Document Name		Confidentiality Level
File Operation		Only for Recipients' Reference
Project Code	Version	Document Code
BP	1.0	BP

File Operation

—— Read and Write File

Prepared by		Date	
Reviewed by		Date	
Approved by		Date	

Copyright Declaration

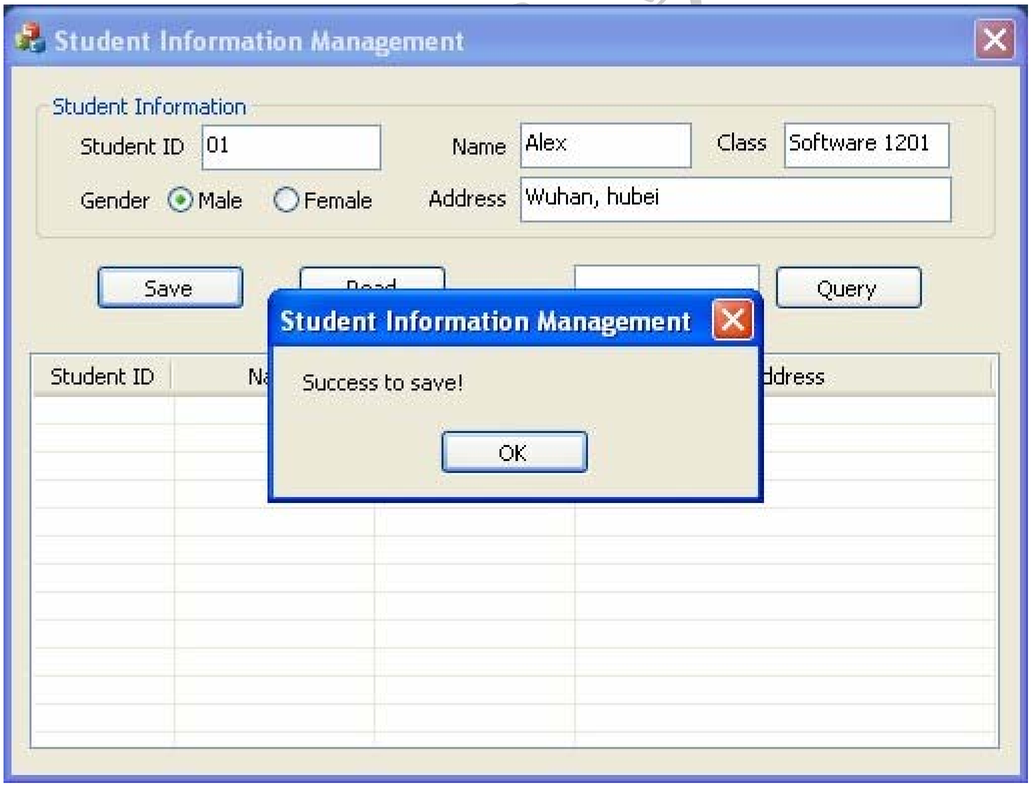
Contents

1 Teaching Tips

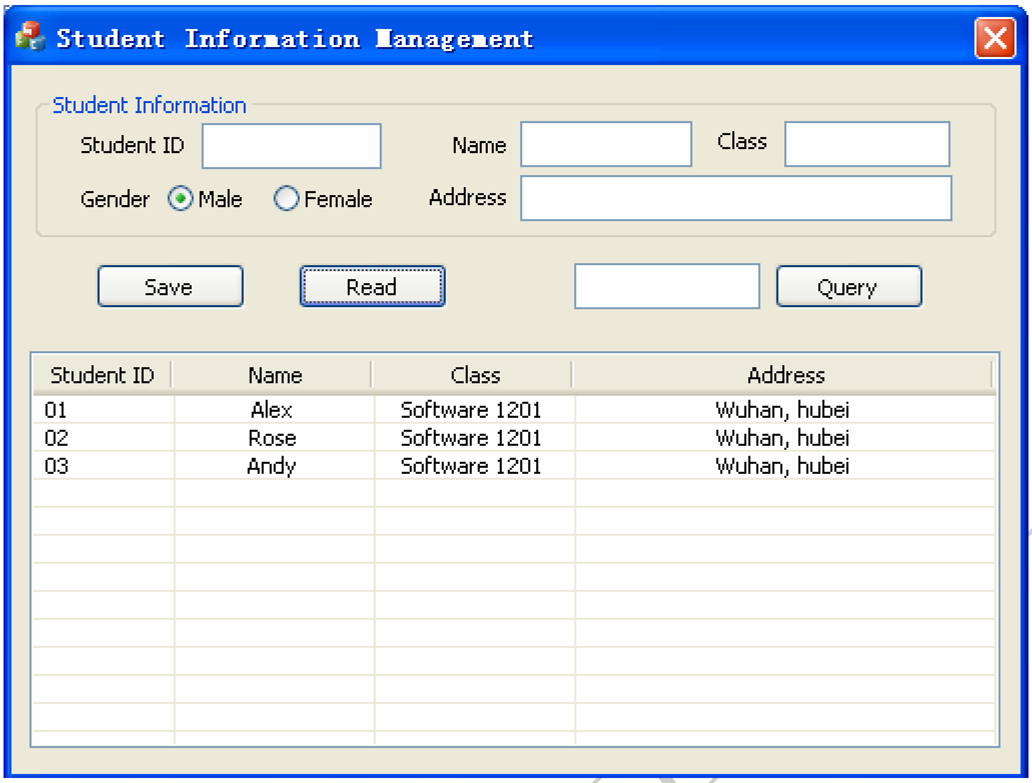
(1) Understand the concepts of file and file management.
(2) Master the file operations.
(3) Understand how to use the basic controls and the list control in MFC Dialog.
(4) Understand how to operate the binary file with CFile class.
(5) Finish the development of "File Operation" program with the above knowledge comprehensively.

