# VC++6.0 应用开发中的几点经验总结

## 1 引言

绿色果品加工系统是用于对各类水果进行保质保成分加工的连续生产系统，采用计算机自动控制以保证水果的加工质量。系统主要功能包括：对三台加热器、四台制冷泵和二台风机进行一定顺序控制和闭环控制，实时采集温度信号和湿度信号，并显示系统中的温度分布状况和湿度值；定时将整理后的温度、湿度值存入数据文件，并可把它用曲线在屏幕上绘制出来，与给定曲线相比较。当温度超出工作允许范围时，系统及时发声报警。要求系统使用简单方便，人机界面友好。根据需求分析，系统采用VC++6.0进行开发，选用单文档多视的工作框架，运用文件进行数据存储。

## 2 C++、Visual C++6.0 和 MFC

C++语言是目前应用最为广泛的面向对象程序设计语言。C++语言是对C语言的扩充（或称为C语言的超集），它继承了C语言高效、灵活的特点，扩充完善了C语言的类型检查、代码重用、数据抽象机制，实现了对面向对象程序设计的支持。

Visual C++是微软公司推出的一种功能强大而复杂的C++编译器，它提供了功能最强大的Windows应用框架。微软基本类库（MFC）封装了Win32 API的函数，提供了大量编写好的代码以实现多数Windows标准操作。在Visual C++中，编写Windows程序时，可以使用SDK调用Windows本身提供的Win32 API函数编程，也可以利用MFC封装的类库编写程序，而最方便的是利用App Wizard向导建立程序的主体部分。

MFC是C++类库事实上的标准，它把Windows API（应用程序接口）的功能封装在逻辑类中。现在MFC第4个版本（4.2版）包含了将近200个类和大约2500个函数，为专业级和企业级Windows应用程序开发提供了强大的支持。

本文就绿色果品加工系统开发过程中的几点经验做总结。

## 3 应用程序框架的选择

用VC语言进行系统设计有别于VB、Delphi等编程语言。在VC中有一个代码生成器（AppWizard），编程人员通过它按照要求指定系统设计的特性，类名及源代码文件名就能产生应用程序的工作框架，从头到尾不需要编写一句程序，这看起来似乎很简单，这是代码生成器的优点，其实更不能忽视它的缺点。因为工作框架的生成只是系统设计的第一步，它决定了系统的基本特性，一旦用AppWizard生成框架以后便不能再更改。如选定程序的语言环境为Chinese，那么当该系统运行在全英文的系统下，程序中所有的中文内容都显示为乱码，这个时候你再想要把语言环境改为English的话，除非你否定掉之前所有的设计，重新从框架选定开始做起，所以应用程序框架的选择很重要，具体涉及到以下几个方面的问题：

(1) 应用类型的选择。一般设计都选择SDI（单文档多视）或MDI（多文档多视）。两者的区别是选择单文档界面，程序每次只能处理一个文档，不同的视图以不同的形式显示该文档中的数据，可以理解为一对多的关系；选择多文档界面，允许同时打开多个文档，一个文档可以对应一个或多个视图，是多对多的关系。一些有经验的编程人员建议选择MDI，因为MDI也可以当作SDI使用，而且有利于系统进一步的开发；但本系统功能比较明确，数据比较单一，只需一个数据文档就能满足系统工作要求，所以本系统开发采用了SDI的框架结构。

(2) 数据库支持的选择。数据储存有很多种方式，其中最通用的一种是使用数据库存储数据，因为用数据库存储便于数据的查询、删除、统计等操作。如果要用数据库进行数据存储的则应选择基于数据库支持的程序框架，如不用数据库而选用文件进行数据存储则选项为None。

(3) MFC library的选择。MFC DLL有三种形式，即Regular statically linked to MFC DLL（标准静态链接MFC DLL）、Regular using the shared MFC DLL（标准动态链接MFC DLL）以及Extension MFC DLL（扩展MFC DLL）。第一种DLL在编译时把使用的MFC代码链接到DLL中，执行程序时不需要其他MFC动态链接类库的支持，但体积较大；第二种DLL在运行时动态链接到MFC类库，因而体积较小，但却依赖于MFC动态链接类库的支持；这两种DLL均可被MFC程序和Win32程序使用。第三种DLL的也是动态连接，但作为MFC类库的扩展，只能被MFC程序使用。

在VC的AppWizard中，可选择两种MFC library形式：a shared DLL（共享动态链接库）和astatically linked library（静态链接库）。如果选择使用共享DLL可以得到较小的程序，但发布程序时需要同时发布这些DLL文件，并且程序移植会比较复杂；而选择静态链接MFC库，则生成的程序比较大，但程序自身是独立的，方便程序移植和发布。如果你的系统要求有高的移植性和独立性，建议选择静态链接库。就本系统而言，它要求有较高的移植性，整个系统比较独立，不受其他文件的限制，所以系统开发采用静态链接库。

