# File Handling

File - A bunch of bytes stored on a system. There are two types of files.

1. Text files
2. Binary files.

| Text files | Binary Files |
|---|---|
| - In these files, data is stored in the form of ASCII codes irrespective of data types. | In these files, data is stored in the same way as it is stored in the memory irrespective of data types. |
| - Can be edited using a text editor or a program | Can be edited only through programs |
| - Size of a text file is bigger | Size of a binary file is smaller |
| - If the data is written to a text file, the output of a file will require conversion of each character before displaying on the screen. | If the data is written to a binary file, the user will see a bunch of blackblocks or some symbols or random binary numbers if opened in any editor application. Therefor each character is stored the way it is stored in the memory hence no conversion is required |

Note: By default C++ opens files in text mode.

Operations on files.

1. Creation of a file
2. Opening an existing file.

4. Reading from a file.
5. Writing to a file
6. Modifying (Manipulating) a file.
7. Detecting the end of file
8. Closing of a file.

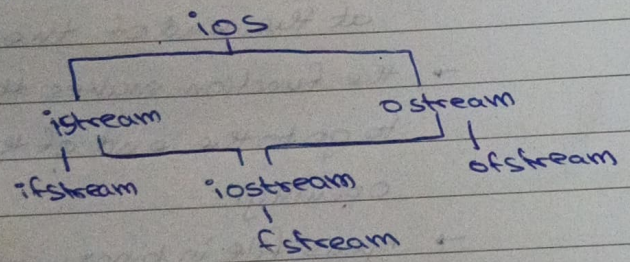
Stream classes.

Stream - A sequence of bytes. There are two types of streams.

① Input stream - Supplies data to a program. It is used to read from a file.

② Output stream - Recieves the data from the program or file. It is used to write to a file.

Input Output Stream class hierarchy




Functions.

1. ifstream - provides input operation which contains function open() with default input mode. The object of ifstream is used to open a file in read mode.

2. ofstream - provides output operations which contain function open() with default output mode. The object of ofstream is used to open a file in write mode.

3. fstream - provides support for simultaneous input and output operations which contains function open() with default input mode. The object of fstream class allows both,

read and write operations on files. Hence, it is required to specify the purpose of the mode.

Different file modes

File modes - the general form of opening a file using function open() requires two arguments. stream object name. open ("filename", mode) The second argument specifies the purpose for which the file is opened.

| | file mode | | purpose. |
|---|---|---|---|
| 1. | ios::in | → | opens file for only reading. |
| 2. | ios::out | → | opens file for only writing |
| 3 | ios::app | → | opens file to append the data at the end of the file |
| 4. | ios::ate | → | the function makes the position to go to the end of the file on opening. |
| 5. | ios::binary | → | opens file in binary mode. |
| 6. | ios::nocreate | → | open fails if the file does not exist |
| | ios::noreplace | → | opening of the file fails if the file already exists |
| 7. | ios::trunk | → | on opening a file, it deletes the contents of the file if it exists |

Errors in opening files

1. For reading

```
ifstream a;
a.open ("filename");
if (!a)
cout << "Error";
```

For writing

```
ofstream a;
a.open("filename");
if(!a)
cout<<"Error";
```

Detecting the end of file

1. 
```
ifstream a;
a.open("filename");
while(a)
{ == }
```

using eof() function

2. 
```
ifstream a;
a.open("filename");
if(a.eof()!=0)
{ == }
```

Reading and writing functions for text files.

1. get()
- extracts a single character from the input stream

getline()
reads the entire line of characters until a user-defined delimiter ("\n", "\t") is encountered

2. get()
- extracts a single character from the input stream
- the character extracted can be a white space character.

put()
inserts a single character to the output stream. puts the next character to the output stream even if it is a white space character.

3. ios :: out
- creates a new file and writes the data to it if the file is not present. If the file is present, it over writes the existing data and allows to

ios :: app
allows to append the text in an already existing file (appends at the end of the file). If file is not present then it creates a new one and allows to enter

enter new contents to the file. data to the file.

Reading and writing to a text file using objects, get() and put()

1. class name bill

public : billno - integer.

billamt - float.

Write a function main to open a file bill.txt and write the billnumber and bill amount entered by the user to the file bill.text. Also display the contents of the file.

