Input / Output with Files

In C++:

ofstream : Stream class to write on files

ifstream : Stream class to read from files

fstream : Stream class to both read and write from/to files.

Example

```
#include <iostream>
#include <fstream>

void main()
{
    ofstream myfile;
    myfile.open("example.txt");
    myfile << "writing this to a file.\n";
    myfile.close();
}
```

ios::ate set the initial position at the end of the file.

ios::trunc if the file opened for output operations already existed before its previous content is deleted an replaced by the new one.

- open file

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

ios : a built-in class for input/output stream

ios::out : open for output operations

ios::app : all output operations are performed at the end of the file.

ios::binary : open in binary mode

To check if a file stream was sucessfully openned.

```
if (myfile.is_open())
{
    ...

}
```

myfile.is_open() → 'true' means it was openned successfully

```
Void main()
{
    long  pos1, pos2, pos3;

    ifstream myfile ("example.txt");

    pos1 = myfile.tellg();

    myfile << ....;

    pos2 = myfile.tellg();

    myfile << ... ;

    pos3 = myfile.tellg();

    myfile.seekg (pos1);

     // more reading from the beginning.
      ...

     myfile.close()
 }
```

– Binary files.

Don't use <<, >>, getline() for binary file.

Else : write (memory_block $\xrightarrow{}$ char *c , size $\xrightarrow{}$ ifstream :: pos_type);

read (memory_block , size);

Example.

```
#include  <iostream>
#include  <fstream>
ifstream:: pos_type  size;
char * memblock;
void main()
{
 ifostream file ("example.txt", ios::in | ios::binary | ios::ate);
  if ( file.is_open())
  {
     size = file.tellg ();
     memblock = new char [size];
     file.seekg ( 0, ios::beg);
     file.read (memblock, size);
 } } file.close();
```

↗ at the end of the file.

- Close a file.

  myfile.close();

- in.ignore(10, ' ');

  Ignore up to 10 characters until first space is found.

- Text file

  By default, we are handling text files without including ios::binary flag.

- Checking State flags

  fail(). return true if read/write operation fails.

  eof(): return true if a file open for reading has reached the end.

  good(): return false if something becomes wrong.

  clear(): reset the state flags checked by any of the above functions

```
Void main()
{ char buffer [80];
  fstream myfile;
  myfile.open("test.txt", ios::in);
  myfile << "test";
  if(myfile.fail())
  { cout << "Error writing to test.txt\n";
  } myfile.clear();
  myfile.getline (buffer, 80);
  cout << buffer << endl;
}
```

- get and put stream pointers.

All i/o stream objects have, at least, one internal stream pointer.

ifstream, has a pointer known as the get pointer that points to the elements to be read in the next input operation

ofstream, has " put " " the location where the next element has to be written.

fstream contains both pointers.

<u>Member functions</u> <u>that manipulate these pointers</u>

- tellg() → returns the current position of the get pointer : integer
- tellp() → " " put " : integer
- seekg(~~position~~) → change the position of the get pointer to the absolute posit 'position' (counting from the beginning of the file).

  integer type
- seekp(position) → " " " put " " 'position'.
- seekg(offset, direction)
- seekp(offset, direction)

  offset: integer → offset value.

  direction
  - ios::beg . offset counted from the beginning o the stream
  - ios::cur " " the current position
  - ios::end " " end

How to delete a file?

In C++, there is no function to do this. You need to use C function.

```
#include <cstdio>
#include <cstdlib>

std::remove ("abc. txt");
```

How to detect the existence of a file?

Way 1.   fast

```
#include <sys/stat.h>
#include <sys/types.h>

bool fexist (char * filename)
{
  struct stat buffer;
  int j = stat (filename, & buffer);
  if ( y == -1)
      return false;
  else
      return true;
}
```