Лабораторная работа № 14
Создание шаблона приложения

Соглашение об использовании имен

Корпорация Microsoft разработала для программистов специальное соглашение об использовании имен в программах. Согласно этому соглашению для функций используются имена, построенные из глаголов и существительных, причем первые буквы этих слов — заглавные. Для имен переменных Microsoft предлагает более сложную систему предусматривающую обозначение именуемых типов данных. Для этого используется небольшой префикс из строчных букв, а собственно имя начинается с заглавной буквы. Далее приведена таблица префиксов, предлагаемых Microsoft.

Таблица Префиксы переменных

Префикс	Типы данных
b	Булевский (байт)
c	Символ (байт)
s	Строка (char или Cstring)
dw	Длинное беззнаковое целое (DWORD)
f	16 битный флаг (битовая карта)
fn	Функция
h	дескриптор(handle)
l	длинное целое (long)
i	данные типа int
lp	длинный указатель (long pointer)
n	Целое (16 бит)
p	Указатель (pointer)
pt	Точка (два 32 битных целых)
w	Целое без знака (WORD, 16 бит)
sz	Указатель на строку, заканчивающуюся 0 (string zero)
lpsz	длинный указатель на sz (long pointer string zero)
rgb	длинное целое, содержащее цветовую комбинацию RGB

Для программ, написанных с применением библиотеки MFC, можно добавить, что имя класса начинается на C, а имя атрибута класса начинается на m_. Причем при создании классов при помощи VC++, автоматически создаются заголовочные файлы и файлы тела класса без буквы C в начале имени.

Использование соглашения об именах в программа не является обязательным, но при использовании соглашений программа не просто становиться легко читаемой, но и значительно облегчается дальнейшее ее сопровождение. Конечно, указывать или не указывать тип переменной в ее названии — это личное дело каждого, но привычка пользоваться таким соглашением является хорошим тоном.

Структура приложения

Так как мы создаем гидропонную систему на языке VC++ с использованием библиотеки MFC, то, для того чтобы в дальнейшем хорошо ориентироваться в созданном коде программы, кратко рассмотрим структуру приложения MFC.

Мастер создания приложений VC++ (AppWizard) может создавать несколько типов приложений, каждое из которых применяется для своих целей. Перечислим эти типы:

- Single document (приложение работает с одним документом);
- Multiple document (приложение работает с несколькими документами);
- Dialog based (приложение основано на окне диалога).

Для нашего случая наиболее удобна структура приложения, работающего одним документом.

Понятие «документ» для тепличного хозяйства

Понятие «документ» широко распространилось в практике программирования с подачи компании Microsoft. Документом теперь принято называть любую совокупность данных, а не только текста. Так, звукозапись или видео фрагмент, помещенные в компьютер, являются документом.

Для создания программ отображения и редактирования документов компания Microsoft создала шаблон работы с документами, где уже предусмотрены основные необходимые функции, такие как отображение в окне сохранение или печать.

Данные о состоянии теплицы также можно представить в виде документа, хотя это не является обязательным. Просто это более быстрый путь для создания работающего приложения, чем путь создания приложения с нуля на базе окна диалога.

Фактически, если принять за основу шаблон документа, то можно за короткое время создать графическую систему управления процессами в теплице, которая будет отображать на экране состояние устройств в графическом виде и подчиняться оператору, который щелчком мыши может изменить состояние устройств или выбрать другую программу выращивания. Так создаются автоматизированные системы управления технологическими процессами (АСУТП).

Классы, создаваемые мастером приложений

При создании шаблона приложения мастер создания приложений создаст код следующих классов.

- главный класс приложения CGreenhouseApp ;
- класс документа CGreenhouseDoc;
- класс просмотра CGreenhouseView;
- класс для окна «О программе» CAboutDlg;
- класс основного окна программы CMainFrame;

Все приложения VMC++ MFC являются объектами. Таким образом, приложение — это главный класс, который включает в себя все необходимые для работы классы документов, окон просмотра. Этот объект является глобальным и создается при вызове приложения.

Обычно Мастер называет его the App. Соблюдая соглашение об именах, Мастер создаст главный класс приложения с именем проекта, к которому прибавит в начале букву С — class (класс), а в конце App — app1ication (приложение). И получится CGreenhouseApp, который наследуется из библиотечного класса CWinApp.

Мастер сразу переопределит функцию класса с CWinApp::InitInstance(), код которой выполняется в первую очередь при загрузке приложения инициализирует все необходимые ресурсы для работы приложения.

Мастер создания приложения создает класс документа (в нашем случает CGreenhouseDoc), в котором должна проходить обработка данных, и наследует его из библиотечного класса CDocument, и класс просмотра (CGreenhouseView), который отображает данные на экране компьютера. Данные отображаются в окне (класс CMainFrame), которого наследуется из библиотечного класса CFrameWnd. Описанная иерархия показана на рис. 105.

