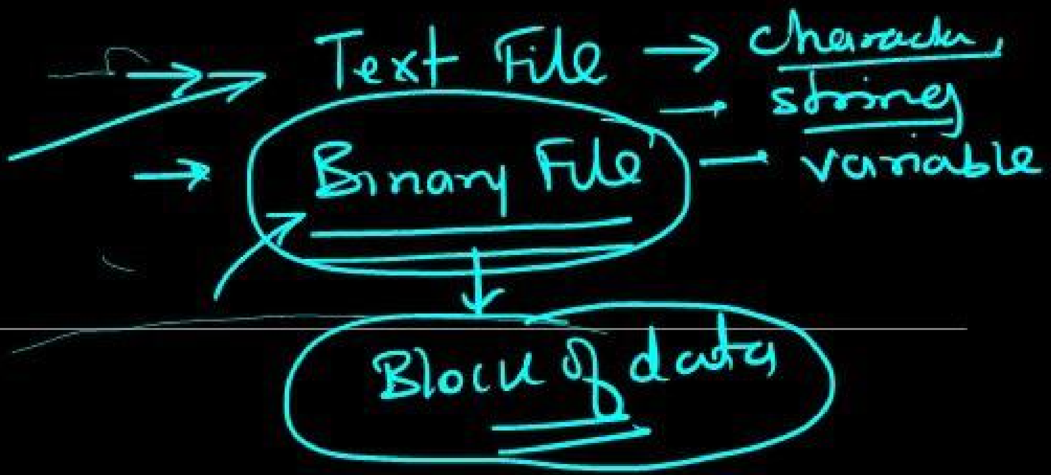
File Handling in C++

Topics



Introduction

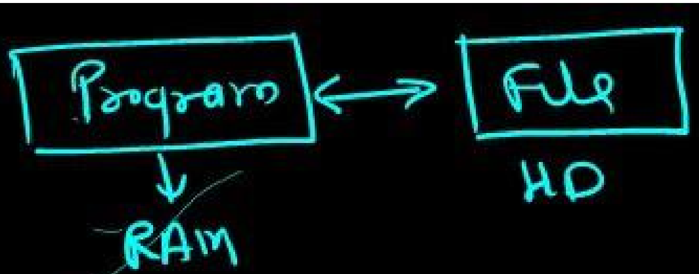
✓ **Opening & closing of files**

✓ **Stream state member functions**

✓ **File operations**

✓ **Binary file operations**

Introduction



HD

- Computer programs are used to manipulate file
  - it helps in storing data & information permanently.
- File - itself a bunch of bytes stored on some storage devices (preferably HD).
- In C++ this is achieved through a component header file called ***fstream.h***
- The I/O library functions manages two aspects-
  - interface with file → connect file with program
  - transfer of data.
- The library predefine a set of operations for all file related handling through certain classes.

The *fstream.h* header file

- Streams act as an interface between files and programs.
- They represent as a sequence of bytes and deals with the flow of data.
- Every stream is associated with a class having member functions and operations for a particular kind of data flow.

