VC++数据库应用

一、 数据库应用分类

（1） 企业级数据库

MySQL、SQL Server、Oracle、DB2、Interbase

（2） 桌面数据库

Access、Paradox、Dbase、Excel

二、 什么是 ODBC

ODBC(Open Database Connectivity，开放数据库互连)是微软公司开放服务结构(WOSA，Windows Open Services Architecture)中有关数据库的一个组成部分，并提供了一组对数据库访问的标准 API（应用程序编程接口）。这些 API 利用 SQL 来完成其大部分任务。ODBC 本身也提供了对 SQL 语言的支持，用户可以直接将 SQL 语句送给 ODBC。开发的应用程序可通过 ODBC 访问数据库。

三、 创建 ODBC 数据源

在 win7 64 位系统中运行命令 C:/Windows/SysWOW64/odbcad32.exe，windows32 位系统中通过“控制面板-》管理工具-》数据源(ODBC)”操作路径找到 ODBC 数据源工具，打开 ODBC 数据源如图 1 所示。

图 1 ODBC 数据源

单击“添加”按钮创建新的数据源，如图 2 所示。在图中选择 Access 数据库驱动。

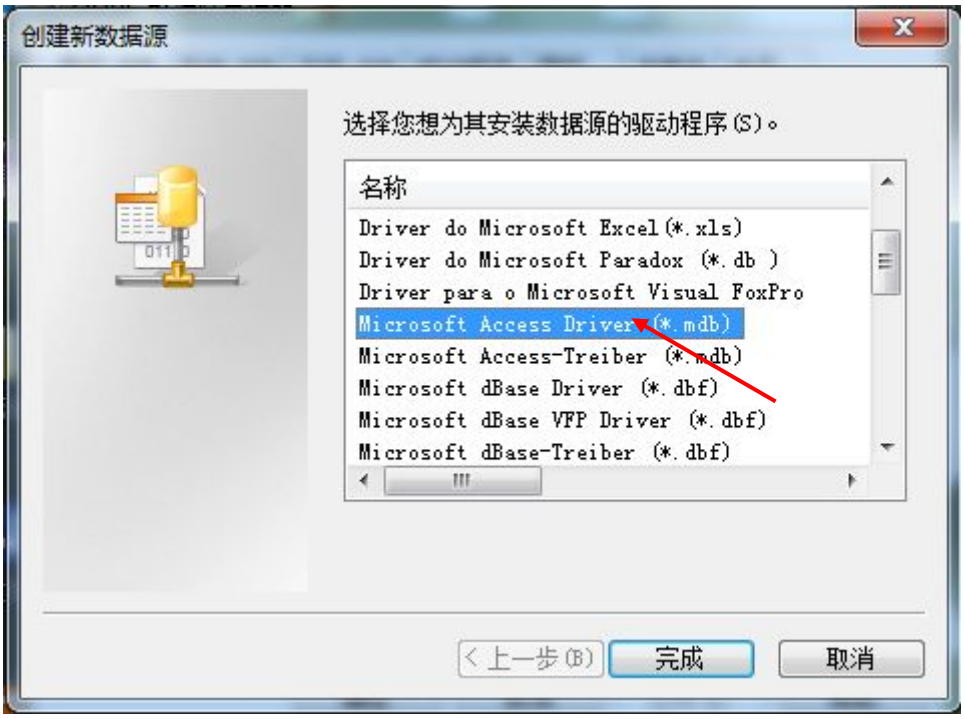


图 2 创建新的 ODBC 数据源

单击“完成”按钮后，出现“ODBC Microsoft Access 安装”对话框，如图 3 所示，在数据源名处输入“sl”作为数据源名。单击“选择(S)”按钮，弹出“选择数据库”对话框，在该对话框中选择 student.mdb 数据库，然后单击“确定”按钮。

