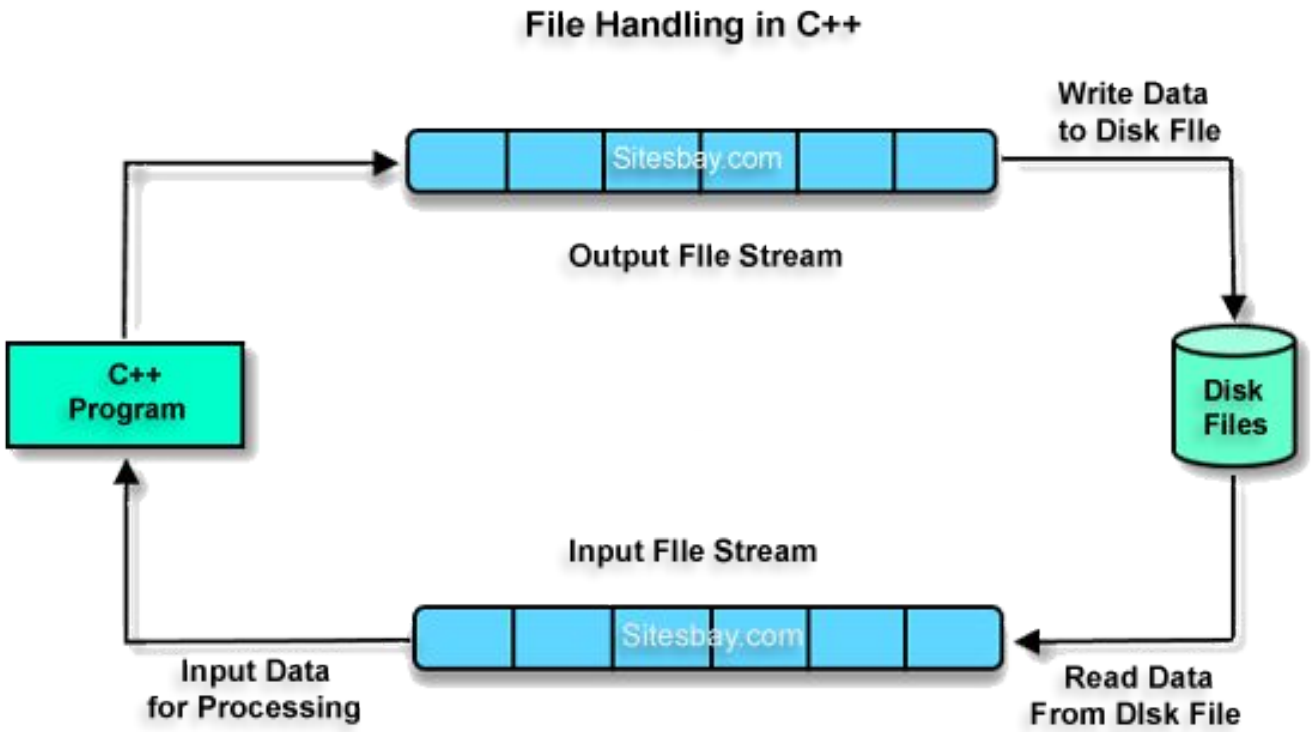
File handling

File handling

File Handling concept in C++ language is used for store a data permanently in computer. Using file handling we can store our data in Secondary memory (Hard disk)



Why use File Handling in C++

- For permanent storage.
- The transfer of input - data or output - data from one computer to another can be easily done by using files.

  For read and write from a file you need another standard C++ library called fstream, which defines three new data types:

Datatype	Description
ofstream	This is used to create a file and write data on files
ifstream	This is used to read data from files
fstream	This is used to both read and write data from/to files

Focus on Software Engineering: The Process of Using a File

Using a file in a program is a simple three-step process

- The file must be opened. If the file does not yet exits, opening it means creating it.
- Information is then saved to the file, read from the file, or both.
- When the program is finished using the file, the file must be closed.

File Input/Output

Done with the same operations (insertion, extraction) as keyboard input and monitor output

Simply open input or output object with connection to a file and use it where you would use cin or cout

To use

- include <fstream>
- create input object of type *ifstream*
- or output object of type *ofstream*

Writing data to a file

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
ofstream output;
output.open("scores.txt");
output<<"John T Smith 90"<<endl;
output.close();
cout<<"done";
}
```

Reading Data from File

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;
int main()
{
ifstream input("scores.txt");
string fname,lname;
int score;
char m;
input>>fname>>m>>lname>>score;
input.close();
cout<<fname<<m<<lname<<score<<endl;
cout<<"done";
}
```

Testing file existence

```
ifstream input("scors.txt");
if(input.fail()) or if(!input)
{
cout<<"file does not exist";
return 0;
}
```

getline

getline(ifstream &input, string s, char delimitChar)

The function stop reading characters when the delimiter character or end-of-file mark is encountered.