Soln

```
void main()
{bill a; clrscr();
 ifstream o;
 ofstream p;
 p.open("bill.txt");
        if(!p)
 cout<<"Error";
 else
 {cout<<"Enter the bill number and bill amount";
  cin>> a.billno >>a.billamt;
  p<< a.billno << a.billamt;} p.close();
  o.open("bill.txt");
  if(!o)
  cout<<"Error";
  else { o>> a.billno >>a.billamt;
      cout << a.billno << a.billamt;} o.close();
  getch();}
```

2. Write a function in c++ to count the words 'to' and 'the' present in a text file poem.txt.

Soln.

```
void    count().
{int c1=0, c2=0; char ch[3];
a.open("poem.txt")
```

```
{while (a)
{ a>> ch ;
  if (strcmp (ch ,"to" )==0)
    c1++
    if (strcmp (ch,"the")==0)
    c2++ }} a.close();
    cout<<c1 << c2 ; getch();}
```

3. To count and display the number of alphabets and number of spaces in an existing file line.txt.

Soln
```
void count()
{int c1=0,c2=0; ifstream a;   char b;
  a.open ("line.txt");
  {while (a)
  {   a.get(b);
  if (b>='A' && b<='Z' || b>='a' && b<='z') OR if (isalpha(b))
    c1++;
    if (strcmp b==' ')
    c2++;}} a.close();
    cout<<c1 << c2 ; getch();}
```

4. To open a file target.txt and copy the following text to the file. Text given: Leroy Martin Do Rosario Noronha - Also open the file source.txt and copy its contents to target.txt. Display the contents of target.txt.

Soln
```
void modify()
{char   a[]="Leroy Martin Do Rosario Noronha", ch;
  fstream b,c;
b.open ("source.txt,ios::in);
c.open ("target.txt,ios::out);
for (int i=0;i<strlen(a);i++)
{c.get (a[i]) ; c.put(a[i]);}
```

```
while (b)
{b.get (ch);
 c.put(ch);}
b.close();
c.close();
c.open("target.txt",ios::in);
while (c)
{c.get(ch);
 cout<<ch;}
c.close(); getch();}
```

5. To calculate the average word size in the text file lines.txt assuming each word is separated by a single space or full stop. Display the number of words and the average word size. (Total number of alphabets / Total number of words)

Soln
```
{int c1=0, c2=0; char ch;
 fstream a;
 a.open("lines.txt",ios::in);
 while(a)
 {a.get(ch)
  if (ch==' '||ch=='.')
     c1++;
  else c2++;} a.close();
 cout<<"the average word size is"<<c2/c1;
 getch()}
```

6. To convert the upper case letters of the already existing file sample.txt to lower case letters and vice versa

Soln
```
{char ch;
 fstream a;
 a.open("sample.txt",ios::in)
 while(a)
```

```
{a.get(ch)
 if(ch>=65 and ch<=90)
   ch=ch+32;
 else
 if(ch>=97 and ch<=122)
 ch=ch-32; cout<<ch;}a.close();}
```

7. To count the number of lower case letters in a text file.

Soln
```
{char ch; int i;
 fstream a;
 a.open("sample.txt",ios::in)
 while(a)
 {a.get(ch)
   if(ch>=97 and ch<=122)
   i++;} a.close();
   cout<<i;
```

8. Assuming that the textfile text1.txt already contains some text written to it, write a function modify() that reads the file text1.txt and creates a new file text2.txt which should contain only those words from the file text1.txt which don't start with an upper case vowel.

Soln
```
{char ch;
 fstream a,b;
 a.Open("text1.txt",ios::in)
 b.open("text2.txt",ios::out)
 while(a)
 {a.get(ch);
  if(ch=='A'||ch=='E'||ch=='I'||ch=='O'||ch=='U')
   {while(ch!=' ')
     {a.get(ch);};
  else b.put(ch){a.close(); b.close(); getch();}
```

9. which start with an upper case vowel.

. Using function getline()

1. Write a C++ program to create two text files named country.txt and capital.txt to store names of n countries and their corresponding capitals. Read both the files and display the names of the countries and their capitals.

