CAP444
OBJECT ORIENTED PROGRAMMING USING C++

Unit4

Created By:
Kumar Vishal
(SCA), LPU

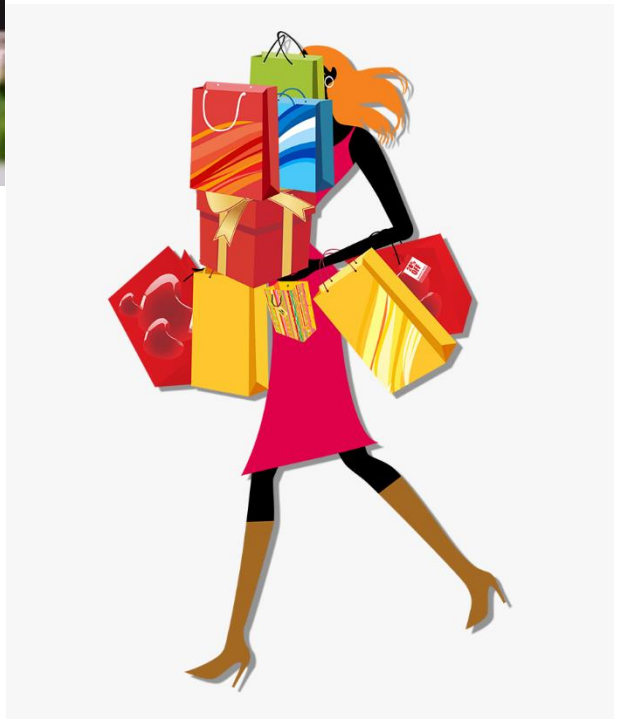
Unit-4

Working with files and streams :

- c++ streams, c++ stream classes,
- classes for file stream operations,
- opening & closing files,
- detection of end of file,
- more about open(): file modes,
- file pointer & manipulator,
- sequential input & output operation,
- updating a file: random access,
- command line arguments

Real Life Example

Real Life Example



Real Life Example

Real Life Example

Why File is required?

A. To store data or information

B. To use data in our program

C. Both above

D. None

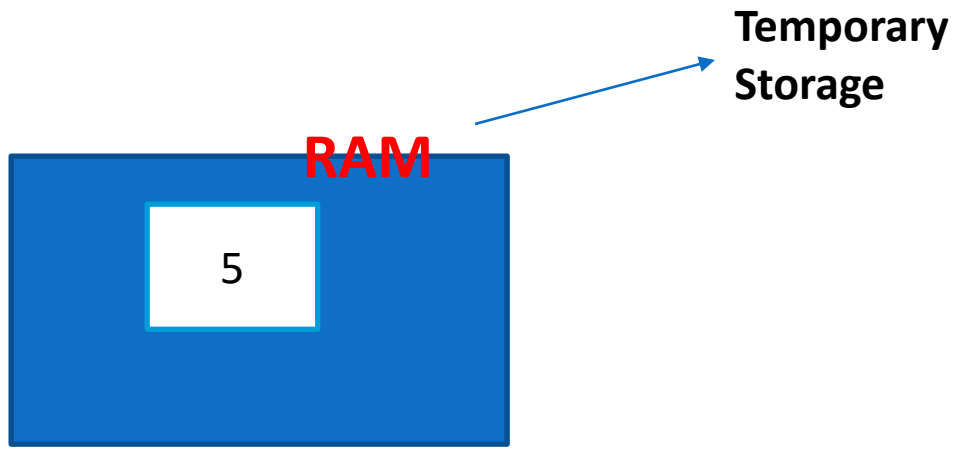
Why File is required?

