面向对象考试之文件存储

1. 对于MFC来讲使用的是序列化操作，直接加代码即可
2. 首先，MFC中肯定没有，我们自定义的结构体的运算符重载函数，需要我么自定义我们结构的运算符重载函数

   运算符重载函数无需声明直接定义使用即可，**最好定义在Doc的序列化方法前面。**

```
//写入结构体
CArchive& operator << (CArchive& ar, PRAMTIPS& pramTips) {
    ar.Write(&pramTips, sizeof(PRAMTIPS));
    return ar;
}
//读取结构体
CArchive& operator >> (CArchive& ar, PRAMTIPS& pramTips) {
    ar.Read(&pramTips, sizeof(PRAMTIPS));
    return ar;
}
```

3. 然后接着在Doc的序列化方法的前面，定义文件头

```
typedef struct tagFileHead
{
    int  version;
    CHAR szPswd[16];
    int dataCount;    //为了便于读取，存的时候往往带上结构体的个数
}FILEHEAD;
```

4. 最后是序列化读写操作

```
// CWQADoc 序列化
void CWQADoc::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
    {
        // TODO: 在此添加存储代码
        //创建文件头
        FILEHEAD fileHead = { 0,"@lzj2023",0 };
        fileHead.dataCount = dataBuffer.GetSize();
        //写入文件头
        ar.Write(&fileHead, sizeof(FILEHEAD));
        //写入内存数据
        ar.Write(&currentFont, sizeof(LOGFONT));         //写入当前字体
        ar.Write(&currentTextColor, sizeof(COLORREF));  //写入当前字体颜色
        //写入内存缓冲区
        for (int i = 0; i < fileHead.dataCount; i++)
        {
            //写入结构体
            ar << dataBuffer[i];
        }
    }
```

```
    else
    {
        // TODO: 在此添加加载代码
        //读取文件头
        FILEHEAD fileHead = { 0,"@lzj2023",0 };
        FILEHEAD thisfileHead;
        ar.Read(&thisfileHead, sizeof(FILEHEAD));
        //验证文件头是否正确
        if (strcmp(thisfileHead.szPswd, fileHead.szPswd) == 0)
        {
            //清除缓冲区（不要忘了清除缓冲区）
            ClearDataBuffer();
            //读取字体和颜色
            ar.Read(&currentFont, sizeof(LOGFONT));         //写入当前字体
            ar.Read(&currentTextColor, sizeof(COLORREF));  //写入当前字体颜色

            for (int i = 0; i < thisfileHead.dataCount; i++)
            {
                //读取结构体内容
                PRAMTIPS tempData;
                ar >> tempData;
                //加到缓冲区中
                dataBuffer.Add(tempData);
            }
        }
        else
        {
            //错误提示处理
            AfxMessageBox(_T("文件类型错误或不支持，请检查或重新选择文件"));
        }
    }
}
```