CAP444
OBJECT ORIENTED PROGRAMMING USING C++

Created By:
Kumar Vishal
(SCA), LPU

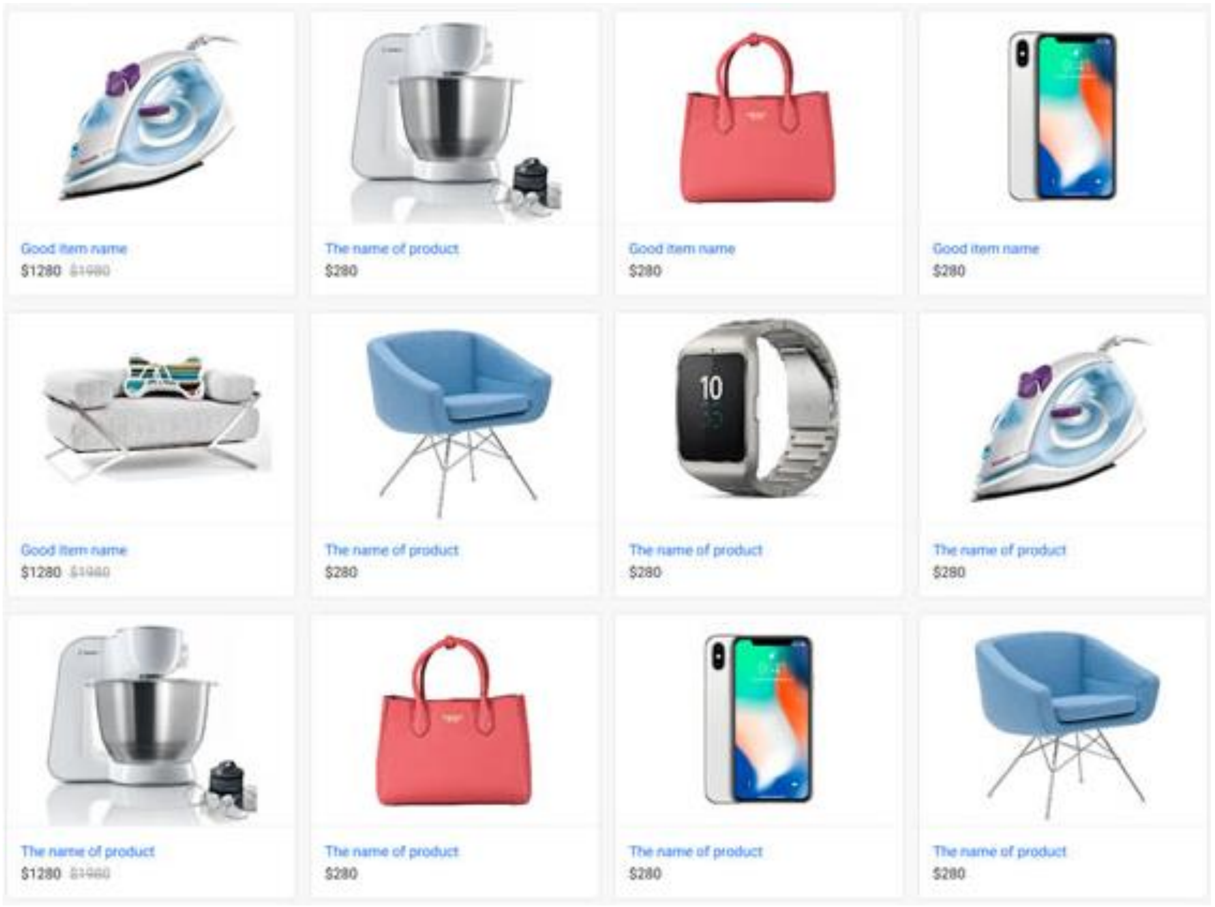
Unit-4

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

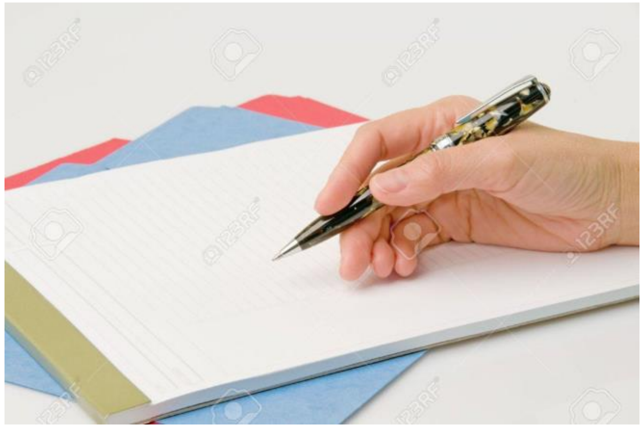
Keeping record of products:

