C++

File handling

Similarities with console I/O

- Same << and >> stream insertion and extraction operations are used to manipulate data in text file
- In case of console I/O, the identifiers cout and cin are built in C++ and hence readily available
- For files we have to
  - Include header fstream
  - declare a file stream variable
  - Open a file, before reading/writing data
  - Instructions for read or write data, as required
  - After reading/writing data, we have to close the file.

File writing example

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

using namespace std;

int main()
{
        string numfile = "numbers.txt";
        ofstream ofile;
        ofile.open(numfile.c_str());
        //if (!ofile)
        int j=0
        while(j<10)
        {
                ofile << j << " " << rand()%850+150 << endl;
                j = j + 1;
        }

        ofile.close();

   return 0;
}
```

ofstream
File opened explicitly in next statement

File reading example

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

using namespace std;

int main()
{
    string numfile = "nums.txt";
    ifstream ifile(numfile.c_str());
    //if (!ifile)
    int a[10];
    for(int j=0; j<10; j++)
    {
        ifile >> a[j];
        cout << a[j] << ",";
    }
    cout << '\b' << " " << endl;
    ifile.close();

    return 0;
}
```

ifstream
Opening file implicitly

nums.txt

```
1 123
2 222
3 333
4 444
5 546
6 666
7 703
8 852
9 990
10 10
```

Reading/writing in different files

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

using namespace std;

int main()
{
            string infile;
            cout << "Enter input file name ";
            cin >> infile;

            string outfile;
            cout << "Enter output file name ";
            cin >> outfile;

            ifstream ifile(infile.c_str());
            //if (!ifile)
            ofstream ofile(outfile.c_str());
            //if (!ofile)
            char c;
            ifile >> c;
            //c = ifile.get();
            while(!ifile.eof())
            {
                        if(c >= 'a' && c <= 'z')
                        {
                                    c = c-'a'+'A';
                        }
                        ofile << c;
                        ifile >> c;
                        //c = ifile.get();
            }

            ifile.close();
            ofile.close();

    return 0;
}
```

Code to make all small letter from input file to capital letter in output file, leaving all other character as they were

C++ string vs. C style string

- In C++, string is a data type used to store strings. Full functionality may be used by including string header (#include <string>), though limited functionality is available in iostream header.
  - string name = “Abdullah”;
- In C, not C++, string are nothing but array of char datatype. In char array, stored string starts from 0 index of array and finished at a location where NULL character \0 is placed (automatically by following statement).
  - char name[] = “Abdullah”;
- Some of C++ functions requires string’s data in the form of C style strings, e.g. **open** function for file variables.
- From a string variable named str, we can have equivalent C style sting as str.c_str(). And a C style string named address can be converted into string type using type casting function string as string(address)

What is a file?

- A file is a sequence of bytes stored on storage media (disk, cd, flash drive, etc.). Every file in one folder has a unique name.
- Files are two types:
  - **Text files:** Contains formatted data, which is human readable, consists of alphabets, digits, punctuations, special characters, spaces, and new lines characters. Using any of commonly available text editors, one can view/edit data stored in text files.
  - **Binary files:** Contains application/software specific data. Only the specific software can use/display/modify data in binary files. E.g. Pictures stored on computer requires image viewing software to view them and image editing software is required to edit them. When these files are open in text editors, funny character are displayed and in general, humans are unable to read/understand them.

File structure/format

- File are used to store data, and reading/writing files generally requires no user interaction during the execution of the program.
- Data in the files generally has some structure/format. Which may be described in file itself in header area of file or in some other file or implicitly known to the programmer.
- Reading and writing data in files should strictly follow the structure/format of the file, otherwise later on file reading can results in errors.
- Some of the file related processing may be done without knowing the structure, e.g. counting lines in the file, counting characters in the file and capitalize small alphabets, etc.

BMP File Format
The 54 byte BMP header

offset	size	description
0	2	signature, must be 4D42 hex
2	4	size of BMP file in bytes (unreliable)
6	2	reserved, must be zero
8	2	reserved, must be zero
10	4	offset to start of image data in bytes
14	4	size of BITMAPINFOHEADER structure, must be 40
18	4	image width in pixels
22	4	image height in pixels
26	2	number of planes in the image, must be 1
28	2	number of bits per pixel (1, 4, 8, or 24)
30	4	compression type (0=none, 1=RLE-8, 2=RLE-4)
34	4	size of image data in bytes (including padding)
38	4	horizontal resolution in pixels per meter (unreliable)
42	4	vertical resolution in pixels per meter (unreliable)
46	4	number of colors in image, or zero
50	4	number of important colors, or zero

Well-known file formats

- There are variety of files in a computer system. Some of them, along with name of software that may manipulate them are mentioned below.
  - PDF: not possible to open in image viewer or text editor, only software recognize PDF file structure/format may view/edit it.
  - JPG: not possible to open in text editors, only software recognize JPG file structure/format may view/edit it. There are dozens of such software.
  - DOC or DOCX: generally Microsoft word is used to edit/view/print it.
  - Many examples, XLSX, PPTX, MP3, MP4, etc.
  - EXE: these files are created by compilers and operating system loads them to memory and executes them.
- Software capable recognizing structure of a file type can manipulate it.

Same file, different views

A file (without any change in stored bytes) may be understandable by different software and they treated data differently. E.g., A PPM (Portable Pixel Map) have following views using a Notepad++ and IrfanView software.

