# C++ File I/O :Stream I/O class

1

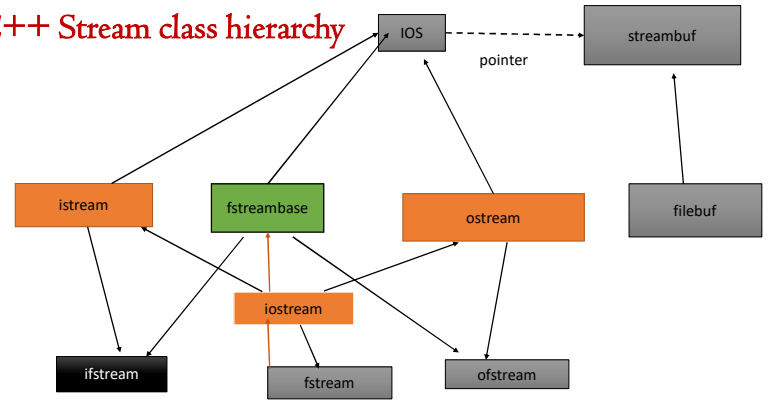
## Learning Objectives

- File as a data type/Data structure
- C++ I/O streams.
- Reading and writing sequential files.
- Reading and writing random access files.

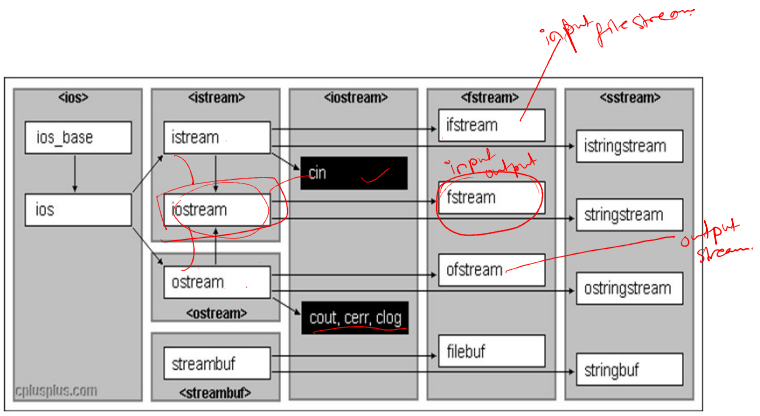


2

## C++ Stream class hierarchy




3




4

## Iostream Library Header Files

- **iostream** library:
  - **<iostream.h>:** Contains **cin, cout, cerr,** and **clog** objects
  - **<iomanip.h>:** Contains *parameterized stream manipulators*
  - **<fstream.h>:** Contains information important to user-controlled file processing operations

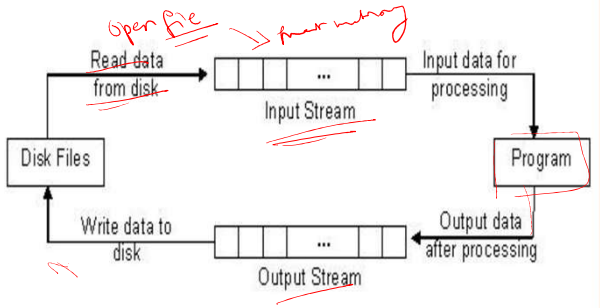
5

## Stream Input/Output Classes and Objects

- **istream:** input streams
  **cin >> someVariable;**
  - **cin** knows what type of data is to be assigned to **someVariable** (based on the type of **someVariable**).
- **ostream:** output streams
  - **cout << someVariable;**
    - **cout** knows the type of data to output
  - **cerr << someString;**
    - Unbuffered - prints **someString** immediately.
  - **clog << someString;**
    - Buffered - prints **someString** as soon as output buffer is full or flushed