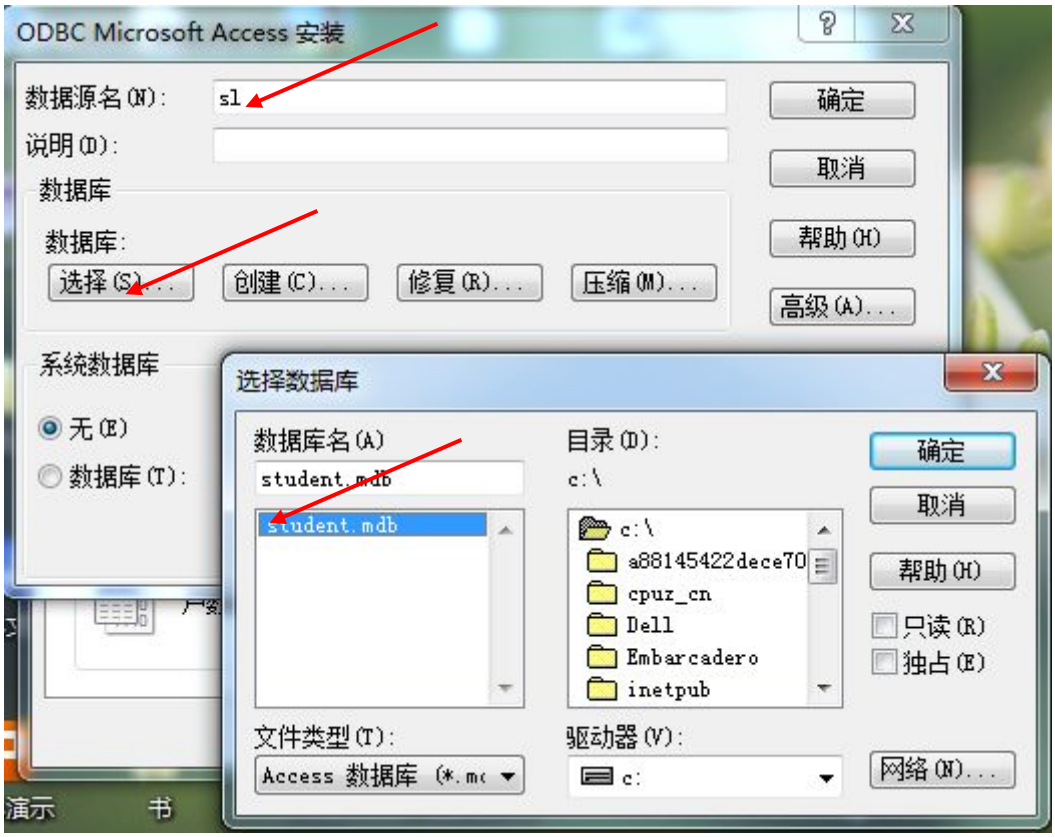


图 3 “ODBC Microsoft Access 安装”对话框

最后出现图 4 的对话框，表明连接 ODBC 