```
P2
# Shows the word "FEEP" (example from Netpbm man page on PGM)
24 7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Example (magnified)

```
P3
3 2
255
# The part above is the header
# "P3" means this is a RGB color image in ASCII
# "3 2" is the width and height of the image in pixels
# "255" is the maximum value for each color
# The part below is image data: RGB triplets
255   0   0     0 255   0     0   0 255
255 255   0   255 255 255     0   0   0
```

Image
(magnified)

https://en.wikipedia.org/wiki/Netpbm_format

File handling functions

- Every **iostream** functionality may be used with files, some commonly used functions for data manipulation are:
  - put() and get(), getline(), ignore(), peek(), etc.
  - io manipulators like endl, setw(), etc.
  - read() and write() for binary data.
- Besides data manipulation, some of the functions are available for testing the state of the stream, e.g.,
  - good(), is_open(), eof(), clear(), etc.

More file handling functions

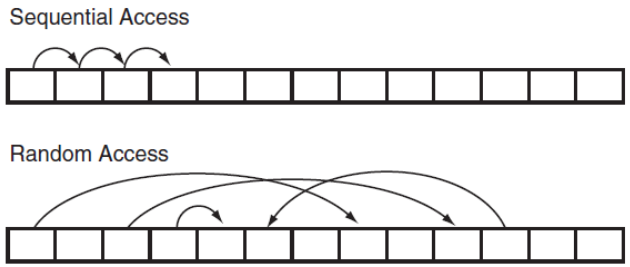
File Access Flag	Meaning
`ios::app`	Append mode. If the file already exists, its contents are preserved and all output is written to the end of the file. By default, this flag causes the file to be created if it does not exist.
`ios::ate`	If the file already exists, the program goes directly to the end of it. Output may be written anywhere in the file.
`ios::binary`	Binary mode. When a file is opened in binary mode, data are written to or read from it in pure binary format. (The default mode is text.)
`ios::in`	Input mode. Data will be read from the file. If the file does not exist, it will not be created and the open function will fail.
`ios::out`	Output mode. Data will be written to the file. By default, the file's contents will be deleted if it already exists.
`ios::trunc`	If the file already exists, its contents will be deleted (truncated). This is the default mode used by `ios::out`.

File open modes

If more than one mode/flag is required, character | has to be in between

e.g. ios::in | ios::out | ios::ate

Random Access



Statement	How It Affects the Read/Write Position
`file.seekp(32L, ios::beg);`	Sets the write position to the 33rd byte (byte 32) from the beginning of the file.
`file.seekp(-10L, ios::end);`	Sets the write position to the 10th byte from the end of the file.
`file.seekp(120L, ios::cur);`	Sets the write position to the 121st byte (byte 120) from the current position.
`file.seekg(2L, ios::beg);`	Sets the read position to the 3rd byte (byte 2) from the beginning of the file.
`file.seekg(-100L, ios::end);`	Sets the read position to the 100th byte from the end of the file.
`file.seekg(40L, ios::cur);`	Sets the read position to the 41st byte (byte 40) from the current position.
`file.seekg(0L, ios::end);`	Sets the read position to the end of the file.

To be continued . . .