C++

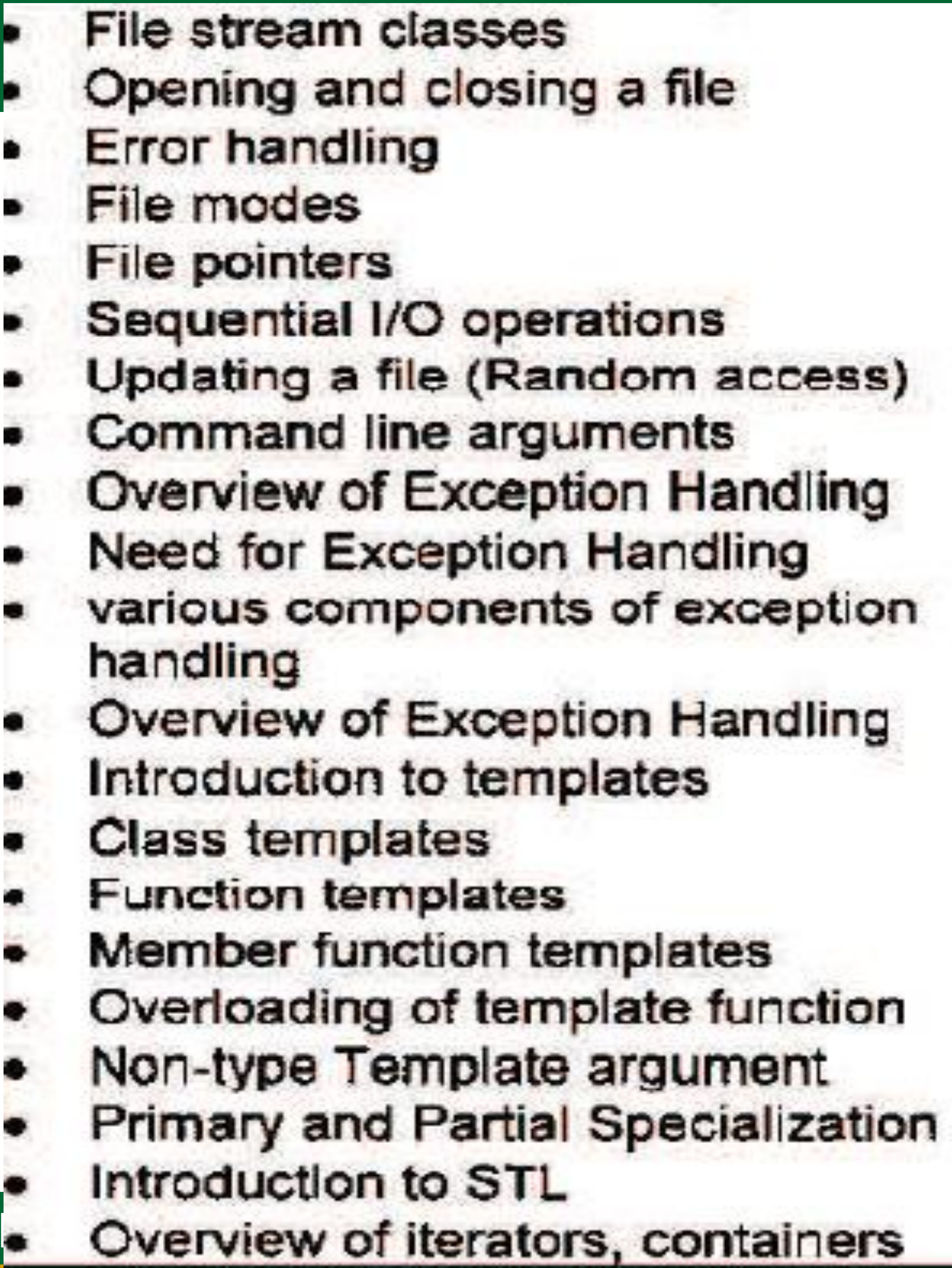
Ch.5

Working with files,

Exception Handling,

Template

Ch.5 Syllabus

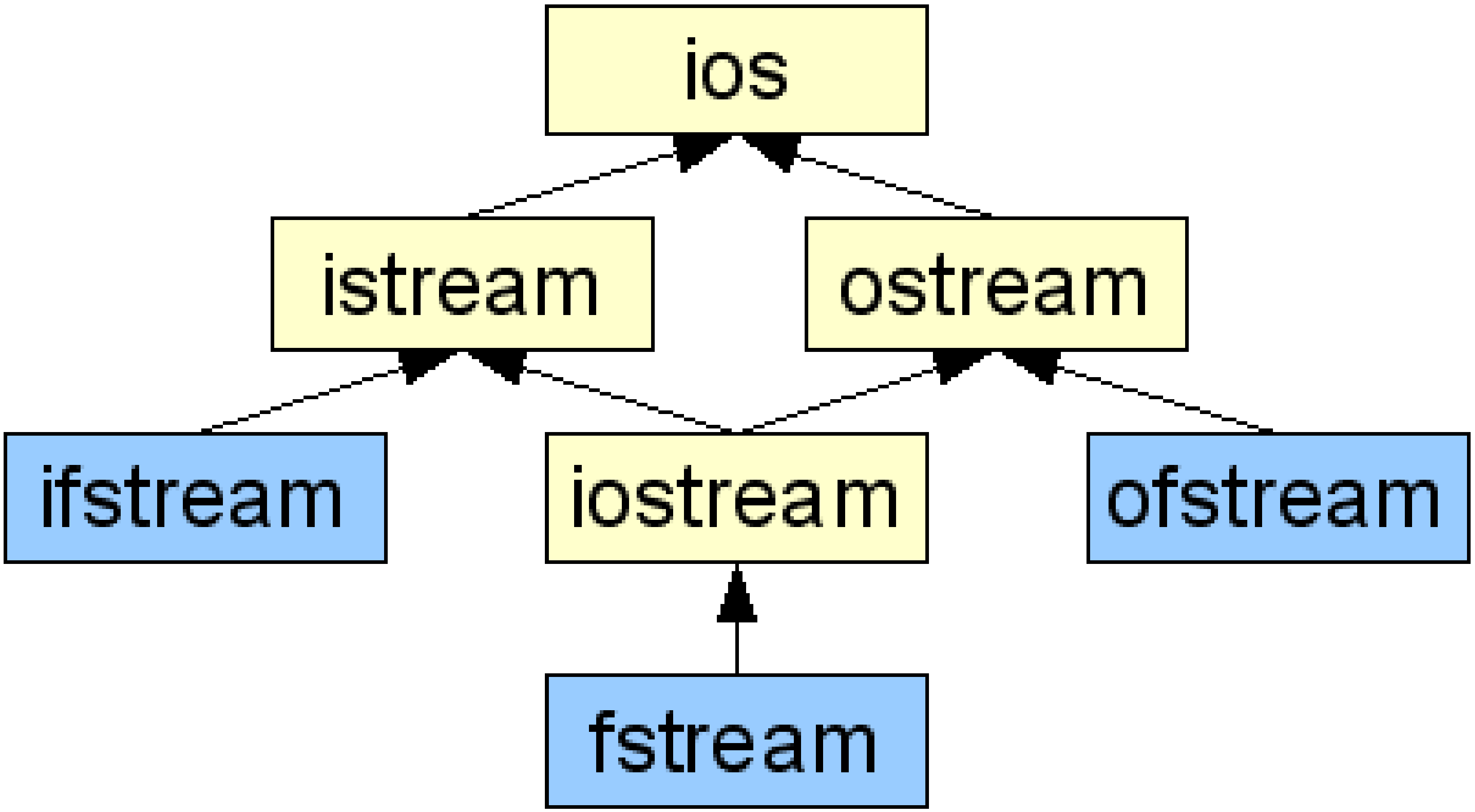


- File stream classes
- Opening and closing a file
- Error handling
- File modes
- File pointers
- Sequential I/O operations
- Updating a file (Random access)
- Command line arguments
- Overview of Exception Handling
- Need for Exception Handling
- various components of exception handling
- Overview of Exception Handling
- Introduction to templates
- Class templates
- Function templates
- Member function templates
- Overloading of template function
- Non-type Template argument
- Primary and Partial Specialization
- Introduction to STL
- Overview of iterators, containers

Introduction :

- As a computer programmer, we have sufficient knowledge of data storage.
- We can store data in computer for future use. We can store and retrieve data as our requirement.
- In C++ files are the best way to read and write data. In C++ we have many useful functionalities to handle files.

File Stream Classes :



File Stream Classes :

- IOS
  - Input, Output, Stream the main file related to C++ functionality…

Class	Description
Fstreambase	Base class for fstream, ifstream and ofstream classes. It contains basic operations common for all file stream classes. It defines open() and close() functions.

File Stream Classes :

Class	Description
Ifstream	It offers operations for file input. Defines open() function with input mode and derives functions from istream such as get(), read(), getline() etc.
Ofstream	It offers operations for output. Defines open() function for output mode and derives functions from istream such as put(), write() etc.

File Stream Classes :

Class	Description
Fstream	It offers both input and output operations. It derives properties of istream and ostrem classes.
Filebuf	It derives streambuf class and uses buffer for fast input and output operations.

Opening and Closing Files :

- Opening a File.
  1. Using the open( ) function.
  2. Using the constructor.
- Using the open( ) function

Syntax : fileObject.open(“***File Name***”);

  - To open a file using object we can use open( ) function.
  - Example :

```
ofstream file;
file.open(“MyFirstFile.txt”);
```

Opening and Closing Files :

- Closing a file :
  - We have to close the object before open another file object.