**摘要**：文章总结了用VC++6.0开发的绿色食品加工系统中，有关框架选择、单文档多视、视图刷新、文件操作和与数据采集卡接口等项经验总结。

**关键词**：SDI　MFC　Visual C++

柳虹　袁南儿

（杭州浙江工业大学　310014）

**4 单文档多视结构**

针对本系统需要多种形式的人机界面，而使用者大多操作能力有限的特点，需要编写一个简洁、明了、不能操作出错的用户界面。我们采用了MFC的单文档-多视图的结构，利用多视图把系统的各个显示模块表现出来。但VC++并不直接支持这种结构，需要程序员通过编程实现。

利用AppWizard设定好工作框架以后，系统框架（Frame）和文档（Document）就生成了，伴随着Frame和Document的生成，系统也自动生成一个视图（View）。在此基础上若要实现多视，那么其他视图需要编程人员自己创建。创建完视图以后，再调用一个用于选择视的函数SelectViewChange(int nViewID)来切换视图。

```
//在框架类CmainFrame中自己定义一个用于视图切换的函数SelectViewChange，函数参数nViewID为视图代号
void CMainFrame::SelectViewChange(int nViewID)
{
//定义一个视图类Cview的指针变量pOldView，通过调用函数GetActiveView()获得系统当前活动视图的指针，同时赋给pOldView变量
CView *pOldView=GetActiveView();
//定义CruntimeClass类的一个指针变量pNewViewClass
CRuntimeClass * pNewViewClass;
//定义系统文档类CcontrolDoc的一个指针变量pDoc
//通过调用函数GetActiveDocument()获得系统当前活动文档的指针
//(CcontrolDoc*)用于强制调用函数GetActiveDocument()返回的是CcontrolDoc类型的指//针
CControlDoc* pDoc=(CControlDoc*)GetActiveDocument();
//通过函数SelectViewChange的参数nViewID来获得各个视图的编号，赋给文档类中定//义的一个变量ViewID
pDoc->ViewID=nViewID;
switch(nViewID)
{
    case 1: //nViewID=1表示视图1的标号为1
    //视图1是由向导生成的，调用函数RUNTIME-CLASS（视图1）获得指向视图1这个类结构的指针赋给pNewViewClass
    pNewViewClass=RUNTIME-CLASS(视图1);
    break;//从switch选择语句中退出
    case 2://nViewID=2表示视图2的标号为2
    //视图2由编程人员创建，调用函数RUNTIME-CLASS（视图2）获得指向视图2 这个类结构的指针赋给pNewViewClass
    pNewViewClass=RUNTIME-CLASS(视图2);
    break;
    case 3: //nViewID=3表示视图3的标号为3
    //视图3由编程人员创建，调用函数RUNTIME-CLASS（视图3）获得指向视图3 这个类结构的指针赋给pNewViewClass
    pNewViewClass=RUNTIME-CLASS(视图3);
    break;
  default:
  //没有选择任何视图时，执行default下面操作
    //宏ASSERT用于发现错误，ASSERT(0)打印一条诊断信息，同时中止程序的运行
    ASSERT(0);
    return;
}
//声明一个类CcreateContext类型的变量
CCreateContext context;
//将pNewViewClass的值赋给context变量中相对应的成员变量m-pNewViewClass
context.m-pNewViewClass=pNewViewClass;
//调用GetActiveDocument()获得指向当前活动文档的指针赋给context变量中相对应的成员变量m-pCurrentDoc
context.m-pCurrentDoc=GetActiveDocument();
//创建context指向的一个新视图，新视图的指针赋给pNewView，调用STATIC-DOWNCAST()，当创建不成功时，该函数会显示异常信息
CView* pNewView=STATIC_DOWNCAST(CView,CreateView(&context));
if(pNewView!=NULL)//pNewView!=NULL说明新视图创建成功
{
//调用ShowWindow()显示pNewView所指向的新视
    pNewView->ShowWindow(SW-SHOW);
    //调用OnInitialUpdate()初始化并更新pNewView所指向的新视
    pNewView->OnInitialUpdate();
    //调用函数SetActiveView()将pNewView所指向的新视设置为当前视
    SetActiveView(pNewView);
    //在框架窗口发生改变的时候，通过调用RecalcLayout()来控制控制条的显示方位和行为
    RecalcLayout();
    //调用DestroyWindow()销毁pOldView所指向的旧视
    pOldView->DestroyWindow();
}
}
```