2 Functional Requirements

Write a MFC dialog program to implement saving, reading and querying the student information. The student information includes: student ID, name, class, gender, home address. Each student ID corresponds to a student. The student ID can't duplicates.
The specific requirements are as follows:
1. Save Student Information
The user inputs the student information on the interface. Save the student information into Student.dat file under the project directory in binary mode.



2. Read Student Information
The user can read Student.dat file under the project directory through the program, and display the student information in the file into the list control on the interface. Student IDs, names, classes and home addresses of all students will be displayed in the list.

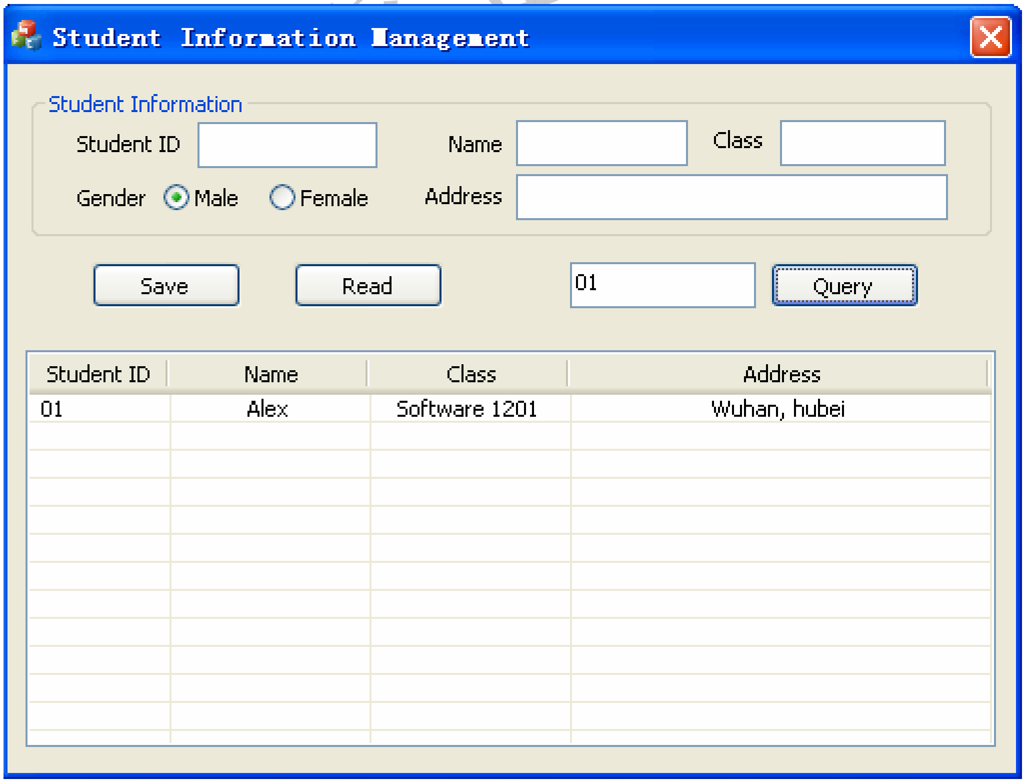


Gender: [✓ Male] Female

Student ID	Name	Class	Address
01	Alex	Software 1201	Wuhan, hubei
02	Rose	Software 1201	Wuhan, hubei
03	Andy	Software 1201	Wuhan, hubei

3. Query Student Information

We can query the student information in Student.dat file through student ID. The found student information will be displayed in the list control. Student IDs, names, classes and home addresses of the found students will be displayed in the list.