- To store data or information

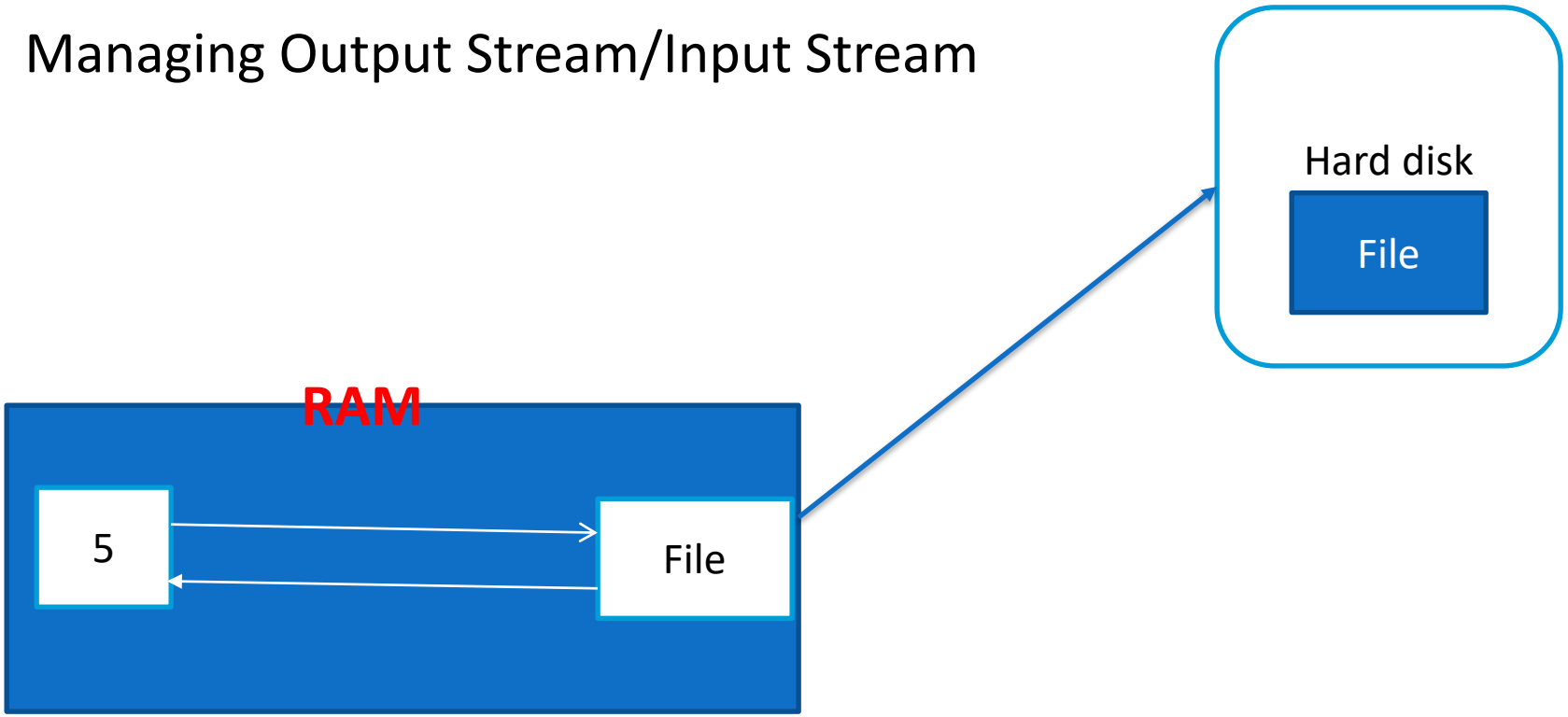
In Programming how we are doing?

- Using variable → through RAM
- Using File → through hard disk

```
#include <iostream>
using namespace std;
int main()
{
    int num;
 cout<<"Enter
number"<<endl;
    cin>>num;
    return 0;
}
```