Soln
```
char ctr[20],cap[20]; int n;
ofstream a; ifstream b,c;
a.open("country.txt"); cout<<"Enter the number of countries";
cin>>n;
for(i=0;i<n;i++){cout<<"Enter the name of country"<<i+1;
gets(ctr); a<<ctr<<endl;} a.close();
a.open("capital.txt");
for(i=0;i<n;i++){cout<<"Enter the name of capital"<<i+1;
gets(cap); a<<cap<<endl;} a.close();
b.open("country.txt"); c.open("capital.txt");
while(b){b.getline(ctr,20); cout<<ctr<<"\n";
c.getline(cap,20); cout<<cap<<"\n";} b.close(); c.close();}
```

2. Write a function count() and display total number of lines and number of lines starting with alphabet 'A' present in an existing file lines.txt.

Soln
```
{ifstream a;
 int c1=0,c2=0; char ar[50];
 a.open("lines.txt"); while(a){a.getline(ar,50); c1++;
 if(ar[0]=='A') c2++;} a.close(); getch();}
```

Reading and writing in binary files.

Reading and writing in binary files is done using functions read() and write() respectively.

if class name is xyz, then reading in binary form is done

1. Reading → ifstream a; xyz b; a.read((char *)&b, sizeof(b));
2. Writing → ofstream a; xyz b; a.write((char*)&b, sizeof(b));

1. class name → inventory

private - name → character array size 20

- code → integer, cost → float

public - readdata(); → allows the user to enter values for data members.

writedata() → displays data members on the screen.

Write function main() to store details of one item into the file stock.dat and read from a file to display the contents on the screen.

(Assume all functions of the class are already defined)

Soln void main()

{inventory a; fstream b; b.open("stock.dat", ios::out | ios::binary | ios::in);

a.readdata(); b.write((char *)&a, sizeof(a));

b.read((char *)&a, sizeof(a)); a.writedata(); b.close();

}

2. Class name - student

private - rollno - integer

name - character array size 30

marks - integer array size 6 storing marks in 6 subjects.

public - input() - allows the user to enter values for data members

show() - displays all data members on the screen

Write function main to accept details of n students from the user and write to a file records.dat and display the details on the screen.

Soln void main()

{student a[20]; fstream b; int n;

b.open("records.dat", ios::out | ios::binary | ios::in);

cout << "Enter the number of students"; cin >> n;

```
for(i=0;i<n;i++)
{a[i].input();b.write((char*)&a[i],sizeof(a[i]));}
for(i=0;i<n;i++)
{b.read((char*)&a[i],sizeof(a[i])); a[i].show();}
 b.close(); getch();}
```

3. Write a function show() to read all records present in an already existing file lines.dat and display it on the screen. Also count and display the number of records present.

```
class vehicle
{char vehicletype[10];int no;
 public getdetails()- to accept the values for data members
      showdetails()- to show the data members on the screen
```

Soln

```
void vehicle::show()
{vehicle v; ifstream a;
  int c=0; a.open("lines.dat",ios::binary);
  while(a)
{ a.read((char*)&v,sizeof(v)); c++; v.showdetails();
 a.close(); cout<<"Total number of records are"<<c;}
```

4. class employee

```
{int code; char name[20]; float salary;
public: void input()- accept data members
        void show()- display
        float returnsalary()- returns the salary of the employee}
```

Write a function display() to read the existing the binary file company.dat, to display all records on the screen where salary is between 20000 & 30000. Also create another file backup to write the same records from the file company.dat.

Soln.

```
void display()
{ fstream a,b; employee e; a.open("company.dat",ios::in|ios::binary);
  b.open("backup.dat",ios::out|ios::binary); while(a)
{ a.read((char*)&e,sizeof(e)); if(e.returnsalary()> 20000 && e.returnsalary
                                                    <30000)
```

6. Given a binary file telephone.dat containing records of the following class

```
class directory
{char name[20], address[30], areacode[5], phoneno[15];
 public: void register();
         void show();
         int checkcode(char A[]);
          {return strcmp(areacode, A[]);}};
```

Write a function copy() that copies all records having areacode 123 from telephone.dat to teleback.dat

Soln

```
void directory::copy()
{fstream a,b;
 directory d;
a.open("telephone.dat", ios::in|ios::binary);
b.open("teleback.dat", ios::out|ios::binary);
while (a)
{a.read((char *)&d, sizeof(d));
 if (d.checkcode("123")==0)
b.write((char *)&d, sizeof(d));
}a.close(); b.close();}
```

7. class game

```
{char gamename[20], participantname[30];
  int noofparticipants;
  public: void input(); void output();};
```

write a function that reads the contents of the file game.dat and creates a file basket.dat copying only records from game.dat where the game is "basketball".

