C++ Files and Streams

Presented By

K. APPASAMY,MCA;M.Phil

Assistant Professor,

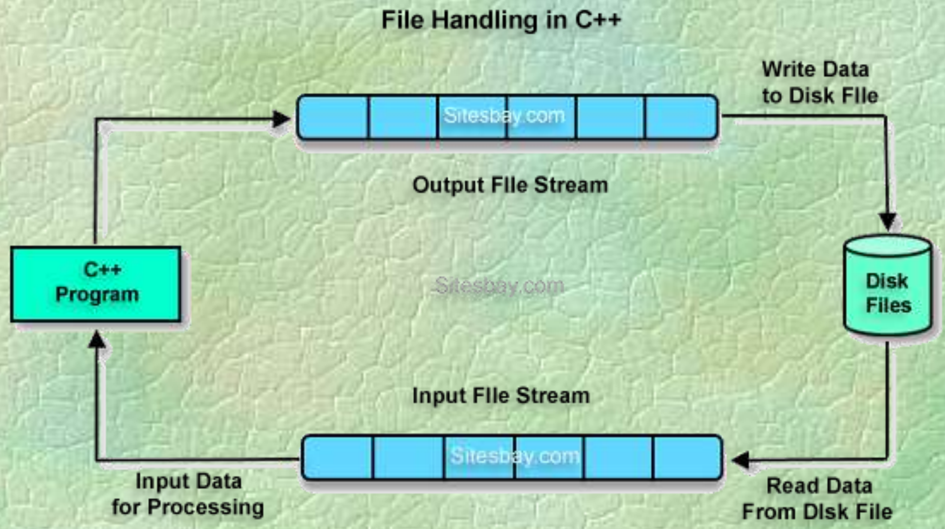
Department of BCA & M.Sc[NT & IT],

ST. JOHN’S COLLEGE,

PALAYAMKOTTAI

C++ Files and Streams

- In C++ Files as a sequence of bytes.
- Each file ends with an ***end-of-file*** marker.
- When a file is ***opened***, an object is created and a stream is associated with the object.
- To perform file processing in C++, the header files **<iostream.h>** and **<fstream.h**> must be included.
- <fstream.> includes <ifstream> and <ofstream>



File Handling Concept is used for store a data permanently in computer. Using file easily transfer data from one computer to another. Sitesbay

Creating a sequential file

```
// Fig. 14.4: fig14_04.cpp  D&D p.708
// Create a sequential file
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
int main()
{
   // ofstream constructor opens file
   ofstream outClientFile( "clients.dat", ios::out );

   if ( !outClientFile ) {  // overloaded ! operator
      cerr << "File could not be opened" << endl;
      exit( 1 );    // prototype in stdlib.h
   }
```

Sequential file

```
cout << "Enter the account, name, and balance.\n"
        << "Enter end-of-file to end input.\n? ";
    int account;
   char name[ 30 ];
   float balance;

   while ( cin >> account >> name >> balance ) {
      outClientFile << account << ' ' << name
                    << ' ' << balance << '\n';
      cout << "? ";
   }

   return 0;  // ofstream destructor closes file
}
```

How to open a file in C++ ?

```
Ofstream outClientFile(“clients.dat”, ios:out)
```

OR

```
Ofstream outClientFile;
outClientFile.open(“clients.dat”, ios:out)
```

File Open Modes

ios:: app - (append) write all output to the end of file

ios:: ate - data can be written anywhere in the file

ios:: binary - read/write data in binary format

ios:: in - (input) open a file for input

ios::out - (output) open afile for output

ios: trunc -(truncate) discard the files' contents if
it exists

ios:nocreate - if the file does **NOT** exists, the open
operation fails

ios:noreplace - if the file exists, the open operation fails

How to close a file in C++?

The file is closed implicitly when a destructor for the corresponding object is called

OR

by using member function *close:*

```
outClientFile.close();
```

Reading and printing a sequential file

```
// Reading and printing a sequential file
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
#include <stdlib.h>
void outputLine( int, const char *, double );
int main()
{
   // ifstream constructor opens the file
   ifstream inClientFile( "clients.dat", ios::in );

   if ( !inClientFile ) {
      cerr << "File could not be opened\n";
      exit( 1 );
   }
int account;
   char name[ 30 ];
   double balance;

   cout << setiosflags( ios::left ) << setw( 10 ) << "Account"
        << setw( 13 ) << "Name" << "Balance\n";

   while ( inClientFile >> account >> name >> balance )
      outputLine( account, name, balance );

   return 0;  // ifstream destructor closes the file
}

void outputLine( int acct, const char *name, double bal )
{
   cout << setiosflags( ios::left ) << setw( 10 ) << acct
        << setw( 13 ) << name << setw( 7 ) << setprecision( 2 )
        << resetiosflags( ios::left )
        << setiosflags( ios::fixed | ios::showpoint )
        << bal << '\n';
}
```

File position pointer

<istream> and <ostream> classes provide member functions for repositioning the *file pointer* (the byte number of the next byte in the file to be read or to be written.)