Gender: [✓ Male] Female

Query: 01

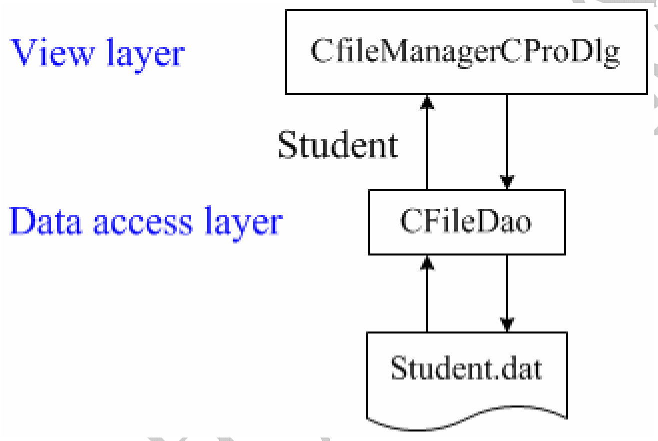
Student ID	Name	Class	Address
01	Alex	Software 1201	Wuhan, hubei

3 Design Ideas

With VS2010 development tool, create a MFC Dialog project. The solution name is FileManagerSln. The project name is FileManagerCPro. With “MFC Dialog + CFile +File” technology, implement the functions of saving, reading and querying the student information.

3.1 Program Structure Design

The program will use the three-layer structure in the architecture. Because the business is rather simple, so we will merge the business logic layer to the view layer, and complete the logic operations in the program at the view layer. Pass the student information entity between the layers. The program structure is as follows:

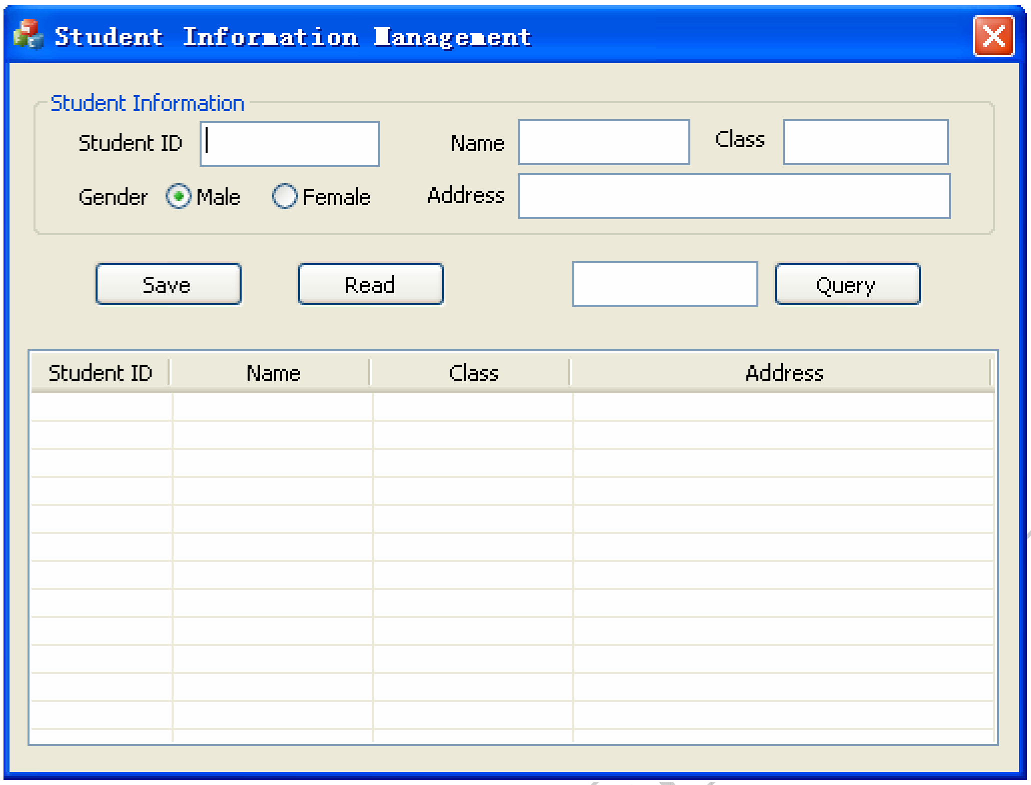


1. View Layer
The view layer is represented with CFileManagerCProDlg class. It will implement the input and output of the student information.

