2020/07/13_MFC_第11课_CAD06_序列化与反序列化

笔记本: MFC
创建时间: 2020/7/13 星期一 15:36
作者: ileemi

- 序列化于与序列化（Serialize and UnSerialize）
- C++序列化
- C++反序列化
  - C++序列化反序列化注意点
- MFC 序列化

序列化于与序列化（Serialize and UnSerialize)

类似游戏存档

存档（序列化）-- 把内存中的对象存储到文件
读档（反序列化）-- 把文件中的对象读到内存

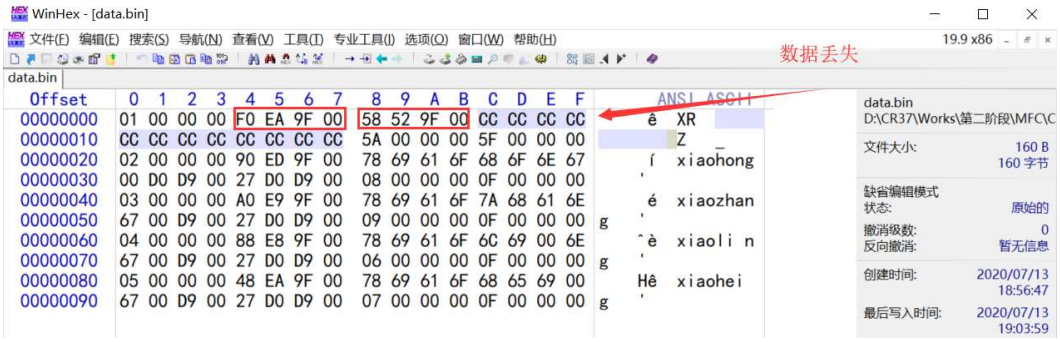
C++序列化

不能直接将对象存储到文件内，如果对象内部存在指针，存储的时候只是将指针的值存储到文件内，等到反序列化的时候，原来对象内部指针指向的内存上存储的数据就不能被正常读取。
示例：

```cpp
int main()
{
    CStudent stu1(1, "xiaobaibasdjsagdjsagdsjadgsajdgsajdgsajdsagdjsagdjsagdsakjdgsajdgsakjdgsajdkgdjsagdsjc
    CStudent stu2(2, "xiaohong");
    CStudent stu3(3, "xiaozhang");
    CStudent stu4(4, "xiaoli");
    CStudent stu5(5, "xiaohei");

    fstream file;
    file.open("data.bin", ios::binary | ios::out | ios::in);

    if (!file.is_open())
    {
        cout << "文件打开失败，请检查文件是否存在" << endl;
    }
    // 序列化
    file.write((char*)&stu1, sizeof(stu1));
    file.write((char*)&stu2, sizeof(stu2));
    file.write((char*)&stu3, sizeof(stu3));
```



正确的存储方法，在对应的类中写一个序列化方法，参数为fstream& fs，使用这只需要将文件传入，这个方法负责按照指定的格式进行存储（或者使用运算符重载）。代码示例：

```cpp
void Serialize(fstream& fs);
friend fstream& operator<<(fstream& fs, CStudent& stu);
```

```cpp
void Seralize()
{
    CStudent stu1(1, "xiaobai");
    CStudent stu2(2, "xiaohong");
    CStudent stu3(3, "xiaozhang");
    CStudent stu4(4, "xiaoli");
    CStudent stu5(5, "xiaohei");

    fstream file;
    file.open("data.bin", ios::binary | ios::out | ios::in | ios::trunc);

    if (!file.is_open())
    {
        cout << "文件打开失败，请检查文件是否存在" << endl;
    }
    // 序列化
    // 类内序列化
    void Serialize(fstream& file)
    {
        // 存储ID
        file.write((char*)&m_nID, sizeof(m_nID));
        // 存储姓名的长度
        int nNameSize = m_strName.size() + 1;
        file.write((char*)&nNameSize, sizeof(nNameSize));
        // 存储姓名
        file.write(m_strName.c_str(), nNameSize);
    }
```