➢ using file handling mechanism

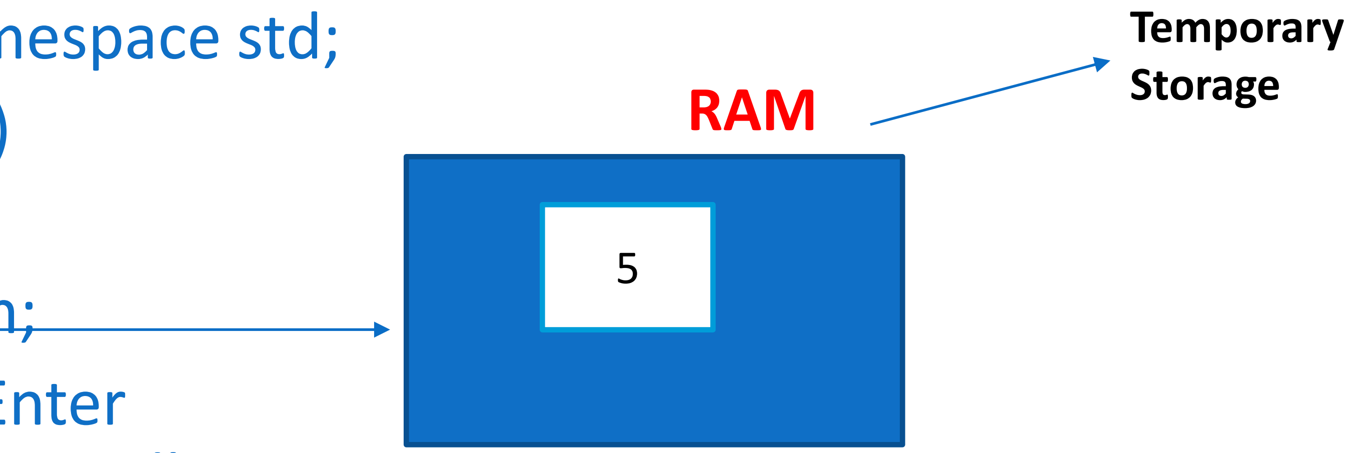


E-COMMERCE

Real Life Example

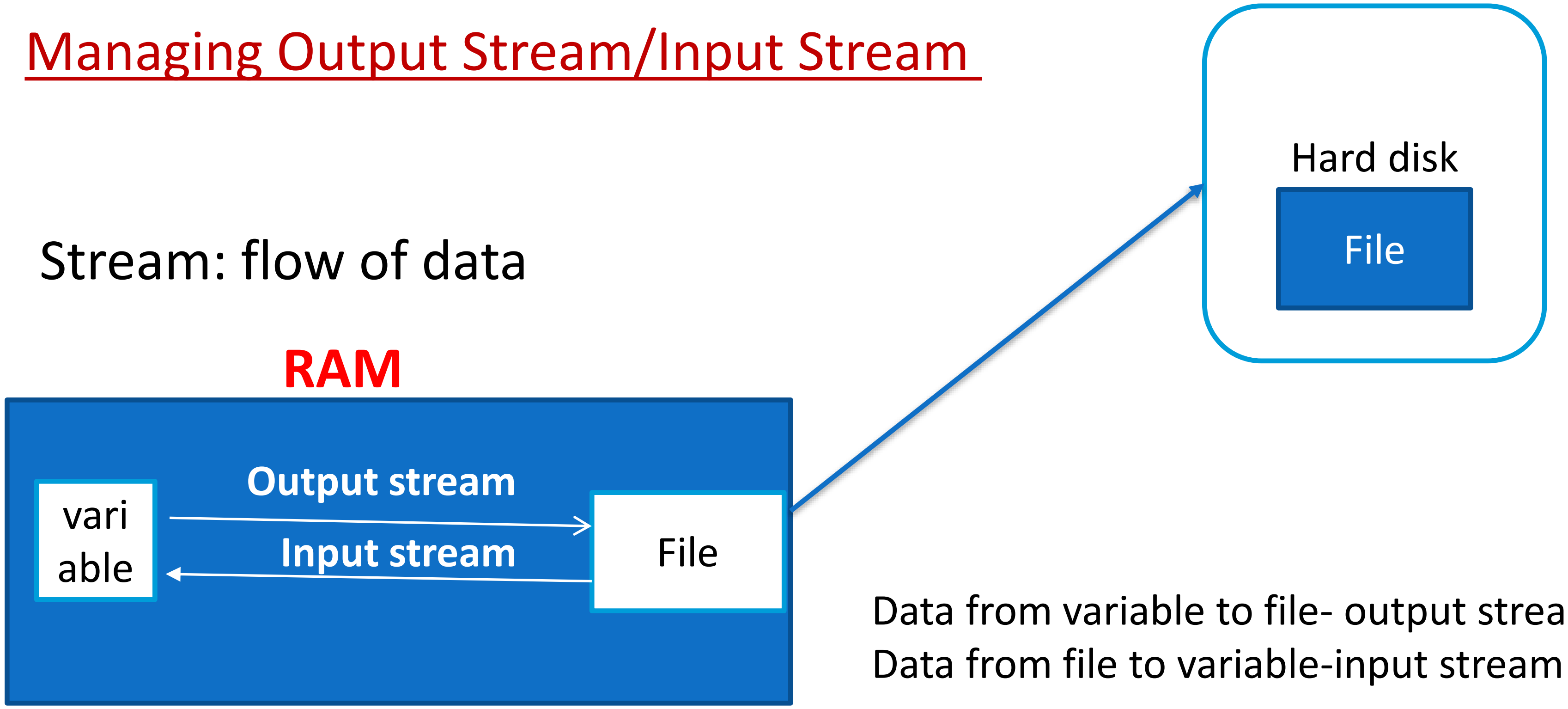


```
#include <iostream>
using namespace std;
int main()
{
    int num;
 cout<<"Enter number"<<endl;
    cin>>num;
    return 0;
}
```