的数据源 sl 创建完成。

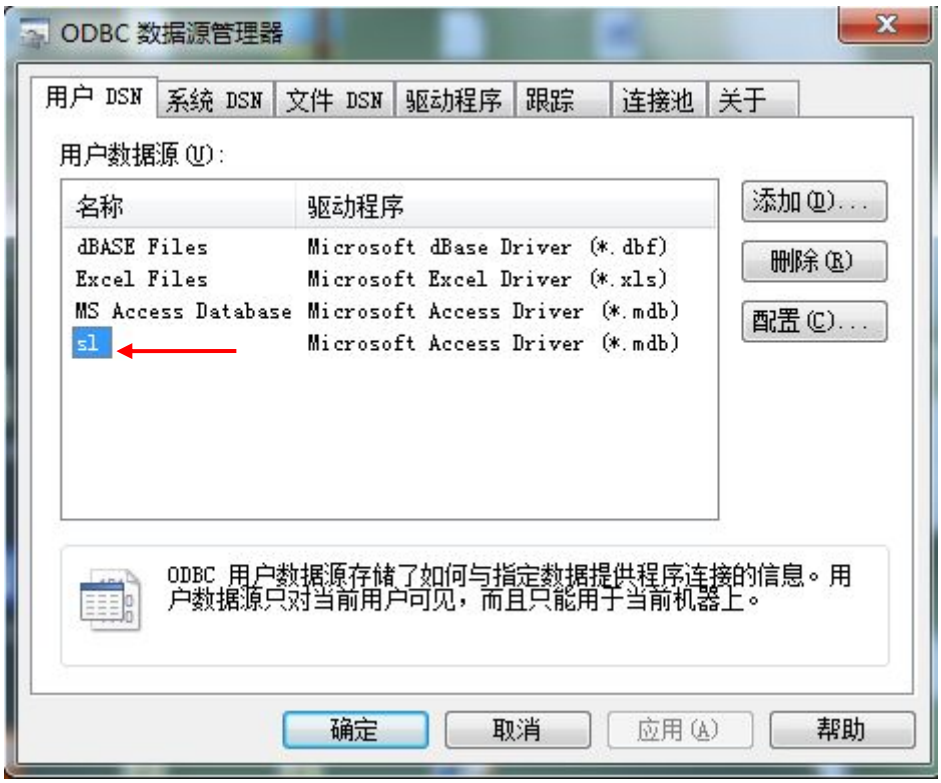
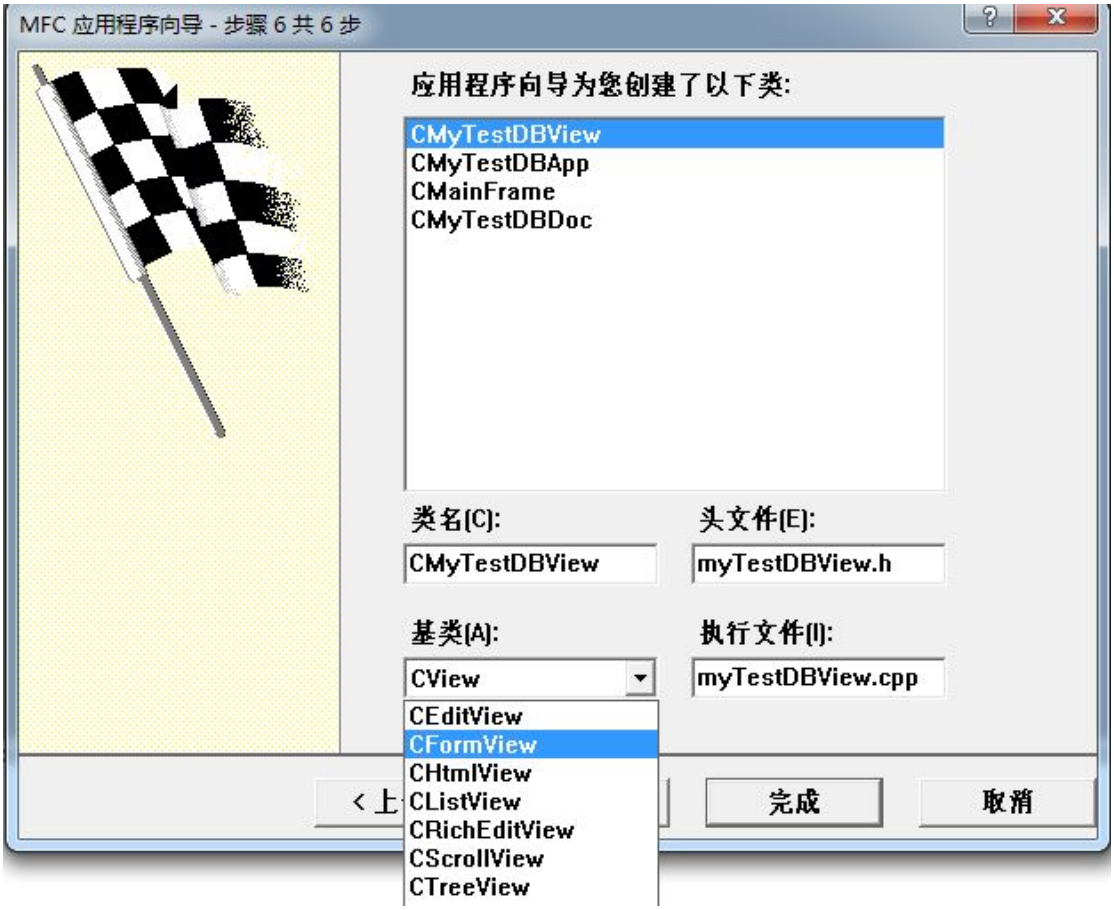