  Syntax : *fileObj.close*( );

  Example :

```
   ofstream file;
   file.open("MyFirstFile.txt");
   file.close();
```

Opening multiple files :

- We can open more then one data file in a single file.
- But we have to close opened file fist.
  - Example :

    ```
    ofstream File;
    File.open("File.txt");
    …
    File.close();
    ```

    Then open another file…

To write in a file…

- To write in a file, we can use **put( )** or **write( )** function.
- Use **put( )** function to write one character a time and use **write( )** function to write multiple character at once.
- We can also use the insertion (<<) **operators to write content** and extraction (>>) operators to read contents from file. But extraction operator reads data till the white space only. So the data after the white space will not be read.

Def. Wap to print A to Z in **abcd.txt** file. Using put( ) function.

```
void main()
{ ofstream file; //To write data in file...
  file.open("monarch.txt");
  char c='A';
  for(int x=1;x<=26;x++)
  {   file.put(c);
      c++;
  }
  file.close();
}
```

Def. Wap to print Z to A in **dcba.txt** file. Using put( ) function.

Def. Wap to print Half pyramid of three line in **pyramid.txt** file. Using put( ) function.

```
A
BC
DEF
```

Def. Wap to store your address in **myAddress.txt** file using **<<** operator

```
#include<iostream.h>
#include<fstream.h>
void main()
{ ofstream file;  //To write data in file...
  file.open("monarch.txt");
  file<<"MONARCH Sankul"<<endl;
  file<<"Sanghavi Street, Lathi"<<endl;
  file<<"Ta. Lathi, Di. Amreli"<<endl;
  file.close();
}
```

Read Data From a FILE

- To read data from a file we have to use seekg( ) function to navigate the pointer.
  - Syntax : ***FileObj.***seekg***(****int position****)***
  - *Example :*

    File.seekg(0); // move to first character

Def. WAP to read data from **abcd.txt** file.

```
void main()
{ ifstream file; //To READ data in file...
  file.open("abcd.txt");
  file.seekg(0);
  char c;
  while(file)
  {  file.get(c);
     cout<<c;      }
  file.close();
}
```

Def. Wap to Enter your name and address in **address.txt** file. Then print the data using **get()** function.

Def. Wap to Enter student result for BCA Sem3 (4 subject + 2 practical) and store it in RESULT.txt file. Then print data using get().

Opening file using constructor :

- We can open data text file using constructor also.
- Specify the file name as an argument to the constructor. The file is opened as the object is created and no need to call a function.
- But the limitation of opening a file using constructor is you cannot open multiple files using one object.

Opening file using constructor :

- Example :

```
ifstream File("MyData.txt");

ofstream File("MyData.txt");
```

- When you create an object to open a file, the file with the specified name is created. If the file with same name exists in the same folder, the file is overwritten.

Def. WAP to enter student result and print it on disk...

```
#include<iostream.h>
#include<fstream.h>
void main()
{ int seat, s[6],x;
  ofstream f1("result.txt");
  cout<<"Enter SeatNo :";
  cin>>seat;
  f1<<seat<<endl;
```

```
for(x=0;x<6;x++)
{  cout<<"Marks Sub"<<x+1<<": ";
   cin>>s[x];
   f1<<s[x]<<endl;
}
f1.close();
//Print Entered DATA
cout<<"Print Student Data"<<endl;
ifstream f2("result.txt");
f2>>seat;
cout<<"Seat No : "<<seat<<endl;
```

```
  for(x=0;x<6;x++)
  {  f2>>s[x];
     cout<<"Sub"<<x+1<<s[x]<<endl;
  }
  f2.close();
  getch();
}
```

End of File :

- While reading a file we need to detect the end of file to prevent reading after the last byte of file. We can detect the end of file by two ways…
  - Using while loop
  - Using eof( ) function

End of File :

- Using while loop
  - We can use object of input file stream in while loop.
  - Example :

```
ifstream File;
while (File)
{ //Work until end of file position
}
```

  - Here, the object File returns 0 if any error occurs during the file reading such as end of file otherwise it returns non-zero value.

End of File :

- Using eof() function :
  - We can use eof() function to detect **end of file** which is a member function of ios class. It returns a non-zero value if the file pointer is reached at the end of file during read operation otherwise it returns zero.

```
ifstream File;
if(File.eof()!=0)
{ //Work until end of file position
}
```

File Modes

- C++ supports various file opening modes that can be used with **open()** function. The file mode can be passed as the second argument in the open() function. Following is the syntax of open() function.
- System :

  FileStreamObject.open(“File”,Mode);
- So far we have passed only one argument in the open() function that is file name. The second argument specifies the file mode which can be one of from table.

File Modes

File Mode	Meaning
ios::app	Append. Opens file in appending mode. Allows adding data at the end of file only.
ios::ate	At the End. Puts the cursor at the end of the file on opening. Allows adding data anywhere in the file.
ios::binary	Used to open the binary file.
ios::in	Opens the file in read-only mode.

File Modes

File Mode	Meaning
ios::out	Opens the file in write-only mode.
ios::nocreate	Does not create the file if it does not exist. In that case file opening fails.
ios::noreplace	Does not replace the file if it already exists. In that case file opening fails.
ios::trunc	Clears the file contents if it already exists.