使用的时候，先创建自己的视图2和3，在框架类中添加成员函数SelectViewChange(int

nViewID)，nViewID表示视图标号，在实际使用时，用自己创建的视图名修改该函数中的视图2和3，对应好了视图标号和视图之间的关系后，就可以在需要切换视图的地方调用SelectView Change(int nViewID)，如调用SelectViewChange(2)表示系统切换到视图2。

## 5 文件操作

在一些数据操作较为简单的应用系统中，使用文件操作可使系统紧凑。但是使用文件存储数据，数据的管理就比较困难，当需要修改文件中间的某个数据时，需要将文件中所有的数据都读出来，再将更改好的数据和其他的数据信息重新写入文件，如此达到数据修改的目的。至于数据文件的管理，可以定期删除和备份数据文件，也可以采用同名文件覆盖的办法，也就是说用同名文件只保存最近一次的数据信息。

使用文件存储数据，可以定义FILE类型的文件变量或者是定义MFC提供的类CStdioFile的文件变量。

### 5.1 FILE类型变量的使用

```
FILE *file; //定义变量file为FILE类型的文件指针
if((file=fopen(文件名,"w"))==NULL) // fopen进行打开文件的操作
//w表示对文件进行写操作，r表示进行多操作，a表示进行添加操作
AfxMessageBox("File don't open \n");
// 如果调用fopen()函数后返回的指针为空，说明文件打开操作失败，调用//AfxMessageBox()显示提示信息
else
{
fprintf(file,"%s \n",s);
//fprintf为写操作语句，把变量s中的内容写入文件中
fclose(file); // 关闭文件
}
```

### 5.2 CStdioFile类型变量的使用

```
CStdioFile f; // 声明CstdioFile类型的变量f
f.Open(文件名,CFile::modeRead);
//modeRead表示对文件进行读操作，modeCreate表示创建文件
f.Seek( 0, CFile::begin );
// Seek对文件指针进行定位CFile::begin将文件指针定位到文件头
CString str;//声明字符串类型Cstring的变量str
f.ReadString(str);//ReadString读文件中的内容，并把读出的内容赋给str
```

以上两种类型的变量在程序中可以同时使用。但是在一种情况下必须使用CStdioFile类型的变量：当程序在对文件进行读操作的同时系统对同一文件进行写操作，即出现系统准备读取正在写入文件中的内容，这样的话会造成系统出错，系统不知道到底应该响应哪个操作。为了避免此种情况的发生，可以定义CStdioFile变量对文件写数据，用FILE变量读文件数据。因为CStdioFile变量定义本身具有互斥的性质，如果CStdioFile变量在写数据，它就禁止其他文件变量读该文件的内容。

## 6 刷新

如果系统需要显示实时曲线，则涉及到视图刷新的问题。其中Cdocument::UpdateAllView用于刷新所有的视图，而Cwindow::Invalidate(TRUE)用于更新整个窗体区域。一般来说，曲线的显示只占据了整个窗体或视图的一部分，用"全刷新"的方式容易造成界面的闪烁和抖动，所以建议调用InvalidateRect()来具体刷新显示曲线的那个区域。

```
CRect rect;// 声明Crect类型的变量rect
rect.left=25;rect.right=735;rect.top=80;rect.bottom=450;
//显示实时曲线的区域,该区域左边的位置是25，右边是735，顶部为80，底部为450，该区域的宽度为735-25，高度为450-80
InvalidateRect(rect);//调用InvalidateRect()更新rect所指向的这块区域
```

## 7 与数据采集卡的接口

实现计算机与数据采集卡的信息交换有很多种方式。最直接、省力的一种方式是调用采集卡自带的DLL(动态链接库)中的功能函数来实现。而最简单、快捷的一种方式是利用-inp，-outp直接对数据采集卡的端口进行写和读操作，当然前提是你必须清楚知道该数据的端口号和接口的工作原理。

-inp，-oup包含在conio.h文件中，使用这两个函数时，先要将它们所在的头文件包含进来，这样才可以使用这两个函数的功能。

```
#include <conio.h>
-outp (端口号，数据)；//将数据从端口号中输出
int temp=-inp(端口号)； //读端口号的数据信息送入temp变量中
```

## 8 结论

以上几项程序设计方法已在实际系统设计中得到应用，并经过现场长期运行的考验，证明设计的正确性。可供借鉴。


参考文献

1 David J Kruglinski，Scot Wingo，George Sheperd.Programming Visual C++6.0技术内幕[M]，北京希望电子出版社，1999。

2 斐民、赵栋伟、杨彬等编著，如何使用Visual C++6.0，机械工业出版社，1999，5。

3 侯俊杰著，深入浅出MFC第2版，华中科技大学出版社，1998。