图 4 ODBC 的数据源 sl 创建完成

四、 VC++通过 ODBC 操作数据库

（1）创建 SDI 应用程序，在创建过程中第 6 步更改视图类为 CFormView，如图 5 所示。



如图 5 第 6 步更改视图类

知识要点：

（1） 连接数据库使用 CDatabase 类，格式如下：

```
CDatabase myDatabase;
myDatabase.Open(_T("sl"));//连接打开数据库
```

（2） 操作记录集使用 CRecordset 类，建立记录集与数据库联系格式如下：

```
CRecordset mydataset;
mydataset. m_pDatabase=& myDatabase;
```

（3） 打开记录集

```
mydataset.Open( );
```

注意：使用数据库的相关类要加头文件 afxdb.h

（2）为了方便操作记录集，一般将 CRecordset 类派生出子类，子类中包括了与数据库中表关联的字段变量，下面建立这样的记录集子类 mydb。使用菜单“插入-》类”操作路径打开“建立新类”对话框，按图 6 方式填写。

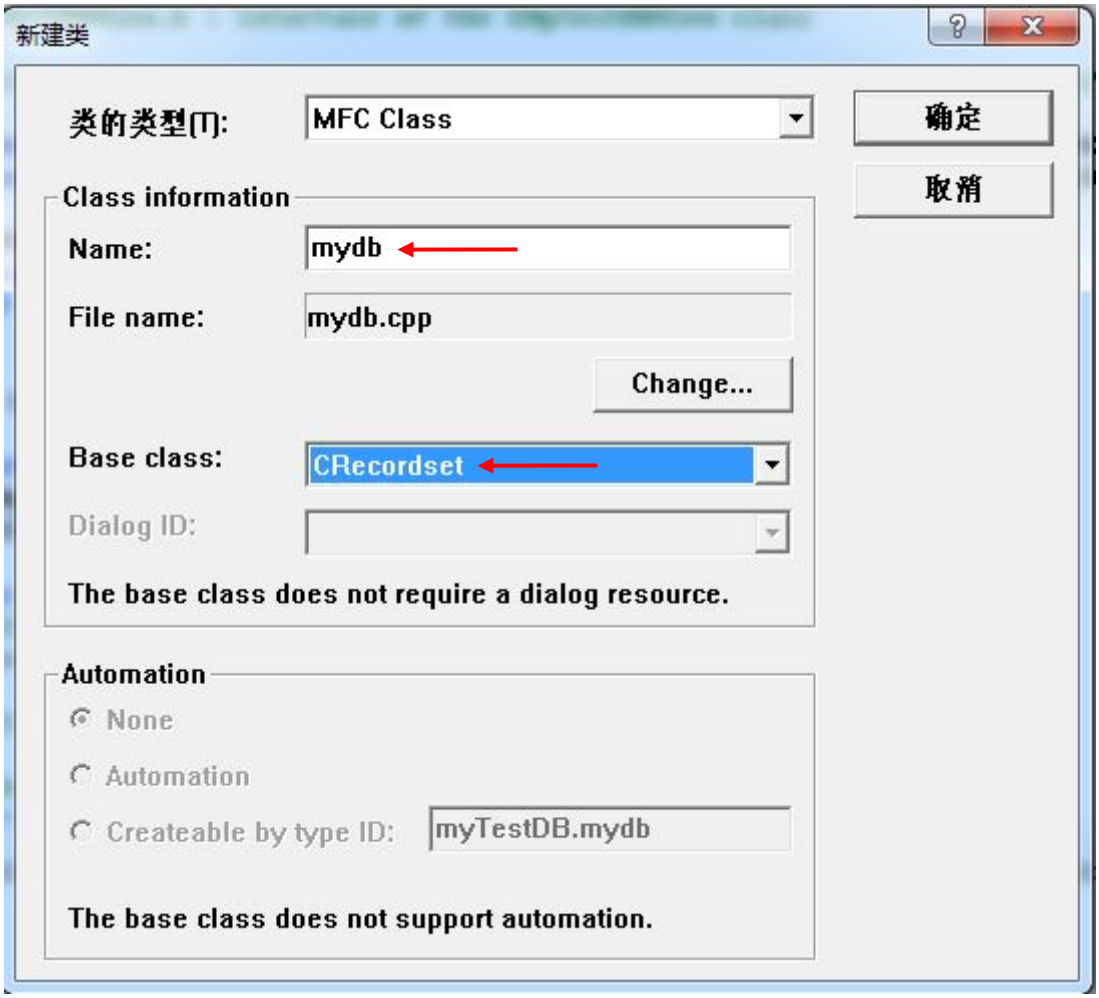


图 6 创建 mydb 记录集子类

单击“确定”按钮后出现图 7“数据库选择”对话框，数据源 ODBC 选择“sl”，记录集类型使用 Snapshot（快照），单击“OK”按钮后出现图 8“选择数据库表”对话框，选择“学生基本信息表”然后单击“OK”按钮即可生成 mydb 类。

