The National School of Artificial Intelligence
المدرسة الوطنية العليا للذكاء الاصطناعي

Data Structure & Algorithms 1

CHAPTER 7: FILE PROCESSING IN C++

Sep – Dec 2023

Introduction

- Storage of data
  - Arrays, variables are temporary
  - Files are permanent
    - Hard drive, Magnetic disk, optical disk, tapes, etc.

- In this chapter
  - Create, update, process files
  - Sequential access
  - Formatted and raw processing

Data Hierarchy

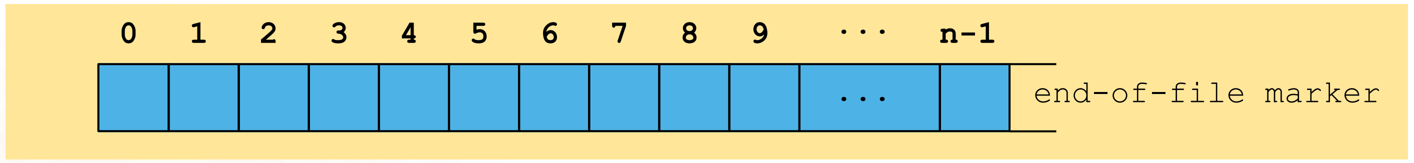
- From smallest to largest
  - **Bit (binary digit)**
    - 1 or 0
    - Everything in computer ultimately represented as bits
    - Cumbersome for humans to use
    - Character set
      - Digits, letters, symbols used to represent data
      - Every character represented by 1's and 0’s
  - **Byte: 8 bits**
    - Can store a character (**char**)

Data Hierarchy

- **Field:** group of characters with some meaning
  - Your name
- **Record**: group of related fields
  - **struct** or **class** in C++
  - In library catalog system, a record could be:
    - title, author, publication date, and availability
  - Each field associated with same book
  - Record key: field used to uniquely identify record
- **File**: group of related records
  - Books' records for entire library
  - Sequential file: records stored by key
- **Database**: group of related files
  - Payroll, accounts-receivable, inventory…

Files and Streams

- C++ views file as sequence of bytes
  - Ends with *end-of-file* marker (`EOF`)



- When file opened
  - Object created, stream associated with it
  - **`cin`**, **`cout`**, etc. created when **`<iostream>`** included
    - Communication between program and file/device

Files and Streams

To perform file processing in C++, you'll need to include the necessary header files:

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

C++ provides class templates for file input, output, and input/output operations:

- **basic_ifstream**: For input operations (file stream specialized for reading).
- **basic_ofstream**: For output operations (file stream specialized for writing).
- **basic_fstream**: For input/output operations (file stream supporting both reading and writing).

Files and Streams

Opening Files

- **Create Objects from Templates**:

Use class templates: `ifstream` for input, `ofstream` for output, and `fstream` for both.

- **Derive from Stream Classes:**

Utilize classes derived from stream templates to interact with files.

- **Stream Methods :**

Leverage stream methods, such as `put(), get(), peek(),` etc.

By following these steps, you can efficiently open files, create corresponding objects, and employ various stream methods for effective file processing.

Creating a Sequential-Access File

- ▶ C++ imposes no structure on file
  - Concept of "record" must be implemented by programmer
- ▶ To open file, create objects
  - Creates "line of communication" from object to file
  - Constructors take *file name* and *file-open mode*

```
ofstream outClientFile( "filename", fileOpenMode );
```

  - To attach a file later

```
Ofstream outClientFile;
outClientFile.open("filename", fileOpenMode);
```

Creating a Sequential-Access File

- File-open modes

Mode	Description
ios::app	Write all output to the end of the file.
ios::ate	Open a file for output and move to the end of the file (normally used to append data to a file). Data can be written anywhere in the file.
ios::in	Open a file for input.
ios::out	Open a file for output.
ios::trunc	Discard the file's contents if it exists (this is also the default action for **ios::out**)
ios::binary	Open a file for binary (i.e., non-text) input or output.

