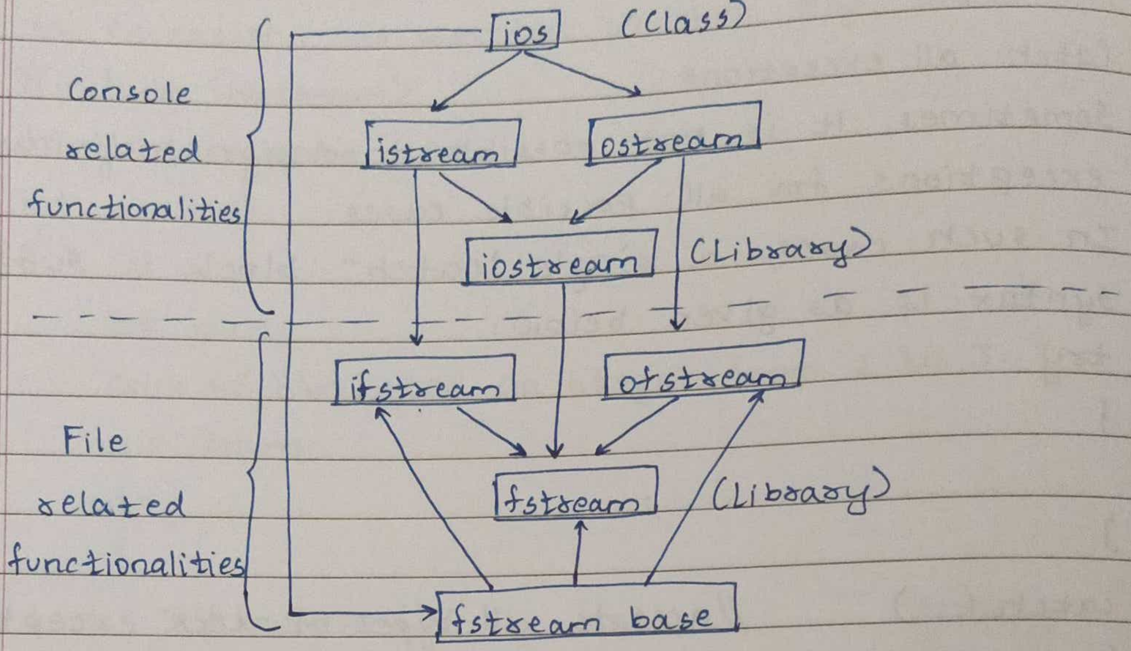
29/11/2022

File Handling

- ☆ cin → istream
  cout → ostream
- ☆ To write something in a file: ofstream (output file stream
  To read something from a file: ifstream (input file stream)
- ☆



- ☆ When large amount of data needs to be handled, we need to store the data ~~on a~~ somewhere on a secondar storage memory.
- ☆ The data is stored on a secondary storage device using the concept of files.
- ☆ File is a collection of "related data" stored in a particular area of a disk.
- ☆ Programs can be designed to perform read and write operations on these files.
- ☆ The io system of C++ handles file operations which are very much similar to console's input and output operations.
- ☆ It uses filestreams as an interface between the programs and files.

classmate
Date
Page

☆ Filestream Classes

→ Input-output system of C++ contains a set of classes that defines the file-handling methods. These include:

1. ifstream
2. ofstream
3. fstream

→ These classes are derived from fstream base and corresponding iostream class.

☆ Classes

1. fstream base
→ Provides operations common to filestreams.
→ Serves as a base for ifstream, ofstream and fstream classes.
→ It contains open and close functions.

2. ifstream
→ Provides input operations.
→ Contains open() with default input mode and close().
→ Inherits the functions such as get(), getline(), read(), seekg() (allocates the ~~po~~ starting position in a file), tellg() (gives the position of the cursor ~~of~~ in the file), etc.

3. Ofstream.
→ Provides output operations.
→ Contains open() with default output mode and close().
→ Inherits put(), seekp() (putting the "pointer" at an offset), tellp() and write() from ostream.

4. fstream
→ Provides support for simultaneous input/output operations

→ Inherits all functions from istream and ostream classes through iostream.
→ Contains open() with default input mode.

★ **Write Data in File**

→
```
#include <fstream>
int main()
{
      ofstream file;
      file.open("sample.txt");   //path of the file
      file << "Hi all!";         //writes "Hi all" in the file.
      file.close();
}
```
→ file.open("....") creates the file if it does not exist.

★ **Read Data in File**

→
```
#include <fstream>
int main()
{
      ifstream file;
      file.open("sample.txt");
      ~~file>>~~ char str[50];
      file >> str;
      file.close();
}
```
→ Line-by-line:
```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    ifstream file;
    char str[50];
    file.open("sample.txt");
    if (!file.is_open())
    {
        cout<<"Unable of open file.\n";
        return 0;
    }
    while(getline(file, str))
        cout<<str<<endl;
    file.close();
    return 0;
}
```

→ Taking n inputs in one iteration:

```
ifstream file;    file.open("sample.txt");
int a, b, c;
while (!file.eof())
{
    getline(file, a, '\n');
    getline(file, b, '\n');
    getline(file, c, '\n');
    cout<<endl<<a<<b<<c;
}
file.close();
```

→ File: Biology $11   12

```
ifstream file;    file.open("Sample.txt");
int temp;
string subject;
int cost, score;
file>>subject;
temp= file.tellg();   //to get $'s position.
```

```
file.seekg(temp+1);  //move pointer past $.
file >> cost >> score;
file.close();
```