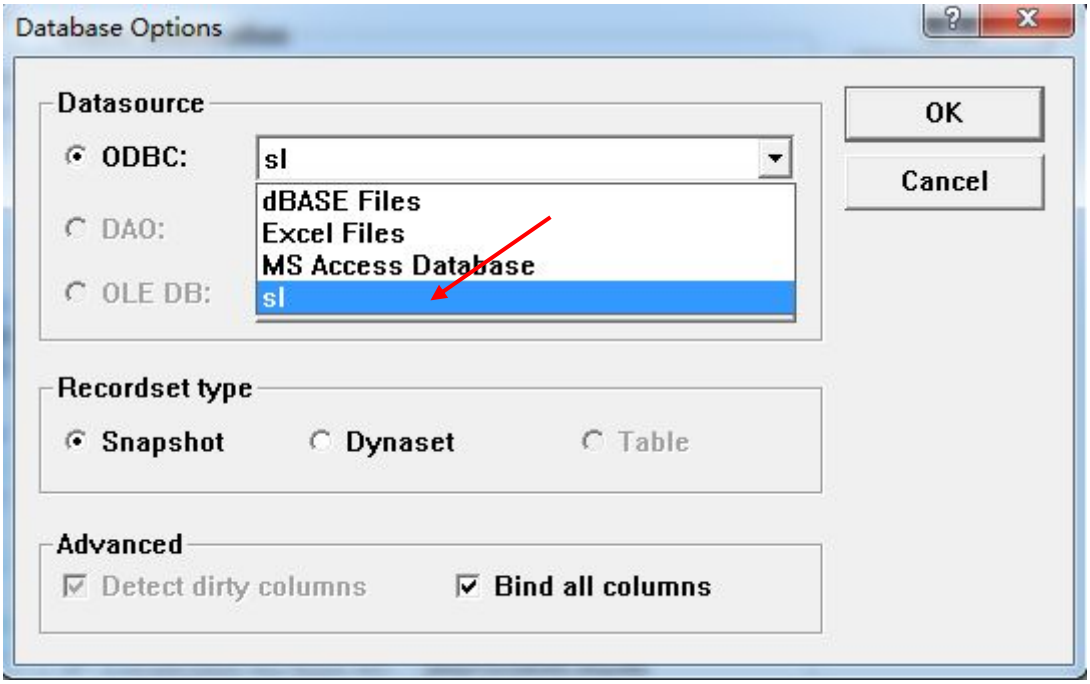


图 7“数据库选择”对话框

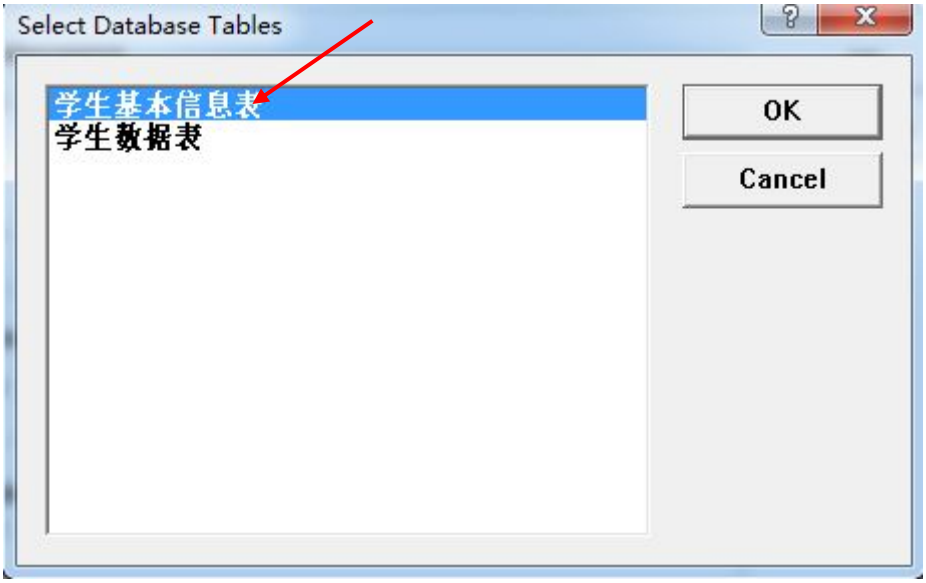


图 8 “选择数据库表”对话框

打开 mydb 类头文件，加入#include “afxdb.h”代码，如图 9 所示。同时，可看到类中声明的变量。

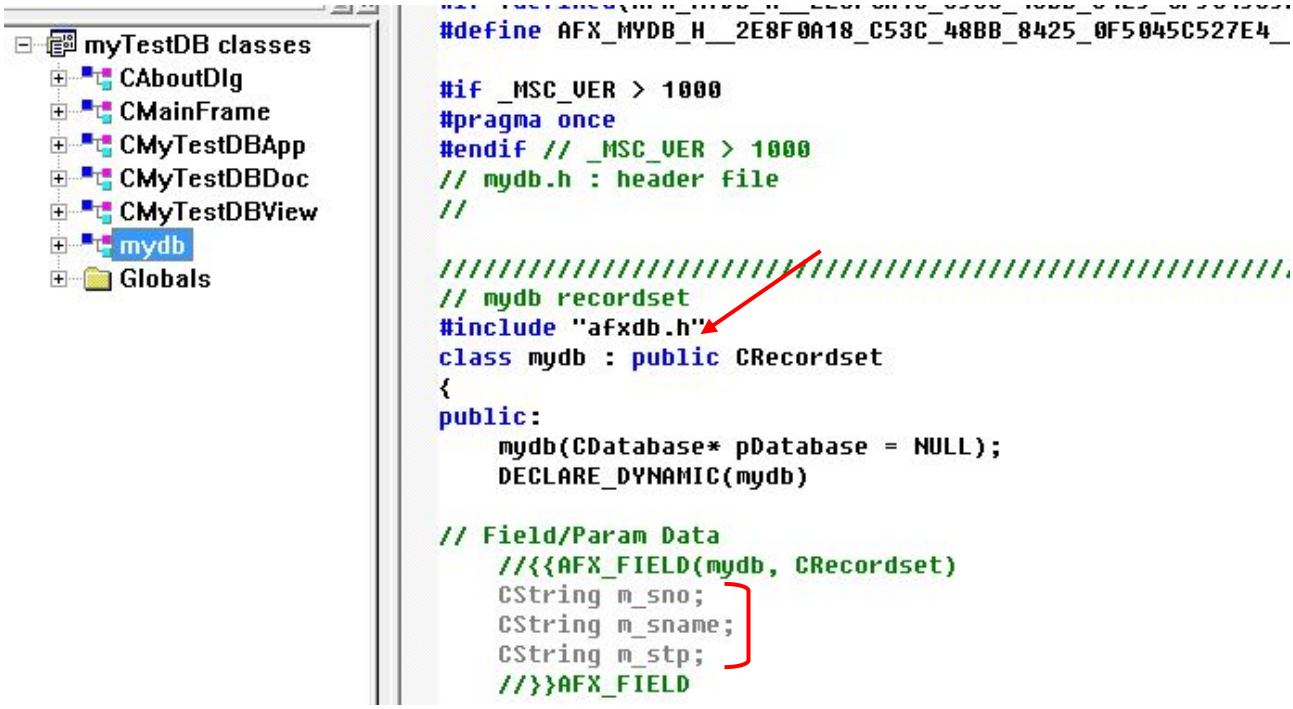


图 9 加入头文件 afxdb.h

打开 mydb 类的源文件，如图 10 所示，可看到 GetDefaultConnect 函数返回数据源“sl”，GetDefaultSQL 函数设置了要访问的默认数据库表，并且“学生基本信息表”的字段通过字段交换函数 DoFieldExchange 映射到 3 个变量中。