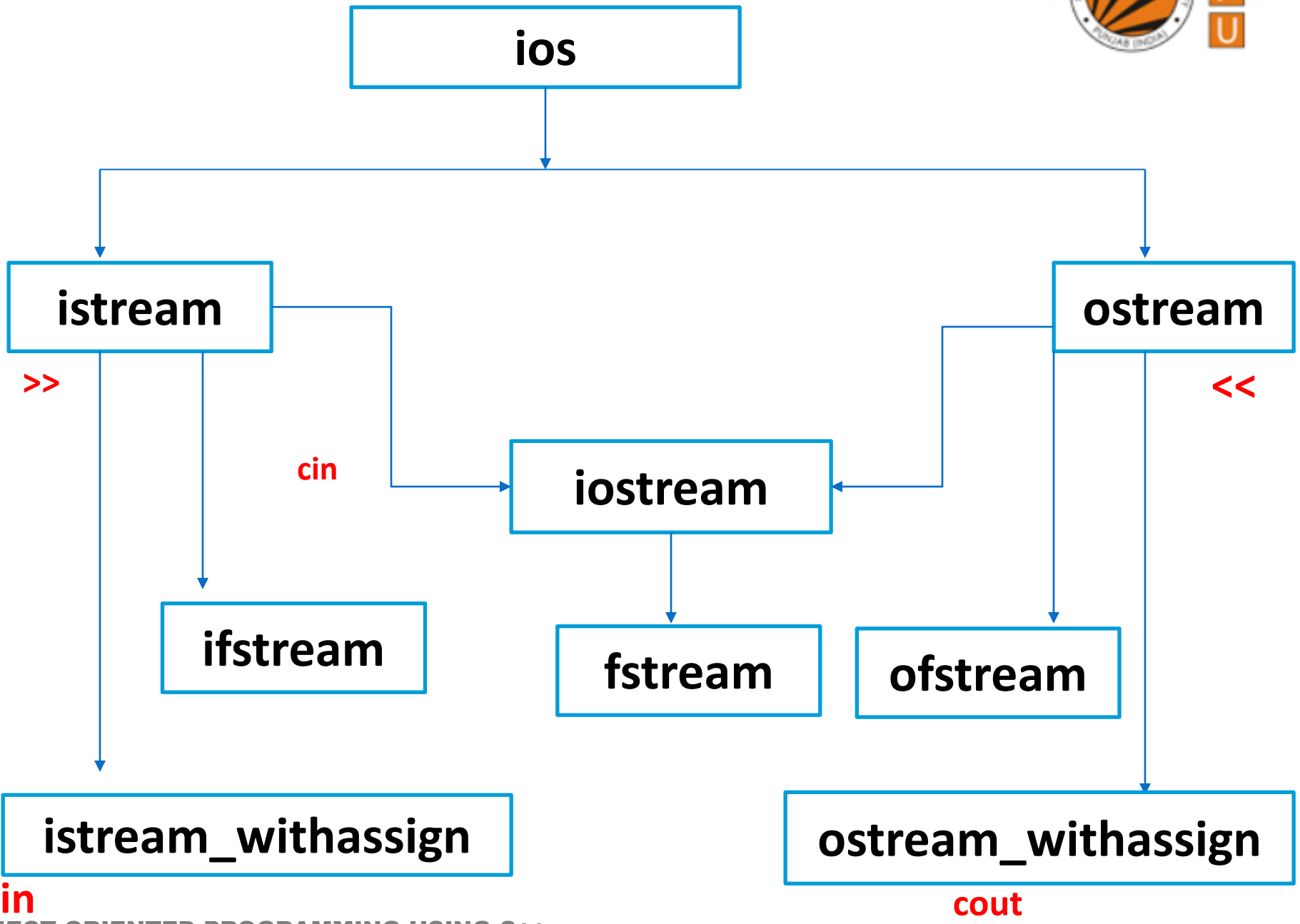


Managing Output Stream/Input Stream



We have predefine classes

Classes for file stream



In C++, files are mainly deal with three classes fstream, ifstream, ofstream.

ofstream: This Stream class indicates the output file stream and is applied to create files for writing information to files

ifstream: This Stream class indicates the input file stream and is applied for reading information from files

fstream: This Stream class can be used for both read and write from/to files.