☆ ~~Reading~~

☆ Reading Class Objects ~~into a File~~ from a File.

```
→ #include <fstream>
#include <iostream>
#include <string>
class Employee
{     public:    //or private
      string  Name;
      ~~$~~int   Emp_ID;
      int  Salary;
};
int main()
{
      Employee  emp;
      ifstream file;
      file.open ("Employee.txt", ios::in);
      file.seekg(0);
      file.read ((char*) & Emp, sizeof(emp));
      cout << emp.Name << emp.Emp_ID << emp.Salary;
      file.close();
      return 0;
}
```

☆ Writing Class Objects into a File.

```
→ #include <fstream>
#include <iostream>
#include <string>
```

```
class Employee
{   public:
    string Name;
    ~~EE~~ int ID;
     int salary;
};
int main()
{
    Employee emp;
    emp.Name = "John";
    emp.ID = 1001;
    emp.Salary = 110000;
    ofstream file;
    file.open("Employee.txt", ios::app); //Append mode.
    file.write((char*)&emp, sizeof(emp));
    file.close();
    return 0;
}
```

★ <u>File Modes</u>.

1. <u>in</u>

→ Open for reading. It should be specified for input files.

2. <u>out</u>

→ Open for writing. It should be specified for output files.

3. <u>ate</u>

→ Seek to end of file upon original open.

4. <u>app</u>

→ Append mode.

5. <u>trunc</u>

→ Truncate file if already exists.

6. noreate
→ Open fails if file does not exist.
7. noreplace
→ Open fails if file already exists.
8. binary
→ Opens file as binary.
E
→ We can open files in different modes, which are as follows:
1. ios::app
→ It is useful for appending the content in an existing file.
2. ios::ate
→ It means that open the file and go to the end of the file.
3. ios::binary
→ Open the file in binary mode.
4. ios::in
→ Open the file only for reading purpose.
5. ios::out
→ Open the file writing into it.
6. ios::trunc
→ Open and truncate the file.
→ Delete all existing content from the file.
E

★ Reading and Writing Class Object in the File.
→ ~~Writing~~: Syntax:
```
fstream file;
file.open("Employee.txt", ios::in, ios::out);
file.write((char*)&emp, sizeof(emp)); //values are given to em
```
// to write data as string and the last byte is to be specified using: 'sizeof() operator'.

classmate

Date
Page

```
//Reading data from the file:
file.seekg(0);
file.read((char*)&emp, sizeof(emp));
```

★ Function	Meaning	Example
1. seekg()	Moves input pointer to a given position.	file.seekg(20); Moves file pointer by 20 bytes
2. seekp()	Moves output pointer to a given position.	file.seekp(20);
3. tellg()	Gets the current position of "get pointer".	file.tellg(20);
4. tellp()	Gets the current position of "put pointer".	file.tellp(20);

★ seekg() is related to read, seekp() is related to write.

★ If file is opened in output mode, we use seekp() and tellp() functions.

★ If file is opened in ~~ou~~ input mode, we use seekg() and tellg() functions.

★ ~~E int main()~~
{
~~ofstream file;~~

★ Ran

→ It uses:

1. seekp() {To be used in write mode}

2. tellp() {To be used in write mode}.
3. seekg() {To be used in read mode}.
4. tellg() {To be used in read mode}.

→ For ~~accep~~ accessing the position of a pointer / setting the position of pointer, seekg() and seekp() tellg() and tellp() are used.

→
```
#include <iostream>
#include <~~fst~~stream>
using namespace std;
int main()
{
        ofstream file;
        file.open("Sample.txt");
        cout << file.tellp();      //returns 0.
        file<<"Hello World";
        cout << file.tellp();     //returns 11.
        file.seekp(-5, ios::end);  // file.seekp(6, ios::beg);
        file<<"SYComp";       // File contains "Hello SYComp".
        // If file<<"all"; is written, file would contain
        // "Hello allld"
        file.close();
        ifstream file1;
        file1.open("Sample.txt");
        ~~file1.seekg(6, ios~~
       /*file1.tellg();*/  cout<<file1.tellg();
        file1.seekg(6, ios::beg);
        cout << ~~te~~ file1.tellg();
        char ch;
        while(!file1.eof())     //end of file
        {
            file1.get(ch);
        cout<< ch;             //prints the whole file letter-by-
    }                          //letter.
    file1.close();
    return 0;
}
```

☆
```
void search()
{
    fstream file;     Sample obj;   cout<<"Roll: "; int r; cin>>r;
    file.open("Sample.txt", ios::in | ios::out | ios::binary);
    while(file.read((char *)&obj, sizeof(obj));
    {
        if(r== obj.roll)
        {
            cin>> obj.clas;
            cin>> obj.marks;
            file.write((char *)&obj, sizeof(obj));
        }
    }
    file.close();
}
```

★ Offset

→ seekg() is a function that allows you to seek an arbitrary position in a file.

→ It is defined for istream class.

→ It is used to set the position of the next character to be extracted from the input stream from a given file.

→ Syntax: seekg (streamoff offset, ios_base::seekdir dir);
OR seekg (streampos position);
position: New position in the stream buffer.

offset: Integer value of type streamoff representing the offset in the stream's buffer.

dir: Seeking direction. Takes any of the following value:

- ios_base::beg : From the beginning.
- ios_base::cur : From the current position.
- ios_base::end : From the end.

☆ Index Sequential Files

→ A file created with the help of C++ standard library functions does not impose any structure on how the data is to be persisted.

→ In index sequential files, the records are stored and written to the file sequentially and retrieved or read from the file in the same manner.

→
```
while (file.read((char*) &obj, sizeof(obj)))
{    if(id == obj.ID)
     {
          pos = obj.position;
          break;
     }
}
int offset = pos * sizeof (obj);
file.seekp(offset);
~~obj.ID~~ obj.ID = id;
obj.salary = new_salary;
file.write((char*) &obj, sizeof(obj));
```