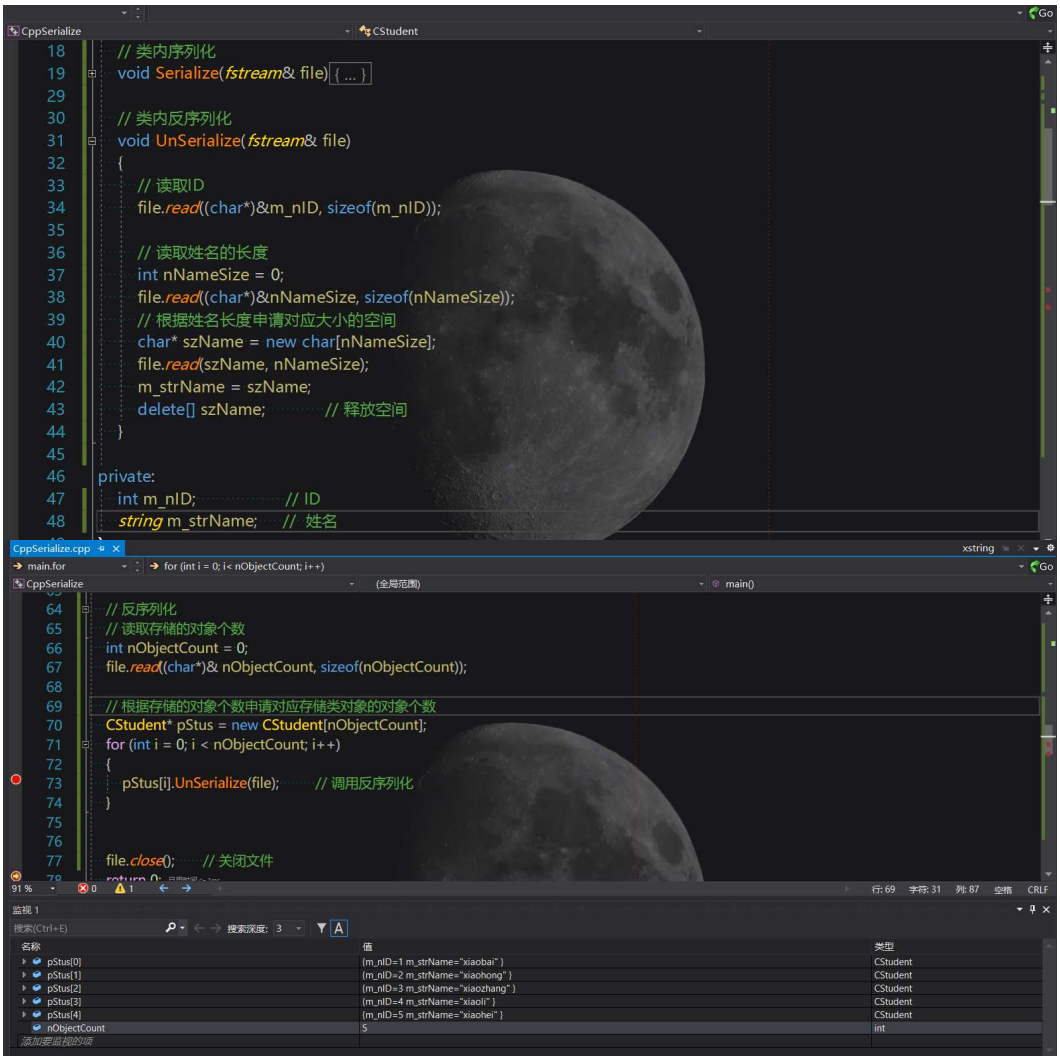
```cpp
// 存储需啊哟存储的类对象个数
int nObjectCount = 5;
file.write((char*)&nObjectCount, sizeof(nObjectCount));
// 存储对象
stu1.Serialize(file);
stu2.Serialize(file);
stu3.Serialize(file);
stu4.Serialize(file);
stu5.Serialize(file);


// 反序列化
cout << "数据写入成功" << endl;
file.close();      // 关闭文件
```

C++反序列化

读取文件数据时，申请存储对应数据的对象应该放置再堆上（使用栈存储会引发一系列问题）。同时存储数据到文件，也有可能是别人申请的，文件内的对象数量并不明确，对于应该申请多少对象是个未知，所以，再存储的时候，应该存储要存储对象的个数。

