Chapter 15

File Input and Output

15.1 Opening and Closing a File

15.2 Reading and Writing Text Files

15.3 Detecting End-Of-File (EOF)

15.4 References

15.1 Opening and Closing a File

In C++, a file is opened by linking it to one of three streams:

```
include <fstream.h>

ifstream in;       // input stream
ofstream out;      // output stream
fstream inAndOut; // input and output
```

The following code opens an input file "test.dat" for reading data from this file:

```
ifstream inFile;
inFile.open("test.dat");
```

After reading the data, the file has to be closed: **`inFile.close();`**

If open() fails, i.e., the file does not exist or has already been opened, the stream will evaluate to false:

```
if(!inFile) cout << "Cannot open file.";
```

You can also check whether the file open command succeeded as follows:

```
if(!inFile.is_open()) {
  cout << "File is not open.";
  // ... }
```

or

```
if(inFile.rdstate() != 0)
{
  cout << "Unable to open file.";
}
```

15.2 Reading and Writing Text Files

The easiest way to read from or write to a text file is to use the << and >> operators.

```
int main()
{
  ofstream outFile("test.dat");
  if(!outFile) {
    cout << "Cannot open file.\n";
    return 1; }

  outFile << 10 << " " << 123.23 << "\n";
  outFile << "A short text.";

  outFile.close();
  return 0;
}
```

```
int main()
{
  char ch, str[80]; int i; float f;

  ifstream inFile("test.dat");
  if(!inFile) {
    cout << "Open error."; return 1; }

  inFile >> i;
  inFile >> f;
  inFile >> ch;
  inFile >> str;

  cout << i << " " << f << " " << ch;
  cout << str;

  inFile.close(); return 0; }
```

Another function that performs input from a file is **`getline()`**, which has the following prototypes:

- **istream &getline(char **buf*, streamsize *num*);**

  Reads characters into the array pointed to by *buf* until either

  - *num*-1 characters have been read,
  - a newline character has been found,
  - or the end of the file has been encountered.

  The array pointed to by *buf* will be null-terminated by **`getline()`**.

  If the newline character ('\n') is encountered in the input stream, it is extracted, but is not put into *buf*.

- **istream &getline(char **buf*, streamsize *num*, char *delim*);**

  Reads characters ... until either

  - *num*-1 characters have been read,
  - the character specified by *delim* has been found,
  - or the end of the file has been encountered.

Reading strings line by line from a file:

```
const int STR_LENGTH = 256;
fstream inFile;
char fileName[STR_LENGTH];

void initLine (char *line)
{
  for (int i=0; i < 81; i++) line[i]=0;
}

void readLine (char *line)
{
  initLine(line);

  inFile.getline(line, 80, '\n');
}
```

```
struct address {
  char name[80];
  char phone[20];
};

void main()
{
  address entry;

  cout << "Enter filename: ";

  cin.getline(fileName, STR_LENGTH, '\n');

  inFile.open(fileName, ios:in);

  // Read in data ...
```

```
    // Read in data ...

    readLine(entry.name);
    readLine(entry.phone);

    cout << "Name: " << entry.name << endl;
    cout << "Phone: " << entry.phone;

    inFile.close();
}
```

15.3 Detecting End-Of-File (EOF)

```
#include <iostream.h>
#include <fstream.h>

int main(int argc, char *argv[])
{ char ch;

  ifstream in(argv[1], ios::in);
  if(!in) {
    cout << "Cannot open file."; return 1;}

  while (!in.eof()) {
    in.get(ch);
    if (!in.eof()) cout << ch;}
  in.close();
  return 0; }
```

15.4 References

- H. Schildt, C++ from the Ground Up, McGraw-Hill, Berkeley, CA, 1998. Chapter 18.