2. Data Access Layer
The data access layer is represented with CFileDao class. It will finish the file read/write operations.

3.2 Interface Design

The interface of this program is a dialog box. The interface effect is shown below:



1. With Edit control, receive the input student ID, name, class, home address, and student ID of the student that we want to find.
2. With the list control, display the found student information. Each column of the list is student ID, name, class and home address, respectively.
3. With the button, response the user's saving, reading and query operations.

3.3 Data Structure Design

1. Student structure
In the program, the student is represented with Student structure variable. Student structure includes student ID, name, class, gender, native place of the student. Student structure is declared in DateStructure.h file. The declaration is as follows:

```
struct Student
{
    char Num[16];       // Student ID
    char Name[16];      // Name
    char Class[32]; // Class
    int Gender;     // Gender
    char Address[128];  // Address
};
```

2. Student.dat File
Student.dat file is used to save the student information. The student information is stored in the file in binary form. Student.dat file is saved under the project directory. The path is defined with

the macro and the relative path. It is specified as Student.dat file under the project directory.

3.4 Function Implementation Design

1. Save Student Information Function
CFlieManagerCProDlg::OnBtnSave() obtains the input student information with the control mapping. And then, it will pass the student information and the file path where the student information is saved to CFileDao::Write().
According to the received student information and path, CFileDao::Write() will save the student information into the file under this path in binary mode.

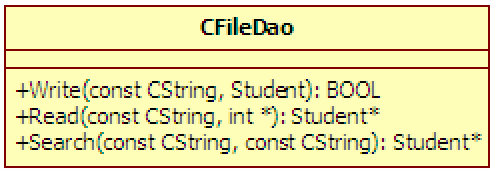
2. Read Student Information Function
According to the received file path, CFileDao::Read() will read the content of the file under this path in binary mode. It will read a piece of the student information every time. After reading, it will return all the student information with the return value.
CFileManagerCProDlg::OnBtnRead() will call CFileDao::Read() function to get the student information, and display the student information in the list control.

3. Search Student Information Function
CFileManagerCProDlg::OnBtnSearch() will obtain the found student ID through the control mapping, and then, pass the student ID and the file path to CFileDao::Search() function.
According to the received file path and student ID, CFileDao::Search() function will read the content of each file under this path item by item, and compare whether the student IDs are the same. If they are the same, return the student information, and search no more.

3.5 Class Design

The classes involved in this program include dialog class CFileManagerCProDlg, file operation class CFileDao and character conversion class CCharHelp.
1. CFileManagerCProDlg Class
The interface dialog class is used to interact with the user.

2. CFileDao Class
The file operation class is used to implement the read-write operations of the file.



```
+Write(const CString, Student): BOOL
+Read(const CString, int *): Student*
+Search(const CString, const CString): Student*
```

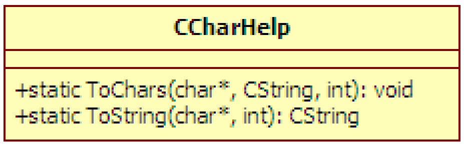
CFileDao::Write() function will save the student information into the specified file in the binary format.
CFileDao::Read() function will read the student information and the number of the student information in the specified path and file. The student information is returned through the return

value.

According to student ID, CFileDao::Search() function will query the student information of the same student ID in the specified path, and return the found student information through the return value.

3. CCharHelp Class

Character processing help class. Because, we may input the student information of CString type, but CString type is a variable-length string. When it is saved into the file, it is easily garbled. So, before it is saved into the file, we should convert CStirng type into char array.


CCharHelp
+static ToChars(char*, CString, int): void
+static ToString(char*, int): CString

4 Technical Analysis

MFC File Operation:

In MFC, it provides CFile class and CStdioFile class to perform the file operations.

1. CFile Class

CFile class is the base class of MFC file class. It directly provides the binary file operations, and supports the text file through its derived class.

2. CStdioFile Class

CStdioFile inherits from CFile. A CStdioFile object represents a C runtime streaming file opened with runtime function fopen. The streaming file is buffered, and can be opened in text mode (default) or binary mode.