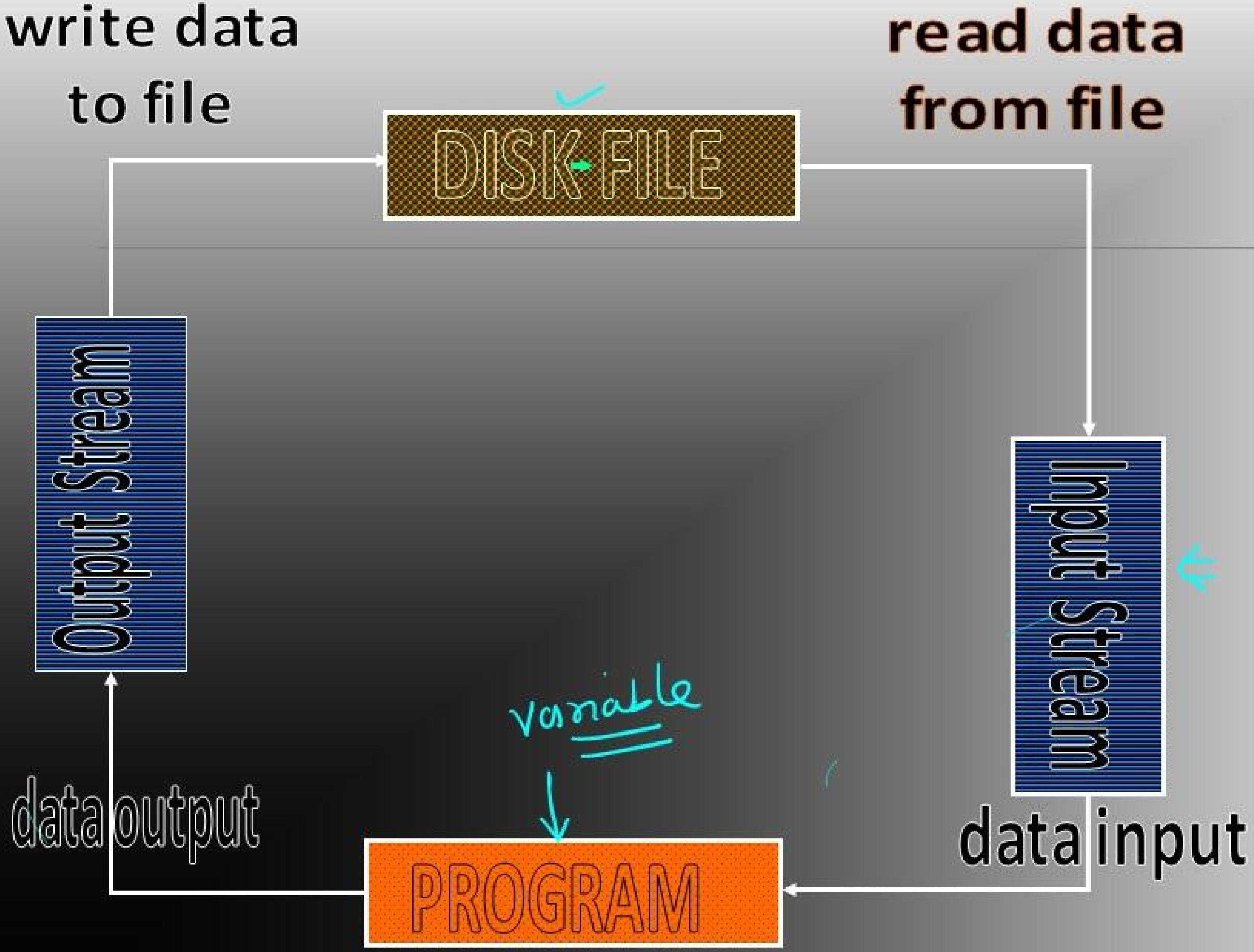
Data flow

- file → program (input stream) - reads
- program → file (output stream) – write

Data flow

⇒

- All designed into fstream.h and hence needs to be included in all file handling programs.
- Diagrammatically as shown in next slide



