects stream `fsIn` to the external file `fname`
- ▶ **fsIn.get(char& c)**
 - ▶ Extracts next character from the input stream `fsIn` and places it in the character variable `c`
- ▶ **fsIn.eof()**
 - ▶ Tests for the end-of-file condition
- ▶ **fsIn.close()**
 - ▶ Disconnect the stream and the associated file
- ▶ **fsIn >> c**
 - ▶ For input value. Behaves just like `cin>>`

Summary of Output File-Related Functions (cont)

```
#include <fstream>
ofstream fsOut;
```

- ▶ **fsOut.open (const char[] fname)**
 - ▶ Connects stream `fsOut` to the external file `fname`
- ▶ **fsOut.get(char& c)**
 - ▶ Inserts character `c` from the output stream `fsOut`
- ▶ **fsOut.eof()**
 - ▶ Tests for the end-of-file condition
- ▶ **fsOut.close()**
 - ▶ Disconnect the stream and the associated file
- ▶ **fsOut << c**
 - ▶ For display output value. Behaves just like `cout>>`

Reference

- ▶ D. S. Malik. 2009. C++ Programming From Problem Analysis to Program Design.