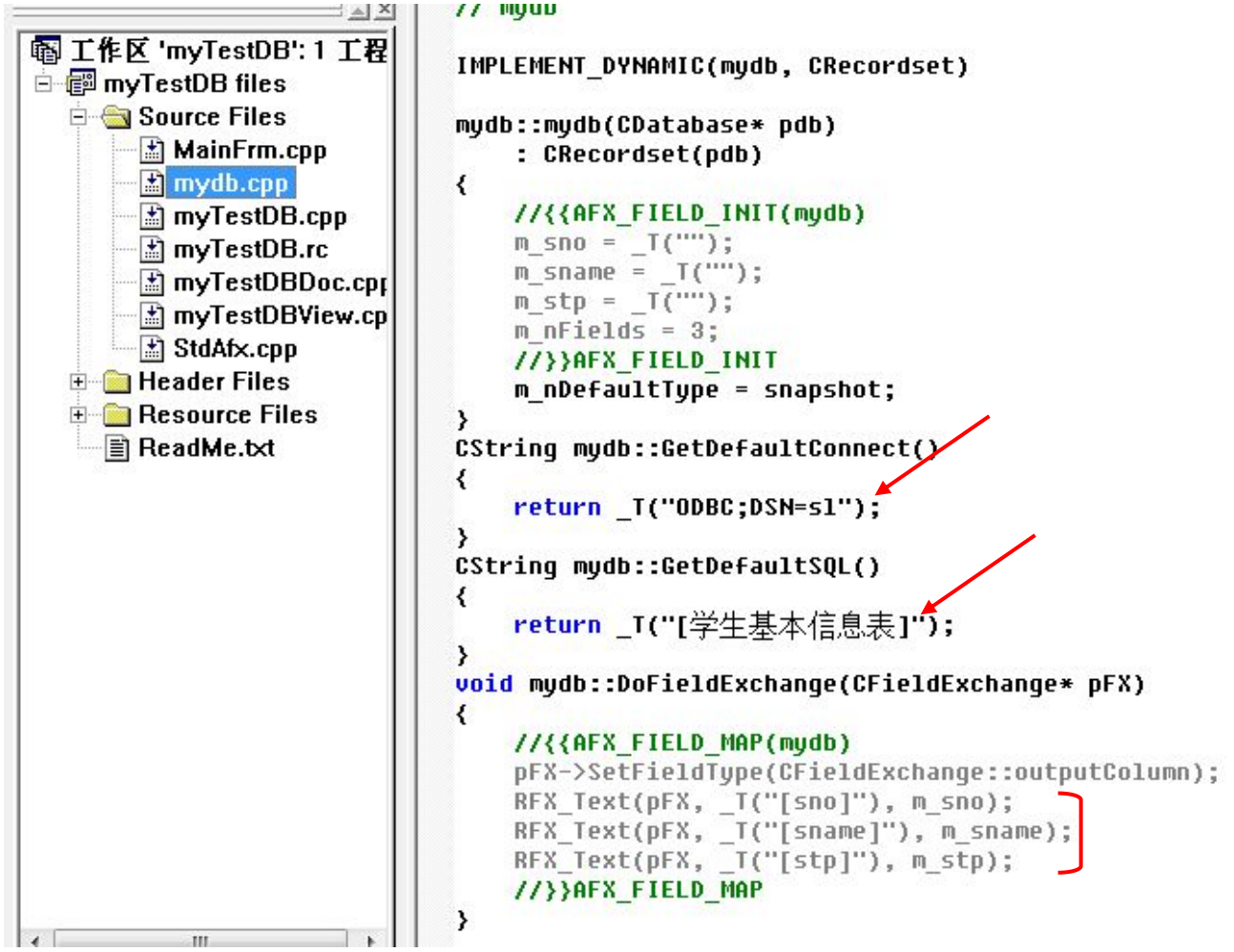


图 10 mydb 类的源文件

（3）在视图类中定义数据库对象和记录集对象 （当然事先加载 mydb 类的头文件）如图 11，在视图类的构造函数中填写如下代码，用于连接数据库和学生基本信息表，如图 12。