write data to file

read data from file

DISK FILE

Output Stream

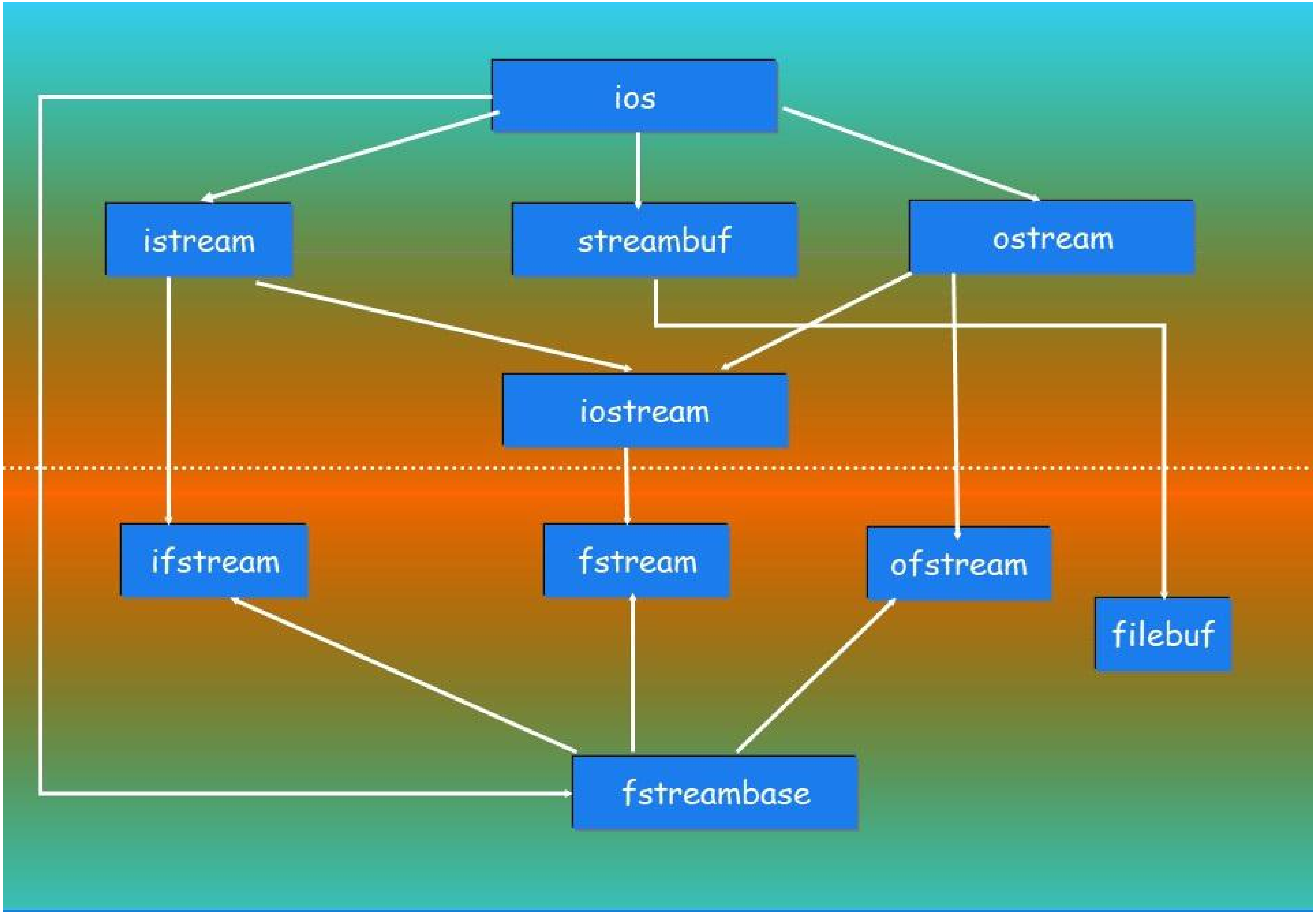
Input Stream

variable

data output

PROGRAM

data input


ios
istream
streambuf
ostream
iostream
ifstream
fstream
ofstream
filebuf
fstreambase

Reasons to use files:

- 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:

1. How to "connect" file to program?
2. How to tell the program to read data?
3. How to tell the program to write data?
4. Error checking and handling eof?

⇒ to make the operations on file

Example: Reading (input) and writing (output) data to a file.

→ ofstream

→ ifstream

open() → predefined fn

```
#include <fstream>
#include <iostream>
using namespace std;
int main ()
{
  char name[50];

  ofstream ofile;   // open a file in write mode.
  ofile.open("abc.txt");

  cout << "Writing to the file" << endl;
  cout << "Enter your name: "<<endl;
  cin.getline(name, 50);
  if (ofile.fail())
 { cout << "Output file could not be opened.\n";
   exit(1);  }
  else
  ofile << name << endl;    // write inputted data
                               into the file.

  ofile.close();       // close the opened file.
```

string

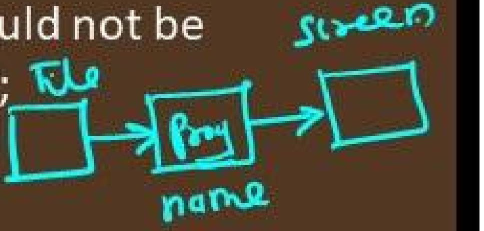
object

gets (name);

```
ifstream ifile;    // open a file in read mode.
ifile.open("abc.txt");

  cout << "Reading from the file" << endl;
  if (ifile.fail())
  { cout << "Input file could not be
                   opened.\n";
   exit(1);
  }
  else
  ifile >> name;

  cout << name << endl;   // write the    data
                                  at the screen.
  ifile.close();        // close the opened file.

  return 0;
}
```