6

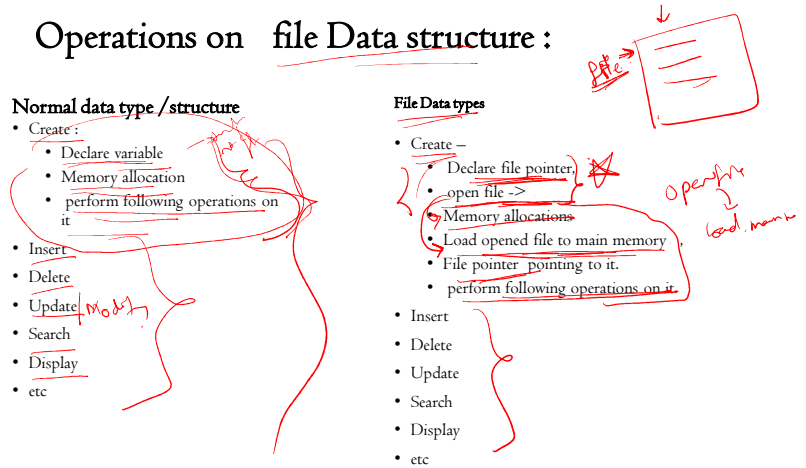
## File input and output streams




7

7

## Operations on file Data structure :




8

## File Operations in c++

- Create
- Open
- Close
- Read
- Write
- Other supporting operations

9

## How to open /create a file in C++ ?

1. Open a file to read/write operations

Syntax ::

```
fileobject.open("filename", openmode);
```

Two possible way to cerate and open

```
fstreamoutFile("account.txt", ios:out);
        OR
fstream outFile;
outFile.open("account.txt", ios:in);
```

Can ask file name at run time

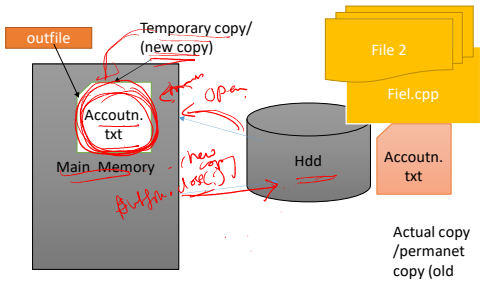
```
int main()
{
   char filename[50];
   cout<< "Enter the valid filename"
   cin>> filename;
   Ofstream outfile( filename, ios::out);
   //OR
   Ofstream outfile;
   outfield.open(filename, ios::out);
}
```

10

10

## What is the outcome of file open /creation File open and close

- Open a file in main memory,
- Allocated part of main memory to it called as tempfie.
- File pointer point to appropriate location of file




11

## Example

```
// Name      : FileOp.cpp
#include <iostream>
#include<fstream>
using namespace std;
int main()
{
 fstream fileI; //data type of file type ( pointer )
 fileI.open("xyz.txt",ios::out);

}
```

```
if (!fileI )
 {
 // overloaded ! operator

   cerr << "File could not be opened" << endl;
   exit( 1 );   // prototype in stdlib.h
 }
 char c;
 while(1)
 {

   cin>>c;
   if(c=='0');
   fileI<<c;

}
```

12

12

## Creating a sequential file

```
// Create a sequential file
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
int main()
{
  // ofstream constructor opens file
  ofstream  outFile( "Account.dat", ios::out );

  if ( !outFile ) {  // overloaded ! operator
    cerr << "File could not be opened" << endl;
    exit( 1 );   // prototype in stdlib.h
  }

  cout << "Enter the account, name, and balance.\n"
       << "Enter end-of-file to end input.\n? ";
  int account;
  char name[ 30 ];
  float balance;

  while ( cin >> account >> name >> balance ) {
    outFile << account << ' ' << name
            << ' ' << balance << '\n';
    cout << "? ";
  }

  return 0;  // ofstream destructor closes file
}
```

Fileprg1.cpp





## File Open Modes

**Default Open Modes :**
- ifstream ios::in
- ofstream ios::out
- fstream ios::in | ios::out

**ios:: app -** (append) write all output to the end of file
**ios:: ate -** data can be written anywhere in the file
**ios:: binary -** read/write data in binary format
**ios:: in -** (input) open a file for input
**ios::out -** (output) open afile for output
**ios: trunc -**(truncate) discard the files' contents if it exists
**ios:nocreate -** if the file does NOT exists, the open operation fails
**ios:noreplace -** if the file exists, the open operation fails





## Mode in combination :

- ofstream outfile;
- outfile.open("file.dat", ios::out | ios::trunc );

- fstream  afile;
- afile.open("file.dat", ios::out | ios::in );

- ofstream myfile ("example.bin", ios::out | ios::app | ios::binary);



## Validation

- To check if a file stream was successful opening a file,
- you can do it by calling to member is_open.
- This member function returns a bool value of true in the case that indeed the stream object is associated with an open file, or false otherwise:
  - if (myfile.is_open())
- Or just checking it
  - If(file1) { }
  - While(file) { }

## How to close a file in C++?

The file is closed implicitly when a destructor for the corresponding object is called

OR

by using member function *close:*

*Fileobject.close();*

*Eg:*

*file1.close();*

**outClientFile.close();**

17

17

## Reading /writing with file : Text file

- Text file streams are those where the ios::binary flag is not included in their opening mode.
- These files are designed to store text
- All values that are input or output from/to them can suffer some formatting transformations,
- which do not necessarily correspond to their literal binary value.