3. Perform File Operation with CFile Class

(1) Open File

With CFile::Open() function, we can open the file under the specified path in specified mode.

1) The function prototype is as follows:

```
virtual BOOL Open( LPCTSTR lpszFileName, UINT nOpenFlags, CFileException*
pError = NULL );
```

2) Parameter meanings

lpsizeFileName: the file path.

nOpenFlags: the file access mode, such as `Read-only, Write-only, Creating new file, Reading and writing.`

The file access modes can be connected with "|" operator. It represents the file is accessed in such modes.

pError: the file access exception

(2) Close File

After the file operation is finished, we should perform close operation to the file. We can perform file closing operation through CFile::Close() function. The function prototype is as follows:

```
virtual void Close();
```

(3) Set Read/Write File Location

CFile file provides Seek(), SeekToBegin(), SeekToEnd() functions to set the location of the file reading and writing.

(4) Read File Operation

1) Open the file.

2) Perform read operation to the file.

CFile class provides Read() function to perform read operation to the file.

Function prototype:

```
virtual UINT Read( void* lpBuf, UINT nCount );
```

Parameter:

lpBuf: points to the pointer of the object read from the file.

nCount: the maximum byte number read from the file.

Return value: the actually read byte number.

3) Close File

(5) Write File Operation

1) Open the file.

2) Perform write operation to the file.

CFile class provides class Write() function to perform write operation to the file.

Function prototype:

```
virtual void Write( const void* lpBuf, UINT nCount );
```

Parameter:

lpBuf: points to the pointer of the object to be written into the file.

nCount: byte number written into the file.

3) Write the cached data into the file.

In order to perform write operation to the file, it will write the data into the cache first, and then, save the data in the cache into the file. So, after the file is written, we need to clear the cached data, and save the data in the cache into the file to prevent data loss.

CFile class provides Flush() function to perform this operation. The function prototype is as follows:

```
virtual void Flush( );
```

4) Close the file.

5 Implementation Idea

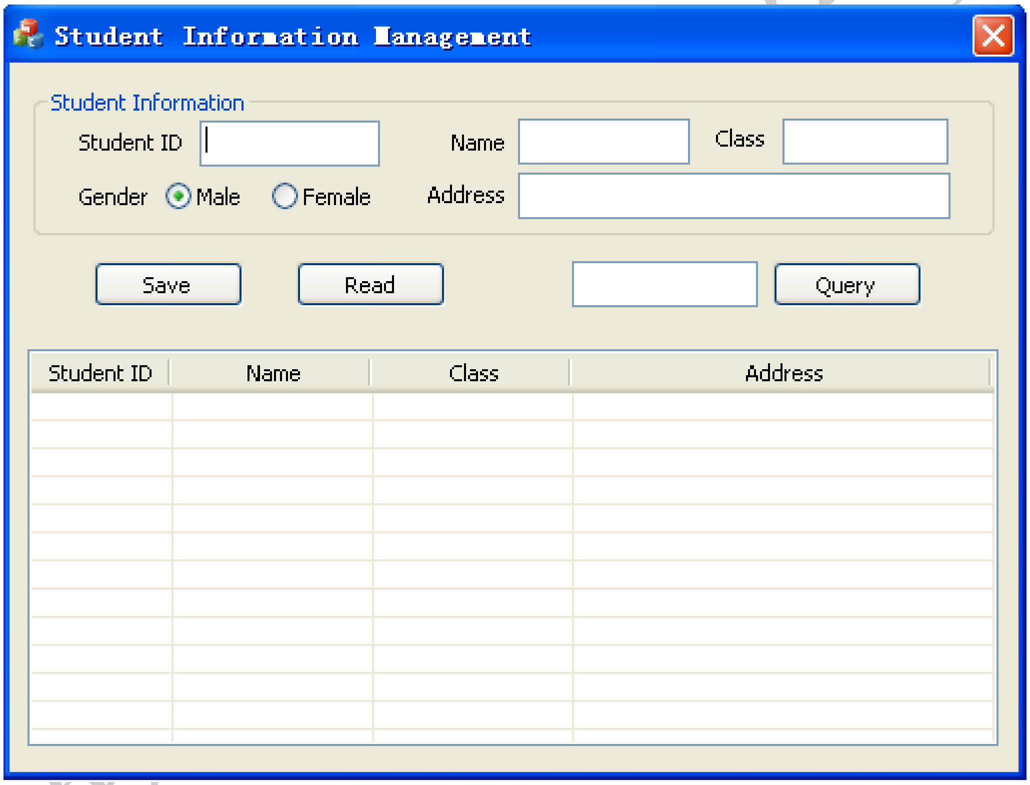
5.1 Create Project

1. Launch VS2010 development tool.
2. With VS2010 tool, create MFC dialog project. The solution name is FileManagerSln. The project name is FileManagerCPro.