File

cin>> name;



File Handling Classes

- **When working with files in C++, the following classes can be used:**
  - **ofstream** - writing to a file ✓ obj 1 → writing
  - **ifstream** - reading for a file ✓ obj 2 - reading
  - **fstream** - reading / writing ✓ obj3 → r|w
- ✓ **What does it all have to do with cout?**
  - When ever we include **<iostream.h>**, an **ostream** object, pointing to **stdout** is automatically defined - this object is **cout**.
- ✓ **ofstream** inherits from the class **ostream** (standard output class).
- ✓ **ostream** overloaded the operator >> for standard output….thus an **ofstream** object can use methods and operators defined in **ostream**.

Opening & Closing a File

- A file can be open by the method "open()"
- by constructor (the natural and preferred way).

```
void ofstream / ifstream::open(const char* filename, int mode);
```

- filename – file to open (full path or local)
- mode – how to open (1 or more of following – using |)
  - ios::app – append
  - ios::ate – open with marker at the end of the file
  - ios::in / ios::out – (the defaults of ifstream and ofstream)
  - ios:nocreate / ios::noreplace – open only if the file exists, / doesn't exist
  - ios::trunc – open an empty file
  - ios::binary – open a binary file (default is textual)
- Don't forget to close the file using the method "close()"

1: *To access file handling routines:*

```
#include <fstream.h>
```

2: *To declare variables that can be used to access file:*

```
ifstream in_stream;   → read
ofstream out_stream;  → write
```

3: *To connect your program's variable (its internal name) to an external file (i.e., on the Unix file system):*

```
in_stream.open("infile.dat");   → read mode
out_stream.open("outfile.dat"); → write mode
```

4: *To see if the file opened successfully:*

```
if (in_stream.fail())
{    cout << "Input file open failed\n";
exit(1);    // requires <stdlib.h>}
```

***5**: To get data from a file (one option), must declare a variable to hold the data and then read it using the **extraction** operator:*

```
int num;
in_stream >> num;
[Compare: cin >> num;]
```

✓ Reading data from file

6: To put data into a file, use insertion operator:

```
out_stream << num;
[Compare: cout << num;]
```

✓ writing data to a file

NOTE: Streams are sequential – data is read and written in order – generally can't back up.

7: When done with the file:

```
in_stream.close();
out_stream.close();
```

} closing the file

Stream state member functions

- In C++, file stream classes inherit a stream state member from the "ios" class, which gives out the information regarding the status of the stream.

FOR E.G.:

end of the file

eof() –used to check the end of file character

fail()- used to check the status of file at opening for I/O

bad()- used to check whether invalid file operations or unrecoverable error .

good()- used to check whether the previous file operation has been successful

object of specific class — ifstream, ofstream, fstream

Reading and writing block of data to binary file

- **To write n bytes:**

  ✓ **write ((unsigned char*) &buffer, sizeof(buffer));**

- **To read n bytes** (to a pre-allocated buffer):

  ✓ **read ((unsighed char*) &buffer, sizeof(buffer));**

```
#include <fstream>
#include <iostream>
using namespace std;
int main ()
{
   int i;
   ofstream outfile; int arr1[10];
   int arr[] = {10,20,30,40,50};
   outfile.open("myfile", ios::out | ios ::binary);
   outfile.write((char*)&arr, sizeof(arr));
   outfile.close();*/
```

```
ifstream infile;
  infile.open("myfile", ios::in);
  infile.read((char*)&arr1,sizeof(arr1));
  for(i=0;i<5;i++)
  {
  cout<<arr1[i]<<" ";
  }

  infile.close();
}
```

" file name "

empty

filename

10	20	30	40	50

block

memory

arr → buffer → memory

10 20 30 40 50 60 70 80 90 100

Appending block of data to binary file

```
#include <fstream>
#include <iostream>
using namespace std;
int main ()
{
   int i;
   ofstream outfile; int arr1[10];
   int arr[] = {60,70,80,90,100};
   outfile.open("myfile", ios::out | ios :: app);
   outfile.write((char*)&arr, sizeof(arr));
   outfile.close();
```

```
ifstream infile;

infile.open("myfile", ios::in);

 infile.read((char*)&arr1,sizeof(arr1));

   for(i=0;i<10;i++)

   {

   cout<<arr1[i]<<"  ";

   }

   infile.close();

}
```