These member functions are:

seekg (seek get) for istream class

seekp (seek put) for ostream class

Examples of moving a file pointer

inClientFile.seekg(0) - repositions the file get pointer to the beginning of the file

inClientFile.seekg(n, ios:beg) - repositions the file get pointer to the n-th byte of the file

inClientFile.seekg(m, ios:end) -repositions the file get pointer to the m-th byte from the end of file

nClientFile.seekg(0, ios:end) - repositions the file get pointer to the end of the file

The same operations can be performed with <ostream> function member *seekp*.

Member functions tellg() and tellp()

Member functions **tellg** and **tellp** are provided to return the current locations of the get and put pointers, respectively.

long location = inClientFile.tellg();

To move the pointer relative to the current location use ios:cur

inClientFile.seekg(n, ios:cur) - moves the file get pointer n bytes forward.

Updating a sequential file

Data that is formatted and written to a sequential file **cannot be modified easily** without the risk of destroying other data in the file.

If we want to modify a record of data, the new data may be longer than the old one and it could overwrite parts of the record following it.

Problems with sequential files

Sequential files are inappropriate for so-called “instant access” applications in which a particular record of information must be located immediately.

These applications include banking systems, point-of-sale systems, airline reservation systems, (or any data-base system.)

Random access files

Instant access is possible with random access files.

Individual records of a **random access file** can be accessed directly (and quickly) without searching many other records.

Example of a Program that Creates a Random Access File

```
#ifndef CLNTDATA_H
#define CLNTDATA_H
struct clientData {
   int accountNumber;
   char lastName[ 15 ];
   char firstName[ 10 ];
   float balance;
};
#endif
```

Creating a random access file

```
// Creating a randomly accessed file sequentially
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include "clntdata.h"
int main()
{
            ofstream outCredit( "credit1.dat", ios::out);
if ( !outCredit ) {
     cerr << "File could not be opened." << endl;
     exit( 1 );
   }
clientData blankClient = { 0, "", "", 0.0 };

for ( int i = 0; i < 100; i++ )
   outCredit.write
           (reinterpret_cast<const char *>( &blankClient ),
           sizeof( clientData ) );
   return 0;
}
```

<ostream> memebr function *write*

The <ostream> member function *write* outputs a fixed number of bytes beginning at a specific location in memory to the specific stream. When the stream is associated with a file, the data is written beginning at the location in the file specified by the “put” file pointer.

The *write* function expects a first argument of type *const char* *, hence we used the *reinterpret_cast* <*const char* *> to convert the address of the *blankClient* to a *const char* *. The second argument of *write* is an integer of type size_t specifying the number of bytes to written. *Thus* the *sizeof(clientData)*.

Writing data randomly to a random file

```
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include "clntdata.h"
int main()
{
   ofstream outCredit( "credit.dat", ios::ate );

   if ( !outCredit ) {
      cerr << "File could not be opened." << endl;
      exit( 1 );
   }
```

```
cout << "Enter account number "
     << "(1 to 100, 0 to end input)\n? ";

 clientData client;
 cin >> client.accountNumber;

 while ( client.accountNumber > 0 &&
         client.accountNumber <= 100 ) {
    cout << "Enter lastname, firstname, balance\n? ";
    cin >> client.lastName >> client.firstName
        >> client.balance;
outCredit.seekp( ( client.accountNumber - 1 ) *
                 sizeof( clientData ) );
    outCredit.write(
      reinterpret_cast<const char *>( &client ),
      sizeof( clientData ) );

    cout << "Enter account number\n? ";
    cin >> client.accountNumber;
  }

  return 0;
}
```

Reading data from a random file

```
#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <stdlib.h>
#include "clntdata.h"
void outputLine( ostream&, const clientData & );
int main()
{
   ifstream inCredit( "credit.dat", ios::in );
   if ( !inCredit ) {
      cerr << "File could not be opened." << endl;
      exit( 1 );
   }
```

```
cout << setiosflags( ios::left ) << setw( 10 ) << "Account"
      << setw( 16 ) << "Last Name" << setw( 11 )
      << "First Name" << resetiosflags( ios::left )
      << setw( 10 ) << "Balance" << endl;

  clientData client;

  inCredit.read( reinterpret_cast<char *>( &client ),
                 sizeof( clientData ) );
```

```
while ( inCredit && !inCredit.eof() ) {

      if ( client.accountNumber != 0 )
         outputLine( cout, client );

      inCredit.read( reinterpret_cast<char *>( &client ),
                     sizeof( clientData ) );
   }

   return 0;
}
```

```
void outputLine( ostream &output, const clientData &c )
{
   output << setiosflags( ios::left ) << setw( 10 )
          << c.accountNumber << setw( 16 ) << c.lastName
          << setw( 11 ) << c.firstName << setw( 10 )
          << setprecision( 2 ) << resetiosflags( ios::left )
          << setiosflags( ios::fixed | ios::showpoint )
          << c.balance << '\n';
}
```

The <istream> function *read*

```
inCredit.read (reinterpret_cast<char *>(&client),
                    sizeof(clientData));
```

The <istream> function inputs a specified (by sizeof(clientData)) number of bytes from the current position of the specified stream into an object.

Thank You