C++ provides us with the following operations in File Handling:

- Creating a file: open()
- Reading data: read()
- Writing new data: write()
- Closing a file: close()

Which of the following is used to create a stream that performs both input and output operations?

a) ofstream

b) ifstream

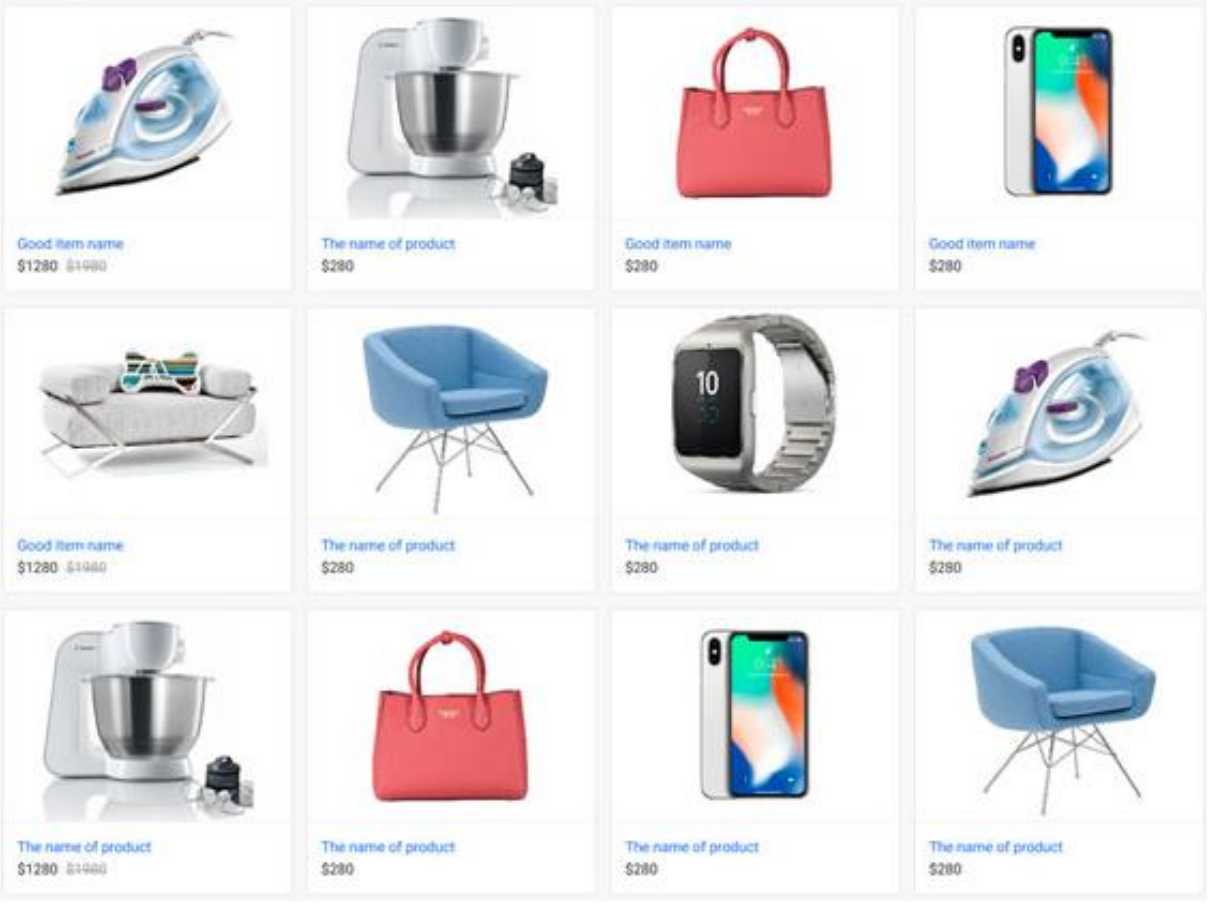
c) iostream

d) fstream

Real Life Example

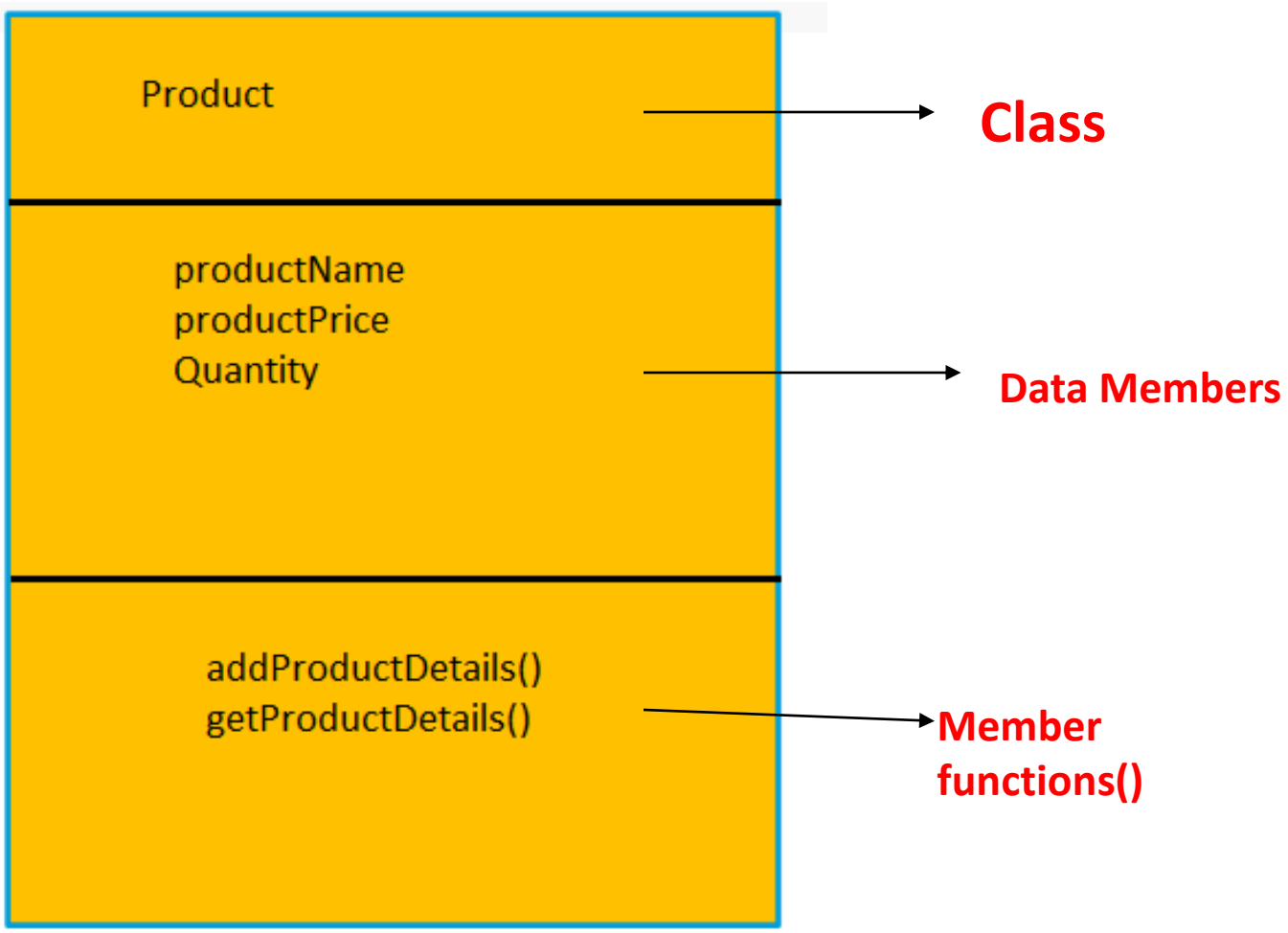
Keeping record of products:

- using file handling mechanism



Steps:

1. Create a product class



Steps:

2. Create a file and fill all product records.

3. Update your file, fill more records into file

3. Display output to the user screen with product details.

Opening Files

- open() In case of creating new file:
  - Using ofstream class

  Syntax:

  ofstream fout;

  Fout.open("filename")
- open() In case of reading file:
  - Using ifstream class

  Syntax:

  ifstream fin;

  fin.open("filename")

Closing Files

- close() In case of creating new file:
  - Using ofstream class

  Syntax:

  ofstream fout;

  Fout.close()
- close() In case of reading file:
  - Using ifstream class

  Syntax:

  ifstream fin;

  fin.close()

Reading and Writing into Files

- Writing File: used ofstream class:

  Syntax:

  ofstream fout;

  fout.open(“filename”);

  fout<<“data”;

- Reading File: used ifstream class:

  Syntax:

  ifstream fin;

  Ifstream.open(“filename”);

  Fin>>“data”; or using get() or getline()

Reading Files: using get() function

The get() function is member of ifstream class. It is used to read character form the file.

```
while(!fin.eof())
    {
       fin.get(ch);
       cout<<ch;
    }
```

will read all the characters one by one upto EOF(end-of-file) reached.

Detecting End-of-File

- While reading data from a file, if the file contains multiple rows, it is necessary to detect the end of file.
- This can be done using the eof() function of ios class.
- It returns 0 when there is data to be read and a non-zero value if there is no data.

Syntax:

```
ifstream fin;
char ch;
Ifstream.open("filename");
while(!fin.eof())
{
   fin.get(ch);
   cout<<ch;
}
```

Reading Files: using getline()

How to process a file line by line in C++?

In C++, you may open a input stream on the file and use the getline() function from the <string> to read content line by line into a string and process them.

```
ifstream file("input.txt");
string str;
while (getline(file, str)) {
  // process string ...
}
```

Check file is existing or not:

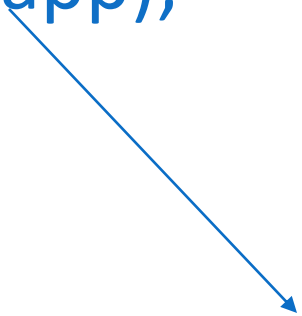
```
ifstream  fin;
fin.open("abc.txt");
If(fin)
{
cout<<"File is existing"<<endl;
}
else{
cout<<"File is not existing"<<endl;

}
```

To append file content

- ios::app

```
ofstream fout;
fout.open("filename",ios::app);
```



File Mode

Which is used for append mode in file?

a) ios::in

b) ios::apt

c) ios::app

d) ios::apd

What is the return type open() method?

a) int

b) char

c) bool

d) float

File Modes

ios::in	Open for input operations.
ios::out	Open for output operations.
ios::binary	Open in binary mode.
ios::ate	Set the initial position at the end of the file. If this flag is not set, the initial position is the beginning of the file.
ios::app	All output operations are performed at the end of the file, appending the content to the current content of the file.

class	default mode parameter
ofstream	ios::out
ifstream	ios::in
fstream	ios::in | ios::out

Which of the following is the default mode of the opening using the ofstream class?

