Advanced Programming in C++

فرشاد حکیم پور

Streams

- Streams are artifacts that are used for data input or output
- I/O occurs in sequence of bytes
- Devices such as console, printer, keyboard, disk drives, network devices, etc. communicate with system using a flow of bytes
- "std::cin" is an example of input stream
- "std::cout" is an example of output stream

Streams

- "std::cin" is a "istream" and "std::out" is a "ostream"
- "<<" and ">>" are operators (or methods) defined on "ostream" and "istream", respectively
- There are other methods defined on "istream" and "ostream"

Stream Methods

- "int get()" method for "istream"
  - Retrieves a byte from the stream and returns its value in integer form
  - E.g. `int I = cin.get();`
- "get(char&)" method for "istream"
  - Retrives a byte from the stream and put it where the input address is referring to
  - E.g. `cin.get(&c);`

Stream Methods

- "get(char* sPtr, int n)" method for "istream"
  - Retrives n-1 byte from the stream and put them where the input address is referring to
  - sPts should be a pointer to an array of type "char"
  - The n^{th} position is set to null
  - Stops reading if end of line (\n) is reached
  - E.g. `cin.get(s, 5);`

Stream Methods

- "read(char* s, int n)" method for "istream"
  - Retrives a block of n bytes from the stream and put them where the input address is referring to
  - sPts should be a pointer to an array of type "char"
  - E.g. `cin.read(s, 4);`
- "eof()" method for "istream"
  - Returns true if the end of file is reached

Stream Methods

- "read(char* s, int n)" method for "istream"
  - Retrives a block of n bytes from the stream and put them where the input address is referring to
  - sPts should be a pointer to an array of type "char"
  - E.g. `cin.read(s, 4);`

Stream Methods

- "put(char)" method for "ostream"
  - sets a byte from the output stream
  - E.g. `cout.put("A");`
- "write(char* s, int n)" method for "ostream"
  - Writes a block of n bytes on the output stream
  - sPts should be a pointer to an array of type "char"
  - E.g. `cout.write(s, 60);`

Stream Methods

- "open(char[] fn)" constructor method for both "istream" and "ostream"
  - Opens a stream for input or output
  - Some streams such as "ifstream" and "ofstream" require opening
- "open(char[] fn, ios::<mode>)" constructor method for both "istream" and "ostream"
  - Opens a stream in a specified mode
  - E.g. `inFile("myfile.dat", ios::binary);`

Stream Methods

- "eof()" method for both "istream" and "ostream"
  - Returns true if the end of file is reached
  - E.g. `while (!cin.eof()) . . .`
- "close()" method for both "istream" and "ostream"
  - Disconnects from the device
  - For output streams it writes all the buffer before disconnecting
  - E.g. `outFile.close();`

- See the following for a reference to C++ standard library descriptions:
  - http://www.cplusplus.com/reference/

Exercise

- Write a program that reads a Bitmap file and draws three histogram for three colors of blue, green and red.
- Deadline: End of Wednesday 27th Azar
- Send the program to: tamrin.ut@gmail.com
- **Subject line should contain:** 'EX3', Your name and you student ID number

```cpp
#include <cstdlib>
using std::exit;

#include <iostream>
using std::cout;
using std::endl;

#include <fstream>
using std::ifstream;

// This program reads the header of a Bitmap file and writes the related
// information on the console
int main()
{
    char *filename = "output.bmp";
    ifstream imageFile;
    imageFile.open(filename);
    if (!imageFile)
    {
      cout<<"System failed to open the file"<<endl;
      exit(1);
    }

    int i;
    char ch;
    imageFile.get(ch);
    cout << ch;
    imageFile.get(ch);
    cout << ch << endl;

    // DO NOT USE imageFile.get(s,4);
    // get method reads n-1 and fills the last char with null!
    char *s = (char*)&i;
    imageFile.read(s,4);  // could be done like: imageFile.read((char*)&i,4);
    cout<<"File Size: "<< i << endl;

    imageFile.ignore(4);

    imageFile.read(s,4);
    cout<<"Image data offset: "<< i << endl;

    imageFile.read(s,4);
    cout<<"Bitmap Header size: "<< i << endl;

    imageFile.read(s,4);
    cout<<"Image width: "<< i << endl;

    imageFile.read(s,4);
    cout<<"Image height: "<< i << endl;

    imageFile.ignore(2);

    cout<<"Bits per pixel: "<< imageFile.get() <<endl;
    imageFile.ignore(1);
    // Compression type
    imageFile.ignore(4);

    imageFile.read(s,4);
    cout<<"Image data size: "<< i << endl;
    // Horizontal resolution
```

```
    imageFile.ignore(4);
    // Vertical resolution
    imageFile.ignore(4);
    // Number of colors in the pallet
    imageFile.ignore(4);
    // Important colors
    imageFile.ignore(4);

    imageFile.close();

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

```cpp
#include <cstdlib>
using std::exit;

#include <iostream>
using std::cout;
using std::endl;
using std::ios;

#include <fstream>
using std::ifstream;
using std::ofstream;

// This program reads a BitmapFile and writes it on a new output file
// after it's blue color is reduced (making it more yellowish)
int main(int argc, char *argv[])
{
    char *inFilename = "Bluehills.bmp";
    char *outFilename = "Yellowsky.bmp";
    ifstream inFile;
    inFile.open(inFilename, ios::binary);

    // Checking if both files are opened correctly
    if (!inFile)
    {
      cout<<"System failed to open: " << *inFilename <<endl;
      exit(1);
    }

    ofstream outFile;
    outFile.open(outFilename, ios::binary);
    if (!outFile)
    {
      cout<<"System failed to open: " << *outFilename <<endl;
      exit(1);
    }

    int i;
    char b,m;
    // Reading the Bitmap signature
    inFile.get(b);
    inFile.get(m);
    cout << b << m << endl;
    // Quit if the signature is not right
    if (b!='B' || m!='M')
    {
        cout << "the input file:" << *inFilename << "is not a Bitmap" << endl;
        exit(1);
    }
    outFile.put(b);
    outFile.put(m);

    // Copy the file size (four bytes)
    char *s = (char*)&i;
    inFile.read(s,4);  // imageFile.read((char*)&i,4);
    cout<<"File Size: "<< i << endl;
    outFile.write(s,4);

    // Copy four bytes of zeros
    inFile.read(s,4);
    outFile.write(s,4);
```

```
    // Copy the image data offset
    inFile.read(s,4);
    cout<<"Image data offset: "<< i << endl;
    outFile.write(s,4);

    // Copy the rest of the header to the point where data starts
    s = new char[200];
    inFile.read(s,i-14);
    outFile.write(s,i-14);
    // Giving back the 200 bytes we got by new
    delete s;

    i = inFile.get();
    while (!inFile.eof())
    {
        // intensity of the blue light
        outFile.put(static_cast<int>(i*0.8));
        i = inFile.get();
        // intensity of the green light
        outFile.put(i);
        i = inFile.get();
        // intensity of the red light
        outFile.put(i);
        i = inFile.get();
    }

    inFile.close();
    outFile.close();

    system("PAUSE");
    return EXIT_SUCCESS;
}
```