图 11 定义数据库对象和记录集对象

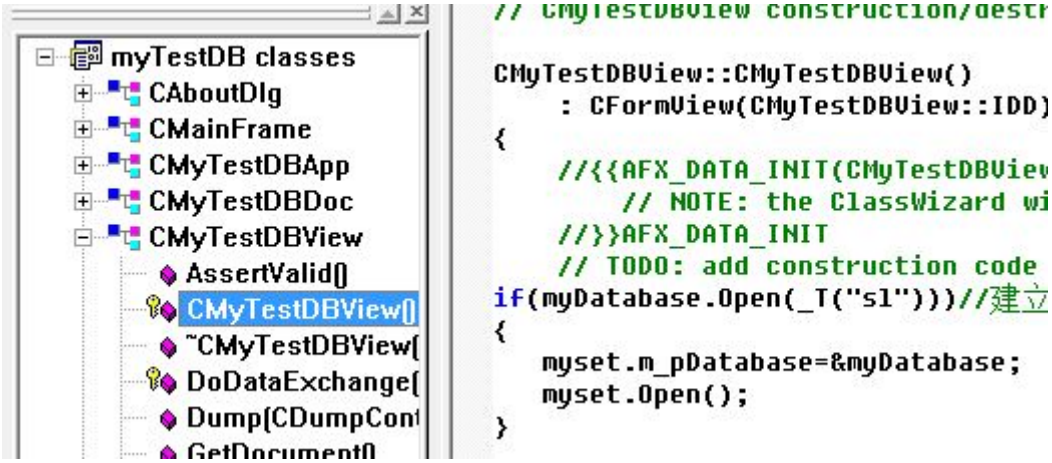


图 12 连接数据库和学生基本信息表代码

（4）设计如图 13 的界面，对文本框进行了数据交换设置。

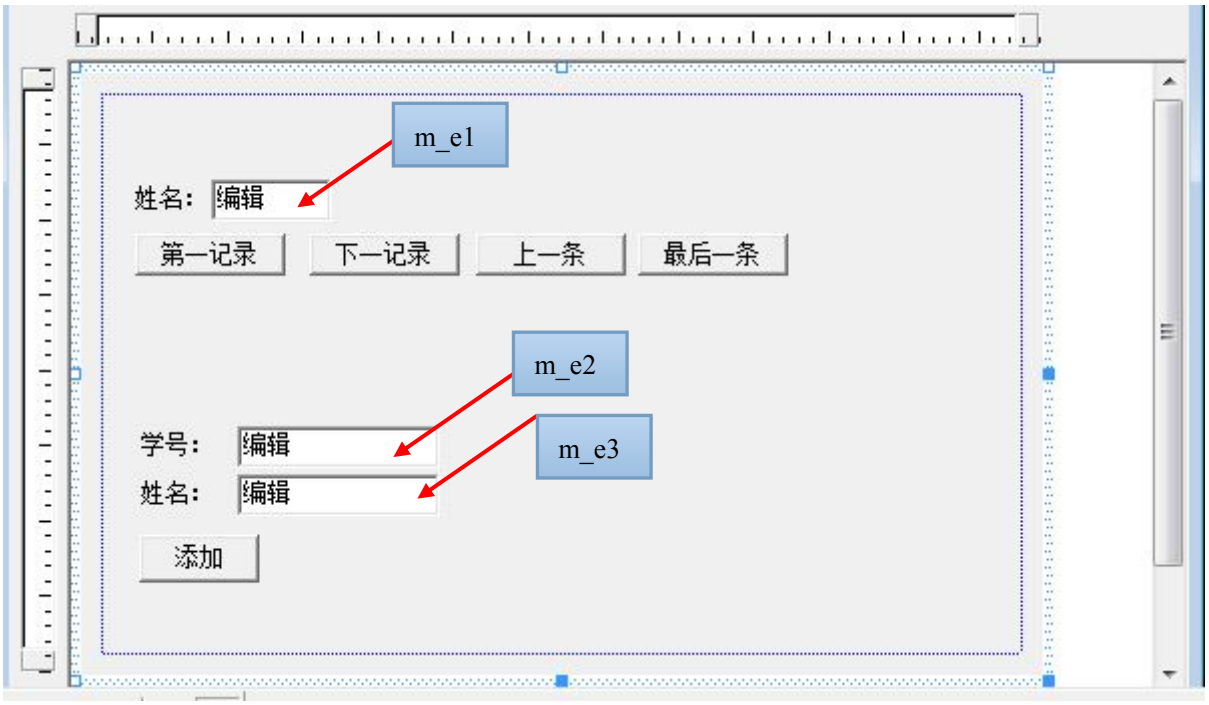


图 13 视图界面

知识要点：（定位记录指针）

CRecordset 提供了几个成员函数用来在记录集中记录指针滚动，如下所示。当用这些函数执行后记录指针滚动到一个新记录时，MFC 框架会自动地把新记录的内容拷贝到对应的数据成员中。

- void MoveNext( ); //前进一个记录
- void MovePrev( ); //后退一个记录
- void MoveFirst( ); //滚动到记录集中的第一个记录
- void MoveLast( ); //滚动到记录集中的最后一个记录

- BOOL IsEOF( ) const;

//如果记录集为空或滚动过了最后一个记录的下面，那么函数返回 TRUE，否则返回 FALSE。

- BOOL IsBOF( ) const;