代码示例：



C++序列化反序列化注意点

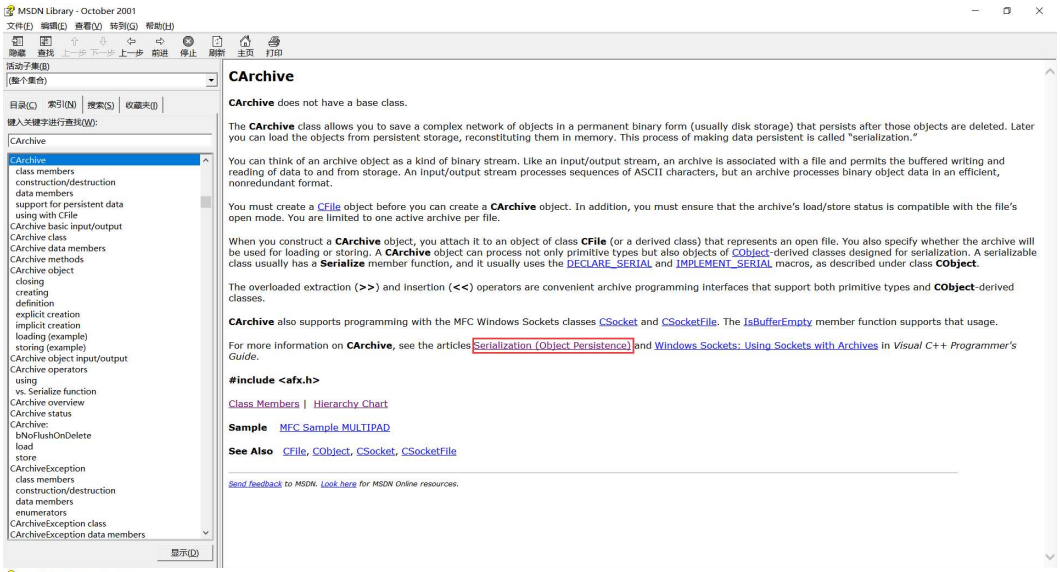
- 对象的类提供序列化和反序列化的成员
- 对象的个数

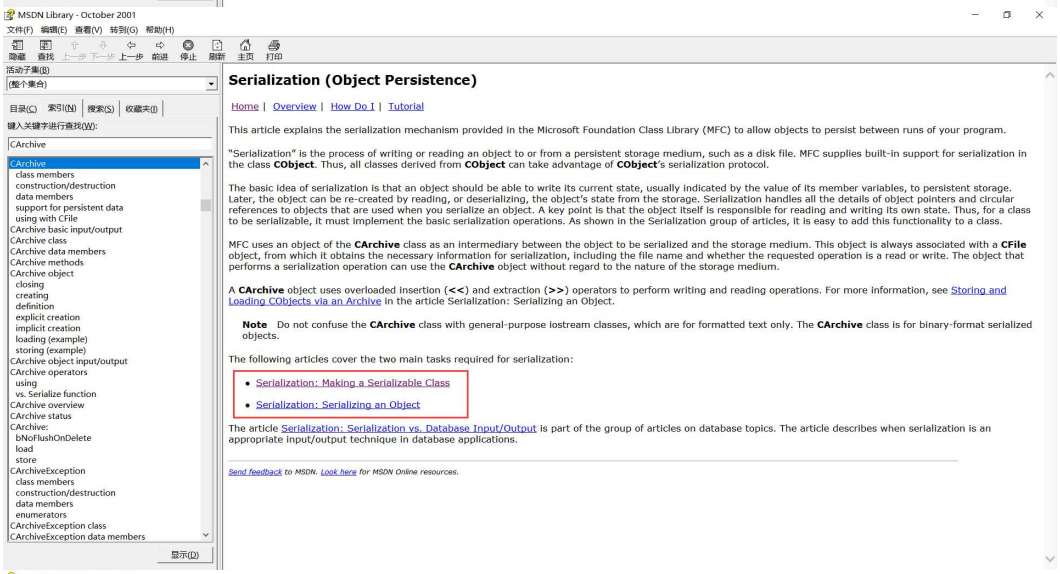
- 对象的类型信息要保存

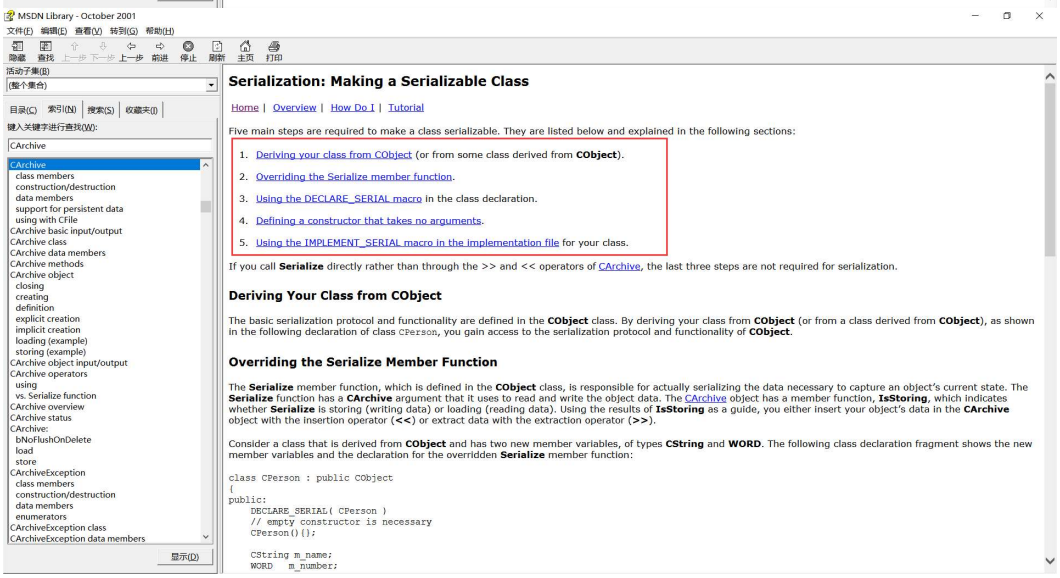
MFC 序列化

MFC 提供了一套序列化的机制

API -- CArchive

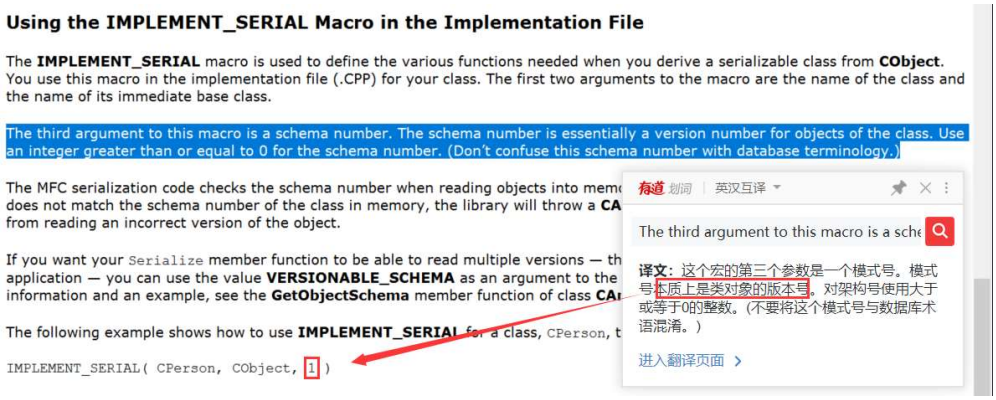






使用前，需要改造自己的类，改造步骤：

1. 继承 CObject
2. 重写（覆盖）Serialize 成员函数
3. 在类声明中使用宏 DECLARE_SERIAL
4. 定义一个没有参数的构造（无参构造）
5. 在类的实现文件中使用宏 IMPLEMENT_SERIAL



在控制台中使用MFC的库：
新建 "Windows桌面程序" --> 勾选"MFC标头"

MFC 文件类 -- **CFile**
CArchive -- 专门用于序列化和非序列化
CArchive提供了多种类型的运算符重载，同时支持 read 和 write 函数：

CArchive Class Members

Data Members

m_pDocument	Points to the **CDocument** object being serialized.

Construction

CArchive	Creates a **CArchive** object.
Abort	Closes an archive without throwing an exception.
Close	Flushes unwritten data and disconnects from the **CFile**.

Basic Input/Output

Flush	Flushes unwritten data from the archive buffer.
operator >>	Loads objects and primitive types from the archive.
operator <<	Stores objects and primitive types to the archive.
Read	Reads raw bytes.
Write	Writes raw bytes.
WriteString	Writes a single line of text.
ReadString	Reads a single line of text.

CArchive::operator >>

friend CArchive& operator >>(CArchive& *ar*, CObject *& *pOb*);
throw(CArchiveException, CFileException, CMemoryException);

friend CArchive& operator >>(CArchive& *ar*, const CObject *& *pOb*);
throw(CArchiveException, CFileException, CMemoryException);

CArchive& operator >>(BYTE& *by*);
throw(CArchiveException, CFileException);

CArchive& operator >>(WORD& *w*);
throw(CArchiveException, CFileException);

CArchive& operator >>(int& *i*);
throw(CArchiveException, CFileException);

CArchive& operator >>(LONG& *l*);
throw(CArchiveException, CFileException);

CArchive& operator >>(DWORD& *dw*);
throw(CArchiveException, CFileException);

CArchive& operator >>(float& *f*);
throw(CArchiveException, CFileException);

CArchive& operator >>(double& *d*);
throw(CArchiveException, CFileException);

Return Value

A **CArchive** reference that enables multiple insertion operators on a single line.