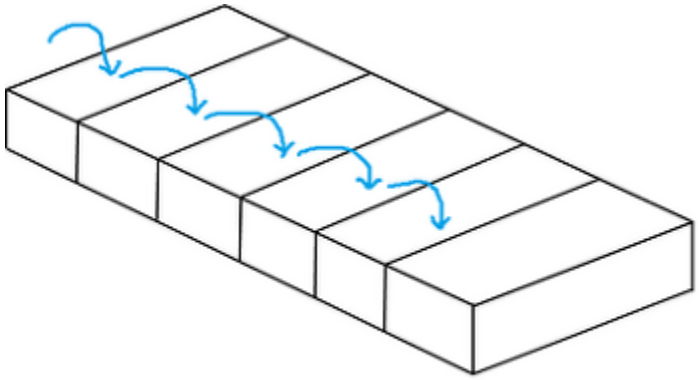
a) ios::in

b) ios::out

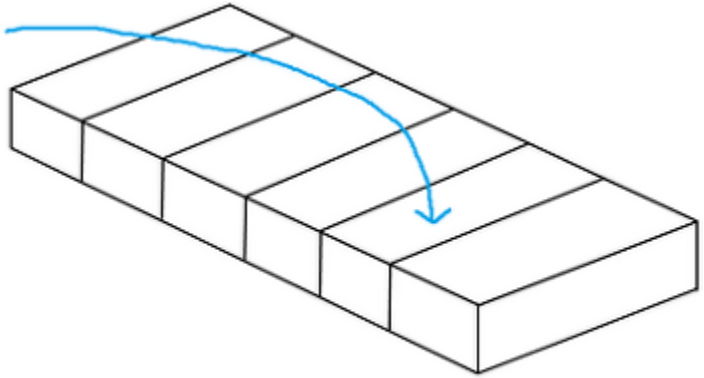
c) ios::app

d) ios::trunc

Real Life Example



sequential access



random access

Sequential and Random I/O

- C++ allows data to be read or written from a file in sequential or random fashion.
- Reading data character by character or record by record is called sequential access.
- Reading data in any order is known as random access.
- The fstream class provides functions like get(), read() for reading data and put(), write() for writing data to a file.

Sequential and Random I/O

- The functions *get()* and *put()* are character-oriented functions

Syntax:

- get(char)
- put(char)

A program demonstrates get() and put() functions

- For example, if you have to modify a value in record no 21, then using random access techniques, you can place the file pointer at the beginning of record 21 and then straight-way process the record. If sequential access is used, then you'll have to unnecessarily go through first twenty records in order to reach at record 21.
- In C++, random access techniques is achieved by manipulating seekg(), seekp(), tellg() and tellp() functions.

File Pointers and Manipulation

Every file will contain two pointers: a read pointer or also known as a get pointer and a write pointer also known as a put pointer. The read pointer or a get pointer is used to read data and the write pointer or put pointer is used to write data to a file. These pointers can be manipulated using the functions from stream classes. Those functions are:

seekg() and seekp()

seekg() and seekp()

- seekg() function allow us to move the Input pointer to specified location for reading purpose within the file. The basic syntax for seekg() function is :

fileObject.seekg(long_num, origin);

- fileObject: pointer to fie
- long_num: no. of bytes in file we want to skip
- origin: where to begin

origin:

Origin	Syntax	Explanation
ios::beg	`fileObject.seekg(0, ios::beg);`	Go to Start: No matter how far into a file we have read, by using this Syntax, the file-pointer will back to the beginning of the file.
ios::cur	`fileObject.seekg(0, ios::cur);`	Stay at Current Position: Using this syntax, the file-pointer will show its current position.
ios::end	`fileObject.seekg(0, ios::end);`	Go to End of the file: using this syntax, the file-pointer will point to end of the file.

File Pointers and Manipulation

- The seekg() and tellg() functions are for input streams (ifstream) and
- seekp() and tellp() functions are for output streams (ofstream)

- ***Example to use tellg() and seekg() functions***

seekg() and seekp()

seekg() function allow us to move the Input pointer to specified location for reading purpose within the file. The basic syntax for seekg() function is :

fileObject.seekg(long_num, origin);

- fileObject: pointer to fie
- long_num: no. of bytes in file we want to skip
- origin: where to begin

origin:

ios::beg start of the file
ios::cur current position of the pointer
ios::end end of the file

Ex:

```
fin.seekg(0, ios::beg);
```

Setting the EOF flag off, to allow the access of file again for reading:-

Ifstream fin;

fin.clear();

seekg()	moves get pointer(input) to a specified location
seekp()	moves put pointer (output) to a specified location
tellg()	gives the current position of the get pointer
tellp()	gives the current position of the put pointer

Which function is used to reposition the file pointer?

a) moveg()

b) seekg()

c) changep()

d) go_p()

Which of the following is used to move the file pointer to start of a file?