//如果记录集为空或滚动过了第一个记录的上面，那么函数返回 TRUE，否则返回 FALSE。

（5）记录集中记录指针滚动代码

```
void CMyTestDBView::OnButton1( ) //记录指针定位到第一条记录
{
    // TODO: Add your control notification handler code here
```

```
    myset.MoveFirst( );
    this->m_e1=myset.m_sname;
    this->UpdateData(false);
}
void CMyTestDBView::OnButton2( ) //记录指针定位到下一条记录
{
    // TODO: Add your control notification handler code here
    myset.MoveNext( );
    if(!myset.IsEOF( ))
    {this->m_e1=myset.m_sname;
     this->UpdateData(false);
    }
    else
    {
      myset.MovePrev( );
    }
}
void CMyTestDBView::OnButton3( ) //记录指针定位到上一条记录
{
    // TODO: Add your control notification handler code here
    myset.MovePrev( );
    if(!myset.IsBOF( ))
    {this->m_e1=myset.m_sname;
     this->UpdateData(false);
    }
    else
    {
      myset.MoveNext();
    }
}
void CMyTestDBView::OnButton4( ) //记录指针定位到最后一条记录
{
    // TODO: Add your control notification handler code here
    myset.MoveLast();
    this->m_e1=myset.m_sname;
    this->UpdateData(false);
}
```

知识要点：（添加记录）

要向记录集中添加新的记录，应该按下列步骤进行：

- 调用 AddNew 成员函数。调用该函数后就进入了添加模式，该函数把所有对应数据成员都设置成 NULL。AddNew 会把当前记录对应数据成员的内容保存在一个缓冲区中，在必要的时候，程序可以再次调用 AddNew 取消添加操作并恢复数据成员原来的值，调用后程序仍处于添加模式。调用 Move 函数可退出添加模式，同时该函数会从缓冲区中恢复数据成员的值。
- 设置记录对应的数据成员值。
- 调用 Update。Update 把数据成员中的内容作为新记录写入数据源，从而结束了添加操作。
- 上面提及的函数是 void AddNew( )和 BOOL Update( )。

注意：如果记录集是快照，那么在添加一个新的记录后，需要调用 Requery 重新查询，因为快照无法反映添加操作。

（6）添加记录

```
void CMyTestDBView::OnButton5( )
{     this->UpdateData(true);
      myset.AddNew( );
     myset.m_sno=m_e2;
      myset.m_sname=m_e3;
     myset.Update( );
}
```

知识要点：（删除记录）

要删除记录集的当前记录，应按下面两步进行：

- 调用 Delete 成员函数。该函数会同时给记录集和数据源中当前记录加上删除标记。注意不要在一个空记录集中调用 Delete，否则会产生一个异常。
- 滚动到另一个记录上跳过被删除记录以完成删除操作。
- 上面提及的函数是 void Delete( )。

（7）删除记录，在界面上添加 Button6 按钮，对应代码如下。

```
void CMyTestDBView::OnButton6( )
{
    // TODO: Add your control notification handler code here
```

```
    myset.Delete( );
    myset.MoveNext( );
     if(!myset.IsEOF( ))
    {this->m_e1=myset.m_sname;
     this->UpdateData(false);
    }
    else
    {
      myset.Requery( );
      myset.MoveLast( );
      this->m_e1=myset.m_sname;
     this->UpdateData(false);
    }
}
```

知识要点：（更改记录）

要修改当前记录，应该按下列步骤进行：

- 调用 Edit 成员函数。调用该函数后就进入了编辑模式，程序可以修改对应的数据成员。注意不要在一个空的记录集中调用 Edit，否则会产生异常。Edit 函数会把当前对应的数据成员的内容保存在一个缓冲区中，这样做有两个目的，一是可以与对应数据成员作比较以判断哪些字段被改变了，二是在必要的时侯可以恢复对应数据成员原来的值。若再次调用 Edit，则将从缓冲区中恢复对应数据成员，调用后程序仍处于编辑模式。调用 Move(AFX_MOVE_REFRESH)或 Move(0)可退出编辑模式(AFX_MOVE_REFRESH 的值为 0)，同时该函数会从缓冲区中恢复数据成员。
- 设置数据成员的新值。
- 调用 Update 完成编辑。Update 把变化后的记录写入数据源并结束编辑模式。
- 上面提及的函数是 void Edit( )和 BOOL Update( )。

（8）更新记录，在界面上添加 Button7 按钮，对应代码如下。

```
void CMyTestDBView::OnButton7( )
{
    // TODO: Add your control notification handler code here
     myset.Edit();
    this->UpdateData(TRUE);
    myset.m_sname=this->m_e1;
    myset.Update( );
}
```

知识要点：（过滤记录）

成员 m_strFilter 用于指定过滤器。m_strFilter 实际上包含了 SQL 的 WHERE 子句的内容，但它不含 WHERE 关键字。

（9）查找记录，在界面上添加 Button8 按钮，对应代码如下。

```
void CMyTestDBView::OnButton8( )
{
    // TODO: Add your control notification handler code here
myset.Close( );
myset.m_strFilter="sname='mm'";
myset.Open( );
}
```