  - **ofstream** opened for output by default
    - **`ofstream outClientFile( "clients.dat", ios::out);`**
    - **`ofstream outClientFile( "clients.dat");`**

Creating a Sequential-Access File

- Operations
  - Overloaded **operator!**
    - **`!outClientFile`**
    - Returns nonzero (true) if **badbit** or **failbit** set
      - Opened non-existent file for reading, wrong permissions
  - Overloaded **`operator void*`**
    - Converts stream object to pointer
    - **0** when when **failbit** or **badbit** set, otherwise nonzero
      - **failbit** set when EOF found
    - **`while ( cin >> myVariable )`**
      - Loops until EOF

Creating a Sequential-Access File

- Operations
  - Writing to file (just like **cout**)
    - **`outClientFile << myVariable`**
  - Closing file
    - **`outClientFile.close()`**
    - Automatically closed when destructor called (will see the concept of destruction in OOP module)

Creating a Sequential-Access File

```cpp

// Create a sequential file.
#include <iostream>

using std::cout;
using std::cin;
using std::ios;
using std::cerr;
using std::endl;

 #include <fstream>

 using std::ofstream;

 #include <cstdlib>  // exit prototype

 int main()
 {
    // ofstream constructor opens file
    ofstream outClientFile( "clients.dat", ios::out );

    // exit program if unable to create file
    if ( !outClientFile ) { // overloaded ! operator
       cerr << "File could not be opened" << endl;
       exit( 1 );

    } // end if
```

Notice the the header files required for file I/O.

`ofstream` object created and used to open file **`"clients.dat"`**. If the file does not exist, it is created.

! operator used to test if the file opened properly.

Creating a Sequential-Access File

```cpp

   cout << "Enter the account, name, and balance." << endl
        << "Enter end-of-file to end input.\n? ";

   int account;
   char name[ 30 ];
   double balance;

   // read account, name and balance from cin, then place in file
   while ( cin >> account >> name >> balance ) {
      outClientFile << account << ' ' << name << ' ' << balance
                    << endl;
      cout << "? ";

   } // end while

   return 0;  // ofstream destructor closes file

} // end main
```

cin is implicitly converted to a pointer. When EOF is encountered, it returns 0 and the loop stops.

Write data to file like a regular stream.

File closed when destructor called for object. Can be explicitly closed with **close()**.

Creating a Sequential-Access File

```
Enter the account, name, and balance.
Enter end-of-file to end input.
? 100 Jones 24.98
? 200 Doe 345.67
? 300 White 0.00
? 400 Stone -42.16
? 500 Rich 224.62
? ^Z
```

Reading Data from Sequential-Access File

- Reading files
  - `ifstream inClientFile( "filename", ios::in );`
  - Overloaded **`!`**
    - **`!inClientFile`** tests if file was opened properly
  - **`operator void*`** converts to pointer
    - **`while (inClientFile >> myVariable)`**
    - Stops when EOF found (gets value **`0`**)

Reading Data from Sequential-Access File

```
// Reading and printing a sequential file.
#include <iostream>

using std::cout;
using std::cin;
using std::ios;
using std::cerr;
using std::endl;
using std::left;
using std::right;
using std::fixed;
using std::showpoint;

#include <fstream>

using std::ifstream;

#include <iomanip>

using std::setw;
using std::setprecision;

#include <cstdlib> // exit prototype

void outputLine( int, const char * const, double );
```

Reading Data from Sequential-Access File

```
int main()
{
   // ifstream constructor opens the file
   ifstream inClientFile( "clients.dat", ios::in );

   // exit program if ifstream could not open file
   if ( !inClientFile ) {
      cerr << "File could not be opened" << endl;
      exit( 1 );

   } // end if

   int account;
   char name[ 30 ];
   double balance;

   cout << left << setw( 10 ) << "Account" << setw( 13 )
        << "Name" << "Balance" << endl << fixed << showpoint;

   // display each record in file
   while ( inClientFile >> account >> name >> balance )
      outputLine( account, name, balance );

   return 0; // ifstream destructor closes the file

} // end main
```

Open and test file for input.

Read from file until EOF found.

Reading Data from Sequential-Access File

```cpp

// display single record from file
void outputLine( int account, const char * const name,
   double balance )
{
   cout << left << setw( 10 ) << account << setw( 13 ) << name
        << setw( 7 ) << setprecision( 2 ) << right << balance
        << endl;

} // end function outputLine
```

```
Account   Name          Balance
100       Jones           24.98
200       Doe            345.67
300       White            0.00
400       Stone          -42.16
500       Rich           224.62
```

Reading Data from Sequential-Access File

- File position pointers
  - Number of next byte to read/write
  - Functions to reposition pointer
    - **seekg** (seek get for **istream** class)
    - **seekp** (seek put for **ostream** class)
    - Classes have "get" and "put" pointers
  - **seekg** and **seekp** take *offset* and *direction*
    - Offset: number of bytes relative to direction
    - Direction (**ios::beg** default)
      - **ios::beg** - relative to beginning of stream
      - **ios::cur** - relative to current position
      - **ios::end** - relative to end