5.2 Interface Layout

1. Add Controls
According to the interface design, with the tool box of VS2010, add the required controls into the dialog.

2. Interface Layout
According to the interface design, set the properties of the controls. The effect is shown below:



5.3 Save Student Information

1. Add student information structure
Add DataStructure.h file. Define student information structure Student in this file.

2. Add file operation class CFileDao.
(1) Define CFileDao class.
(2) Add CFileDao::Write() function.

```
class CFileDao
```

```
{
public:
    CFileDao(void);
    ~CFileDao(void);

public:
    BOOL Write(const CString strPath, const Student stu);
};
```

3. Write Save Student Information function CFileDao::Write().
(1) Open the file.
(2) Save the student information into the file.
(3) Clear the cache.
(4) Close the file.

```
BOOL CFileDao::Write(const CString strPath, const Student stu)
{
    CFile file;
    if (Fail to open file in read-only way)
    {
        if (Fail to open file in creating and writing way)
        {
            Return FALSE
        }
    }
    // Write file in binary way
    // Write buffer data into file
    // Close file
}
```

4. Receive the input student information.
(1) Add BN_CLICKED message response function CFileManagerCProDlg::OnBtnSave() function for “Save” button.
(2) Add CCharHelp class. In the class, add CCharHelp::ToChars() function to convert CString type to Char type.
(3) In CFileManagerCProDlg::OnBtnSave(), according to the control mapping, save the input student information into Student variable.
(4) In CFileManagerCProDlg::OnBtnSave() function, call CFileDao::Write() function to save the student information into Student.dat file.

5. Compile and run the program.

5.4 Read Student Information

1. Add Read Student Information function CFileDao::Read().

```
Student* CFileDao::Read(const CString strPath, int *pCount)
{

}
```

2. Write Read Student Information function.
(1) Open the file in read-only mode.
(2) Obtain the student information in the file.
(3) Close the file.

```
Student* CFileDao::Read(const CString strPath, int *pCount)
{
    CFile file;
    if (Fail to open file in read-only way)
    {
        return NULL;
    }

    // Get the size of the file

    // Allocate the memory according to amount

    // Move to the header of the file

    // Obtain content from the file

    // Close file
}
```

3. Call Read Student Information function.
(1) Add "Read" button BN_CLICKED message response function CFileManagerCProDlg::OnBtnRead().
(2) In CFileManagerCProDlg::OnBtnRead() function, call CFileDao::Read() function, and display the obtained student information in the list.

```
void CFileManagerCProDlg::OnBtnRead()
{
    // Read file
    // Show the read result on the list
    // Release the dynamically allocated memory
}
```

Notice: the content read from the file is the structure. The type of the structure member is char array and int type. The data displayed in the list is CString class. So, we need to add member function CCharHelp::ToString() function into CCharHelp class to convert char array to CString type.

4. Compile and run the program.

5.5 Search Student Information

1. Add Search Student Information function CFileDao::Search().

```
Student* CFileDao::Search(const CString strPath, const CString strNum)
{
}
```

2. Write Search Student Information function.

```
Student* CFileDao::Search(const CString strPath, const CString strNum)
{
    // Open file in read-only way

    // Allocate memory dynamically, used to saving student information

    // Read file
    while(file.Read(pStu, sizeof(Student)) > 0)
    {
        // Decide whether the student ID matching
    }

    // Close file
}
```

3. Call Search Student Information function.

(1) Add BN_CLICKED message response function CFileManagerCProDlg::OnBtnSearch() for "Search" button.

(2) In CFileManagerCProDlg::OnBtnSearch(), call CFileDao::Search() function to query the student information.

(3) In CFileManagerCProDlg::OnBtnSearch(), display the found student information in the list.

```
void CFileManagerCProDlg::OnBtnSearch()
{
    // Get the student ID for querying
    // Query student information according to the student ID
    // Decide whether there is relevant student information
    if (pStu != NULL)
    {
        // If there is student information,show it on the list
    }
}
```

4. Compile and run the program.