Managing Output Stream/Input Stream

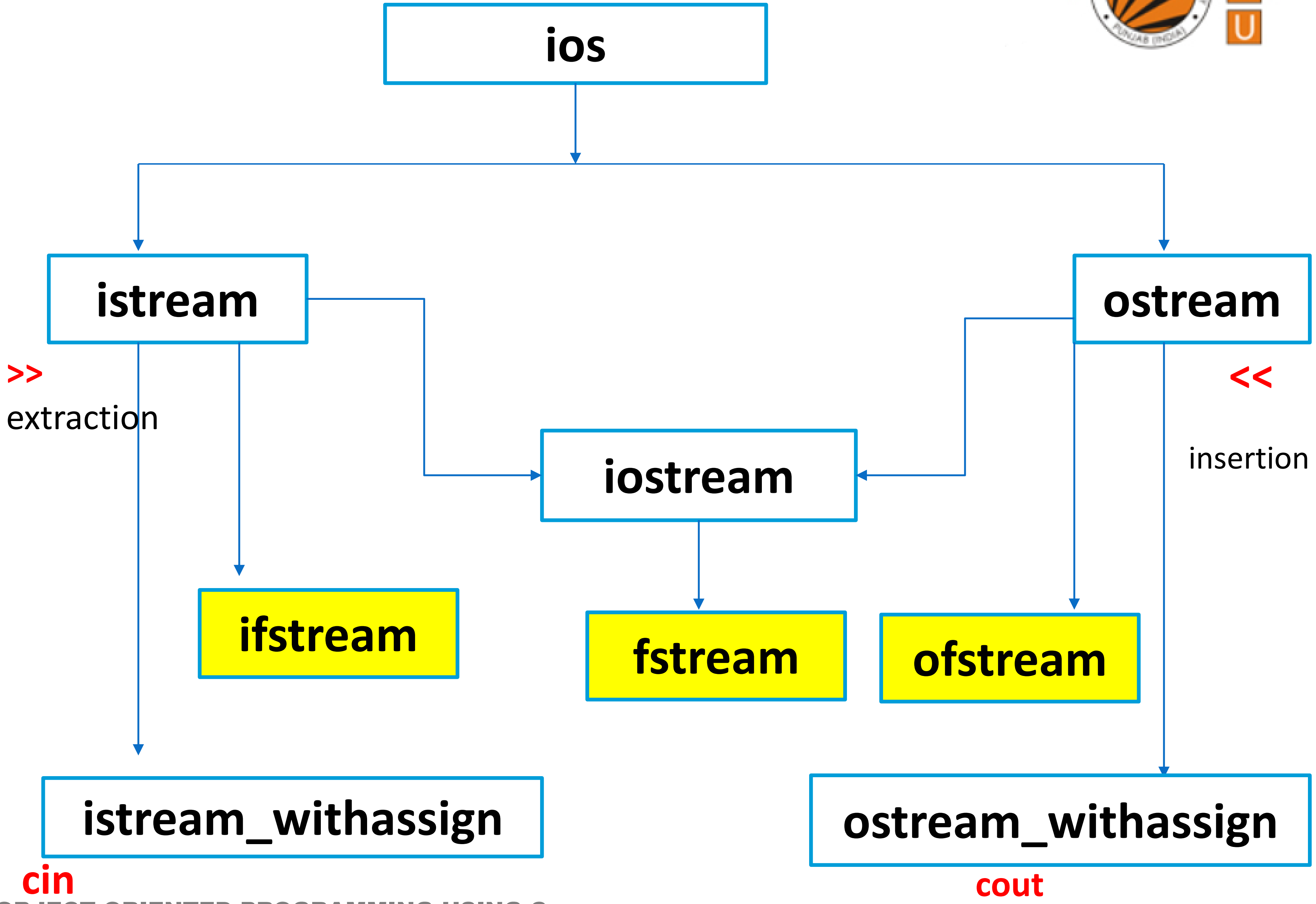
Stream: flow of data



Data from variable to file- output stream
Data from file to variable-input stream