Reading Data from Sequential-Access File

- Examples
  - **`fileObject.seekg(0)`**
    - Goes to front of file (location **`0`**) because **`ios::beg`** is default
  - **`fileObject.seekg(n)`**
    - Goes to nth byte from beginning
  - **`fileObject.seekg(n, ios::cur)`**
    - Goes n bytes forward
  - **`fileObject.seekg(y, ios::end)`**
    - Goes y bytes back from end
  - **`fileObject.seekg(0, ios::cur)`**
    - Goes to last byte
  - **`seekp`** similar

Reading Data from Sequential-Access File

- To find pointer location
  - **`tellg`** and **`tellp`**
  - **`location = fileObject.tellg()`**
- Upcoming example
  - Credit manager program
  - List accounts with zero balance, credit, and debit

Reading Data from Sequential-Access File

```
// Credit-inquiry program.
#include <iostream>

using std::cout;
using std::cin;
using std::ios;
using std::cerr;
using std::endl;
using std::fixed;
using std::showpoint;
using std::left;
using std::right;

#include <fstream>

using std::ifstream;

#include <iomanip>

using std::setw;
using std::setprecision;

#include <cstdlib>
```

Reading Data from Sequential-Access File

```cpp
enum RequestType { ZERO_BALANCE = 1, CREDIT_BALANCE,
   DEBIT_BALANCE, END };
int getRequest();
bool shouldDisplay( int, double );
void outputLine( int, const char * const, double );

int main()
{
   // ifstream constructor opens the file
   ifstream inClientFile( "clients.dat", ios::in );

   // exit program if ifstream could not open file
   if ( !inClientFile ) {
      cerr << "File could not be opened" << endl;
      exit( 1 );

   } // end if

   int request;
   int account;
   char name[ 30 ];
   double balance;

   // get user's request (e.g., zero, credit or debit balance)
   request = getRequest();
```

Reading Data from Sequential-Access File

```
   // process user's request
   while ( request != END ) {

      switch ( request ) {

         case ZERO_BALANCE:
            cout << "\nAccounts with zero balances:\n";
            break;

         case CREDIT_BALANCE:
            cout << "\nAccounts with credit balances:\n";
            break;

         case DEBIT_BALANCE:
            cout << "\nAccounts with debit balances:\n";
            break;

      } // end switch

```

Reading Data from Sequential-Access File

```cpp
      // read account, name and balance from file
      inClientFile >> account >> name >> balance;

      // display file contents (until eof)
      while ( !inClientFile.eof() ) {

         // display record
         if ( shouldDisplay( request, balance ) )
            outputLine( account, name, balance );

         // read account, name and balance from file
         inClientFile >> account >> name >> balance;

      } // end inner while

      inClientFile.clear();    // reset eof for next input
      inClientFile.seekg( 0 ); // move to beginning of file
      request = getRequest();  // get additional request from user

   } // end outer while

   cout << "End of run." << endl;

   return 0; // ifstream destructor closes the file

} // end main
```

Use **`clear`** to reset eof. Use **`seekg`** to set file position pointer to beginning of file.

Reading Data from Sequential-Access File

```cpp
      // read account, name and balance from file
      inClientFile >> account >> name >> balance;

      // display file contents (until eof)
      while ( !inClientFile.eof() ) {

         // display record
         if ( shouldDisplay( request, balance ) )
            outputLine( account, name, balance );

         // read account, name and balance from file
         inClientFile >> account >> name >> balance;

      } // end inner while

      inClientFile.clear();    // reset eof for next input
      inClientFile.seekg( 0 ); // move to beginning of file
      request = getRequest();  // get additional request from user

   } // end outer while

   cout << "End of run." << endl;

   return 0; // ifstream destructor closes the file

} // end main
```

Use **`clear`** to reset eof. Use **`seekg`** to set file position pointer to beginning of file.

Reading Data from Sequential-Access File

```
// determine whether to display given record
bool shouldDisplay( int type, double balance )
{
   // determine whether to display credit balances
   if ( type == CREDIT_BALANCE && balance < 0 )
      return true;

   // determine whether to display debit balances
   if ( type == DEBIT_BALANCE && balance > 0 )
      return true;

   // determine whether to display zero balances
   if ( type == ZERO_BALANCE && balance == 0 )
      return true;

   return false;

} // end function shouldDisplay

// display single record from file
void outputLine( int account, const char * const name,
   double balance )
{
   cout << left << setw( 10 ) << account << setw( 13 ) << name
        << setw( 7 ) << setprecision( 2 ) << right << balance
        << endl;

} // end function outputLine
```

Reading Data from Sequential-Access File

```
Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 1

Accounts with zero balances:
300       White          0.00

Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 2
Accounts with credit balances:
400       Stone        -42.16
```

Reading Data from Sequential-Access File

```
Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 3

Accounts with debit balances:
100       Jones          24.98
200       Doe           345.67
500       Rich          224.62

Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 4
End of run.
```