- Writing operations on text files are performed in the same way we operated with cout with insertion << and excursion >> operators.

18

## Reading writing in the file

**Reading character with >>**

```
Int main()
{   char c;
    ifstream file1("xyz.txt",ios:in);
    if(!file1)
      { cout<<"unable to open a file";return0;}
    else{
        while(file1)
        {
         file1>>c;
         cout<<c; } //end while
      } //end else
}
```

**Writing character with <<)**

```
Int main()
{   char c;
    ofstream file1("xyz.txt",ios:out);
    if(!file1)
      { cout<<"unable to create a file";return0;}
    else{
        while(1)
        { cin>> c;
          file1<<c; //writing in the file
          cout<< "want ot enter more";
          cin>>yn;
          if(yn==1)          continue;
          else               break;
        }//end while
    }//end else  } //end of program
```

19

**Taking data form user Writing data in the file**

```
int main () {
  char data[100];
  // open a file in write mode.
  ofstream outfile;
  outfile.open("afile.dat");
  cout << "Writing to the file" << endl;
  cout << "Enter your name: ";
  cin.getline(data, 100);
  // write inputted data into the file.
  outfile << data << endl;
  cout << "Enter your age: ";
  cin >> data;
  cin.ignore();
  outfile << data << endl; //write to file
// close the opened file.
  outfile.close();
```

**Reading data from file and displacing on screen.**

```
  // open a file in read mode.
  ifstream infile;
  infile.open("afile.dat");
   cout << "Reading from the file" << endl;
  infile >> data;
  // write the data at the screen.
  cout << data << endl;
    // again read the data from the file and display it.
  infile >> data;
  cout << data << endl;
  // close the opened file.
  infile.close();
 return 0;
}
```

Fileprog2.cpp

20

# Reading /writing with file : Binary file

- File streams include two member functions specifically designed to read and write binary data sequentially
- write and read function .
- The write is a member function of ostream (inherited by ofstream)
- The read is a member function of istream (inherited by ifstream). Objects of class fstream have both.
- Their prototypes are:

```
write ( memory_block, size );
read ( memory_block, size );
```

  - memory_block is of type char* (pointer to char), represent the address of temp data storage
  - The size parameter is an integer value that specifies the number of characters to be read or written from/to the memory block.

21

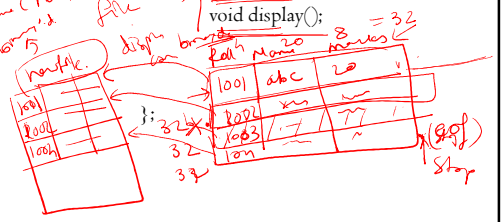
# Reading writing record wise

**Declaration of a record**

```
typedef struct
{       int  dd, mm, yy;
}dob;
typedef struct student
{       int roll_no;
        char name[20];
        int sub[5];
        dob D_of_birth;
        int total;
        float percentage;
        char pnone_no [11];
} student;
```

**Class declaeration**

```
class  stud_data
{
        student s1;
public:
        void input();
        void display();
};
```



22

**Writing**

```
void stud_data::  input()
{
int size;
ofstream outfile("stud_data.txt", ios::app|ios::binary);
 cout<<"\n\t How many records you want to enter :";
  cin>> size;
 for(int i=0;i<size;i++)
 {   //taking one record at a time
        cout<<"Enter the student Information ::";
        cout<<newline1<<" Enter RollNo :::";
        cin>>s1.roll_no;
        cout<<newline1<<" Enter  Name ::";
        cin>>s1.name;
        outfile.write((char*)&s1,sizeof(s1));//
} outfile.close();}//end
```

```
void stud_data::display()
{ ifstream infile("stud_data.txt");
  if (!infile ) {// overloaded ! operator
                cerr << "File could not be opened" << endl;
                exit( 1 );   // prototype in stdlib.h
          }
 cout<<"\n\t      cout<<"\n\n\t Sr.No. \t RollNo \t Name"
          cout<<"\n\t
    int i=0;
  while(!infile.eof()){
 infile.read((char*)&s1,sizeof(s1));
  //hodler  keep the copy after read, how many bytes
  cout<<newline1<<i+1  <<"\t "<< s1.roll_no<<"\t "<<
s1.name;
 i++;      }//end of while
infile.close();
} //end of display
```

23