We have predefine classes to manage all these things

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 not a component of file system

A. Access method

B. Auxiliary storage management

C. Free integrity mechanism

D. None of the above

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”);

  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 up to 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 fin;
fin.open("d://demo//file1.txt");
    string str;
    while(getline(in,str))
    {
       cout<<str<<endl;
    }
```

Which of the following methods can be used to open a file in file handling?

a) Using Open ( )

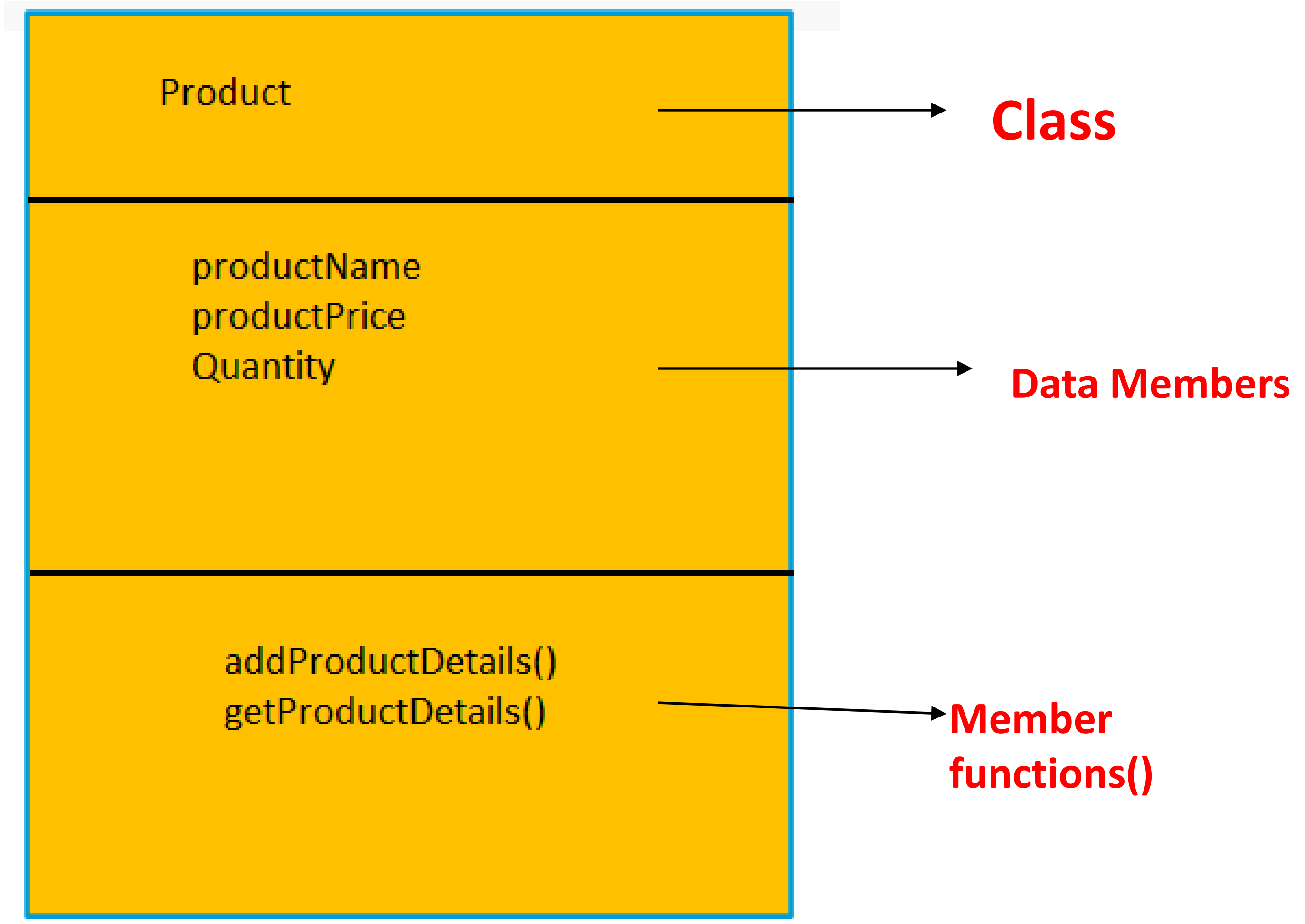
b) Constructor method

c) Destructor method

d) Both A and B

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.

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;

}
```

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

To append file content

ios::app

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

File Mode

Any Query?

Unit-4 End