Soln

```
void game::update()
{fstream a,b; game d;
a.open("game.dat", ios::in|ios::binary);
b.open("basket.dat", ios::out|ios::binary);
```

```
while(a)
{a.read((char *) &d, sizeof (d));
if(strcmp(d.gamename, "Basketball")==0)
b.write ((char *)&d, sizeof (d));
```

8. class camera
{int modelno; float pixelsize; int zoom;
public: void enter();
void display();
int getmodelno(); };
Write a function search to accept modelno from the user and check whether it is present or not. If it is present in the file camera.dat, display the details on the screen.

Soln

Random Access in files

One can move the pointer to any desired location in files.
The pointers associated are get pointer (input pointer) and put pointer (output pointer)

Note: ① Get pointer is associated to a file only when it is opened for reading. eg. HELLO WORLD
↑ get pointer.
In append mode.
② Put pointer is associated to a file when it is opened for writing. eg. HELLO WORLD
↑ put pointer.

③ When a file is opened, all the file pointers shown above take their place automatically, by default. One can move these pointers to a desired location by using the following functions.

i) seekg() - moves the get pointer to any specified location when the file is opened for reading. This function is associated with ifstream.h.

ii) seekp() - moves the put pointer to any specified location when the file is opened for writing. This function is associated with ofstream.h.

iii) tellg() - returns the current position of the get pointer. It does not accept any arguments and is associated with ifstream.h.

iv) tellp() - returns the current position of the put pointer. It does not accept any arguments and is associated with ofstream.h.

Syntax

i) seekg() — ifstream.seekg(offset, ref position). object name

ii) seekp() — ofstream.seekp(offset, ref position). object name

The ref position in seekg and seekp takes one of the following 3 constants defined in 'input-output stream class.

1) ios::beg - start of the file
2) ios::cur - current position of the pointer
3) ios::end - end of the file.

Assume a is the ifstream and b is the ofstream objects. Specify the purpose of the commands.

i) a.seekg(0, ios::beg);
ii) a.seekg(0, ios::cur);
iii) a.seekg(0, ios::end);
iv) a.seekg(m, ios::beg);

v) a.seekg(-m, ios::cur);

vi) a.seekg(-m, ios::end);

vii) b.seekp(50, ios::beg);

viii) b.seekp(-50, ios::end);

ix) b.seekp(0, ios::beg);

x) b.seekp(-m, ios::cur);

write the c++ instructions to

1) move the get pointer by 15 positions backward from the current position. Assume a is ifstream and b is ofstream object.

a.seekg(-15, ios::cur);

2) go to the beginning after opening the file

a.seekg(0, ios::beg);

3) go backward by 20 bytes from the end of the file when it is opened for writing

b.seekp(-20, ios::end);

4) go to byte number 50 in the file when opened for reading

a.seekg(50, ios::beg);

Syntax for tellg and tellp

ifstream object name.tellg();

ofstream object name.tellp();

Note: The position of the get pointer can be used to adjust the position of the put pointer.

eg. Move the position of the put pointer 10 bytes backward from current position of the get pointer

b.seekp(tellg()-10, ios::cur);

File name : student.dat

```
class participant
{int phone;
  char name[30];
 public: void getdata();
         void putdata();};
```

Write a user defined function split that reads the contents of the file student.dat and creates pavaji.dat and baga.dat to store details of all students whose phone number starts from 24 and 26 resp

soln

```
void participant :: split()
{int ph[20]; int x;
 fstream a,b,c;
  participant p;
  a.open("student.dat",ios::binary|ios::in);
  b.open("pavaji.dat", ios::out);
  c.open("baga.dat"; ios::out);
 while(a).
{a.read((char*)&p,sizeof(p));
   x= p.ph;
 while (x<100)
{x = x/10;}
if(x==24)
b.write((char*)&p,sizeof(p));
else if (x==26)
c.write((char*)&p,sizeof(p));}
a.close(); b.close(); c.close();}
```