面向对象考试之数据结构

1. 这里如果类写的不是很熟的话，可以使用结构体。
2. 注意MFC中，数据结构定义在单独的.ccp和.h里面，别的文件要是想用，直接包含这个头文件即可
3. 默认dataStruction.cpp和dataStruction.h，顺便把 `#include "dataStruction.h"` 添加到pch.h里，一劳永逸

   注意dataStruction.h中添加：

   ```
   #pragma once
   #include <afxwin.h>         // MFC 核心组件和标准组件
   #include <afxext.h>
   ```

   在dataStruction.cpp中添加：

   ```
   #include "pch.h"            //自定义的源文件和头文件，必须包含pch.h 否则会报错
   #include "dataStruction.h"
   ```

4. 添加完结构后，先运行一下，没问题，再在.h中定义结构体，格式如下：

   ```
   typedef struct prammingTips {
       //类型
       char type[100];
       //描述
       char describe[100];
       //解决防范
       char solve[100];
       //其他
       char other[100];
   }PRAMTIPS;
   ```

5. 结构定义完后，用C++的动态数组容器去组织数据，直接在dataStruction.cpp中定义**dataBuffer的动态结构体数组，并添加声明**

```
CArray<PRAMTIPS> dataBuffer;  //内存缓冲区定义（PRAMTIPS要修改为自己的结构体名字）

//头文件的声明一定要放在结构体的定义之后
extern CArray<PRAMTIPS> dataBuffer;  //内存缓冲区定义
```

6. 这样的话，整个内存框架就建立好了，现在在dataStruction.cpp里写**内存操作函数**了

```
//向内存缓冲区中增加一条数据（类似类的构造函数）
BOOL AddOneDataToBuffer(char* type, char* describe, char* solve, char* other)
{
    //定义一个临时结构
    PRAMTIPS pramTips;
    //加载要添加的数据
    strcpy_s(pramTips.type, type);
    strcpy_s(pramTips.describe, describe);
```

```
    strcpy_s(pramTips.solve, solve);
    strcpy_s(pramTips.other, other);
    //将临时结构加载入缓冲区(注意A大写)
    dataBuffer.Add(pramTips);
    return TRUE;
}

//在内存缓冲区中删除一条数据(删除指定序号内的数据)
void DeleteOneDataFromBuffer(int id)
{
    dataBuffer.RemoveAt(id);
}

//清空内存缓冲区
void ClearDataBuffer()
{
    dataBuffer.RemoveAll();
}

//头文件中的声明//内存操作函数
BOOL AddOneDataToBuffer(char* type, char* describe, char* solve, char* other);
void DeleteOneDataFromBuffer(int id);
void ClearDataBuffer();
```

7. 清零某个结构体的方法

```
#include <string.h>
#include <stdlib.h>

memset(&(dataBuffer[i]),0,sizeof(PRAMTIPS));
```