Example: Reading and writing object to file

block of data

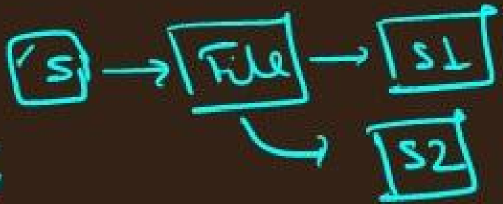
```
#include <fstream>
#include <iostream>
using namespace std;
class student
{
   char name[30];
   int rn;
   char div;
   public:
   void getdata()
   { gets(name);cin>>rn>>div; }
   void putdata()
   {
cout<<name<<endl<<rn<<endl<<div<<endl;}
};
```

33 bytes

Optional

read operation

```
int main ()
{
  int i;
  student  s,s1;, s2 ;
  ofstream  outfile;
  cout<<"Read  data of student"<<endl;
  s.getdata();
  outfile.open("myfile",  ios::out|  ios :: binary);
  if (outfile.fail())
  { cout << "Output file could not be opened.\n";
  exit(1);  }
  else
  {
  outfile.write((char*)&s,  sizeof(s));
  }
  outfile.close();

  ifstream  infile;
  infile.open("myfile",  ios::in);
  infile.read((char*)&s1,sizeof(s1));
  cout<<"Student  data is :"<<endl;
  s1.putdata();
  infile.close();
}
```



char

object

size of an object

Example: Reading and writing array of objects to file

```
#include <fstream>
#include <iostream>
#include<stdio.h>
using namespace std;
class student
{
  char name[30];
  int rn;
  char div;
  public:
  void getdata()
  { gets(name);
     cin>>rn;
     cin>>div;
     scanf("%*c");
  }
  void putdata()
  { cout<<name<<endl<<rn<<endl<<div;}
};
```

33
name
XYZ
12
s[0]
s[1]
s[2]

2 byte

1 byte

XYZ
12
E

2

cin>>name;

```
int main ()
{
   int i;
   student s[3],s1[3];
   ofstream outfile;
   outfile.open("myfile", ios::out | ios :: binary);
   cout<<"Read data of student"<<endl;
   for(i=0;i<3;i++)
   {   s[i].getdata();   }
   outfile.write((char*)&s, sizeof(s));
   outfile.close();

   ifstream infile;
   infile.open("myfile", ios::in);
   infile.read((char*)&s1,sizeof(s1));

   cout<<"Student data is :"<<endl;
   for(i=0;i<3;i++)
   {  s1[i].putdata();   }
   infile.close();
}
```

outfile.write((char *)

& s[i], sizeof
(s[i]))

one object at a time

entire array at a time

File operations(textual file)

the following member functions are used for reading and writing a character from a specified file.

get()- is used to read an alphanumeric character from a file.

put()- is used to write a character to a specified file or a specified output stream

Objects

CHARACTER I/O

C++ has some low-level facilities for character I/O.

```
char next1, next2, next3;
cin.get(next1);
```

Gets the next character from the keyboard. Does not skip over blanks or newline (\n). Can check for newline (next == '\n')

Example:

```
cin.get(next1);
cin.get(next2);
cin.get(next3);
```

Predefined character functions must #include <ctype.h> and can be used to

- **convert between upper and lower case**
- **test whether in upper or lower case**
- **test whether alphabetic character or digit**
- **test for space**

Example: Reading and writing file character by character

I am a CSBS student \0

0 1 2 3 4 5 6 7 8 9 10 11 12 13

name

```
#include <fstream>
#include <iostream>
using namespace std;
int main ()
{
  char name[50];
  char a; int i=0;
  ofstream ofile;
  ofile.open("abc.txt");

  cout << "Writing to the file" << endl;
  cout << "Enter your name: "<<endl;
  cin.getline(name, 50);
  while(name[i]!='\0')
  {
  ofile.put(name[i]); //<< endl;
  i++;
  }
  ofile.close();
```

gets(name);

```
ifstream ifile;
  ifile.open("abc.txt");

  cout << "Reading from the file" << endl;
  while(!ifile.eof())
  {
  ifile.get(a);
  cout<<a<<" ";
  }
  ifile.close();

  return 0;
}
```

end of the file

I am a CSBS stude