If the delimiter is encountered, it is read but not stored in the array.

The third argument delimitChar has a default value (‘\n’)

Example

Suppose a file named state.txt is created that contains the state names delimited by (#) symbol

```
int main()
{
ifstream input("states.txt");
if(input.fail())
{
cout<<"file does not exist";
return 0;
}
//read data
string city;
while(!input.eof())
{
getline(input, city, '#');
cout<<city<<endl;
}
input.close();
return 0;
}
```

Output:

```
New York
New Mexico
Texas
indiana
```

Copy file Cotents

Read data from one file

Put the data to another file

Functions used

- Get()
  - Reads characters
- Put(ch)
  - Write characters
- Eof()
  - Checks whether its end of file or not

```
int main(){
    ifstream input("states.txt");
    ofstream output("scores.txt");
    if(input.fail()){
        cout<<"File does not exist \n";
        cout<<"Exit Program";
        return 0;
    }
    char ch =input.get();
    while(!input.eof()){
        output.put(ch);
        ch=input.get();
     }
    input.close();
    output.close();
    cout<<"\n Copy done";
    return 0;
}
```

fstream and file open modes

Ofstream is used to write data

Ifstream to read data

But it is convenient to use fstream if your program needs to use the same stream object for both input and output

To open an fstream file you have to specify a file mode to tell c++ how the file will me used

File modes

	Description
ios::in	Opens a file for input
ios::out	Open a file for output
ios::app	Appends all output to the end of the file
ios:: ate	Opens the file for output. If the file already exists, move to the end of the file. Data can b written anywhere in the file.
ios::truct	Discards the file's contents if the file already exists.
ios::binary	Opens a file for binary input and output

```
int main(){
fstream inout;
inout.open("city.txt", ios::out);
inout<<"Dallas"<<" "<<"houston";
inout.close();
inout.open("city.txt",ios::out | ios :: app);
inout<<" lahore"<<" "<<"islamabad";
inout.close();
string city;
inout.open("city.txt",ios::in);
while(!inout.eof()){
inout>>city;
cout<<city<<" ";
}
inout.close();
}
```

Testing stream states

C++ provide many functions to test the stream states

Each stream object contains a set of bits that act as flags

These bit values 0 or 1 indicate the state of a stream

Stream state bit values

	Description
ios::eofbit	Set when the end of an input stream is reached
ios::failbit	Set when an operation is failed
ios::hardfail	Set when an unrecoverable error has occur
ios:: badbit	Set when an invalid operation has been occurred
ios::goodbit	Set when an operation is successful

Stream state functions

	Description
eof()	Returns true if eofbit flag is set
fail()	Returns true if the failbit and hardbit flag is set
Bad()	True if badbit is set
Good()	True if goodbit is set
Clear()	Clear all flags

Program

```
void showState(fstream &);
void showState(fstream &stream)
{
cout<<"stream status\n";
cout<<"eof() "<<stream.eof()<<endl;
cout<<"fail() "<<stream.fail()<<endl;
cout<<"bad "<<stream.bad()<<endl;
cout<<"good() "<<stream.good()<<endl;
}
```

```
int main()
{
fstream inout;
inout.open("temp.txt",ios::out);
inout<<"Lahore";
cout<<"Normal operations no errors\n";
showState(inout);
inout.close();
inout.open("temp.txt",ios::in);
string city;
inout>>city;
cout<<"End of File (no errors)"<<endl;
showState(inout);
inout.close();
inout>>city;
cout<<"Bad operation (errors)"<<endl;
showState(inout);
return 0;
}
```

```
int main(){
    fstream fout;
  // opens an existing csv file or creates a new file.
  fout.open("reportcard.csv", ios::out | ios::app);
  cout << "Enter the details of 5 students:"
      << " roll name maths phy chem bio";
  int i, roll, phy, chem, math, bio;
  string name;
  // Read the input
  for (i = 0; i < 2; i++) {
     cin >> roll>> name>> math>> phy>> chem>> bio;
     // Insert the data to file
     fout << roll << ", " << name << ", "<< math << ", "<< phy << ", "<< chem << ", "<< bio << "\n";
  }
}
```