a) ios::beg

b) ios::start

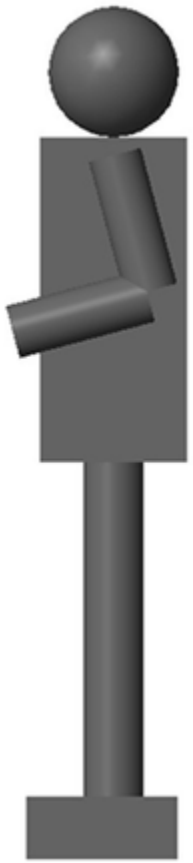
c) ios::cur

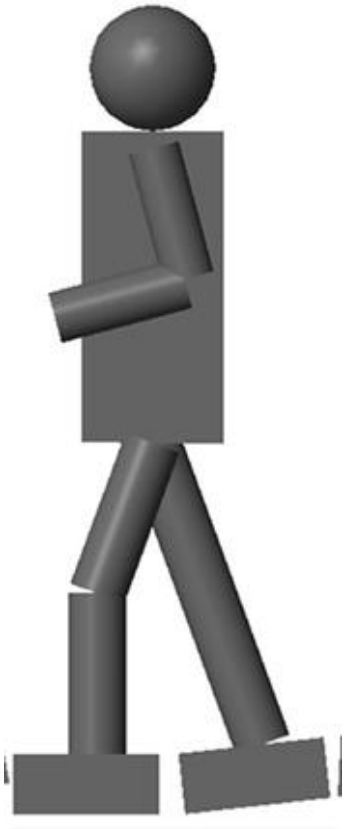
d) ios::first

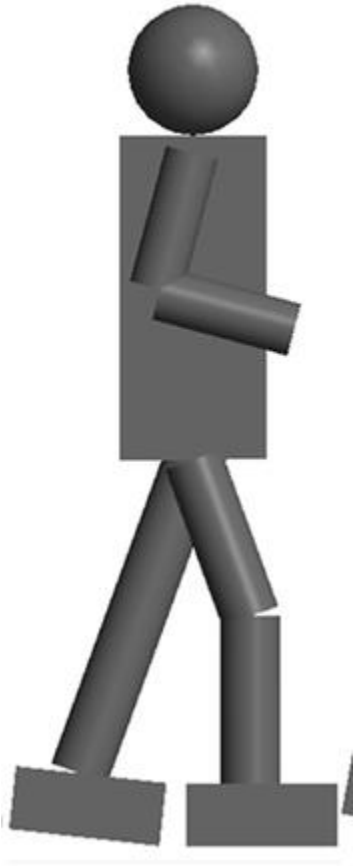
File handling Examples:

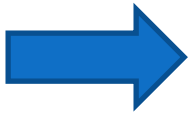
- cplusplus/Program for File handling.pdf at master · vishalamc/cplusplus (github.com)
- cplusplus/RobotProgram.pdf at master · vishalamc/cplusplus (github.com)
- https://github.com/vishalamc/cplusplus/blob/master/fileManipulationEx

A robot moves in a plane starting from the original point (0,0). The robot can move toward LEFT and RIGHT with one step. The trace of robot movement and calculate the total moves of Robot.









With the help of File Pointers and Manipulation functions we can solve this type of problem statement.

Command-line arguments

- Command-line arguments are given after the name of the program in command-line shell of Operating Systems.
- To pass command line arguments, we use main() with two arguments :
  - first argument is the total number of command line arguments and
  - second is list of command-line arguments.

Syntax:

```
int main(int argc, char *argv[ ])
{
return 0;
}
```

first parameter *argc* holds the count of command-line arguments and the second parameter, an array of character pointers holds the list of command-line arguments.

first element in the array, i.e., argv[0] holds the filename. First command-line parameter will be available in argv[1], second parameter in argv[2] and so on.

To run in command line:

D:\>g++ abc.cpp -o obj1.exe

D:\>obj1.exe

Any Query?

Unit-4 End