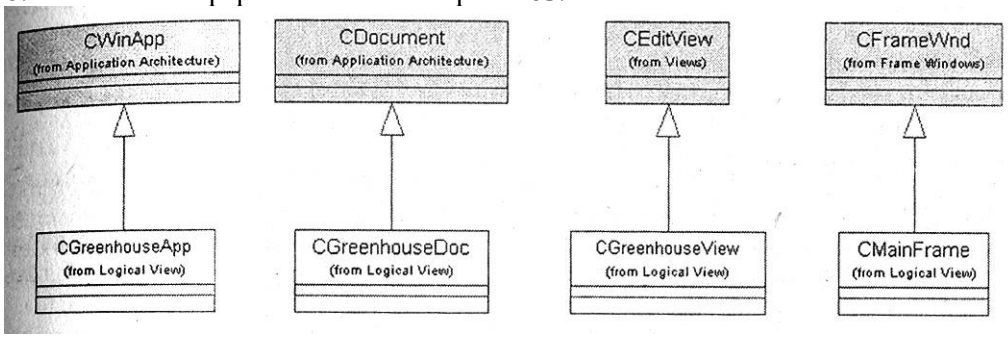

Рис.105. Иерархия классов приложения

Операция OnNewDocument

Важной операцией во вновь созданном классе CGreenhouseDoc будет операция OnNewDocument. Мастером она не создается, но нам необходимо будет ее позднее переопределить. В данной операции будут инициализированы все переменные, которые относятся к документу.

Так как мы будем отображать состояние устройств посредством класса CEditView, то наиболее удобным будет включить обработку события таймера именно в этот класс. При создании обработчика таймера при помощи инструмента Nodel Assistant она получит имя OnTimer.

Таким образом, на основе стандартных классов документа, предоставляемых MFC, мы получим приложение, в котором нам необходимо будет только добавить функциональность, а за отображение документа на экране будет отвечать библиотека.

Теперь, когда понятна структура приложения, необходимо создать его код. Советую создать новый проект VC++, для того чтобы пройти путь создания проекта еще раз и исключить последствия, которые могли остаться от предыдущих экспериментов.

Создание шаблона приложения

Для того чтобы ассоциировать проект Rational Rose с проектом Visual C++ необходимо выбрать окно Menu=>Tools=>Visual C++=>Component Assignment Tool.

В нем мышкой перетаскиваем необходимые классы в строку VC++ на вопрос, хотите ли вы создать компонент VC++, для того чтобы назначить выбранные классы в новый компонент, необходимо ответить «Да».

Все классы, для которых необходимо создание исходного кода, необходимо помещать в программные компоненты. Для простоты работы создадим только один компонент, который заключен в проект greenhouse.

В появившемся окне Select a VC+ Project File нажать кнопку Add, после чего заполнить окно создания нового проекта так, как указано на рис. 106.

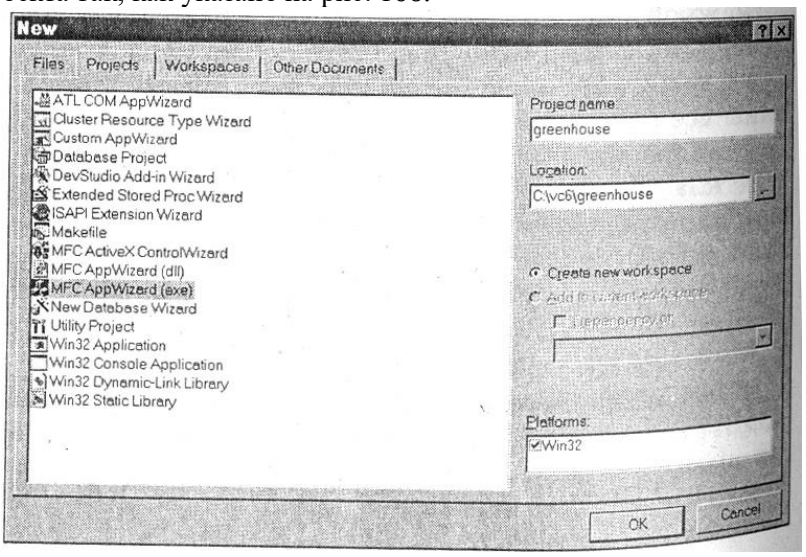

Рис.106. Создание нового проекта VC++

Таким образом, мы воспользуемся мастером создания MFC приложения.

Создадим однооконное приложение обработки документа при помощи мастера, для чего установим кнопку Single document, как показано на 107. Дальнейшие шаги мастера показывать нет необходимости, так как в них можно оставить все предлагаемые параметры без изменений. Пройдите все шаги мастера, последовательно нажимая кнопку Next.

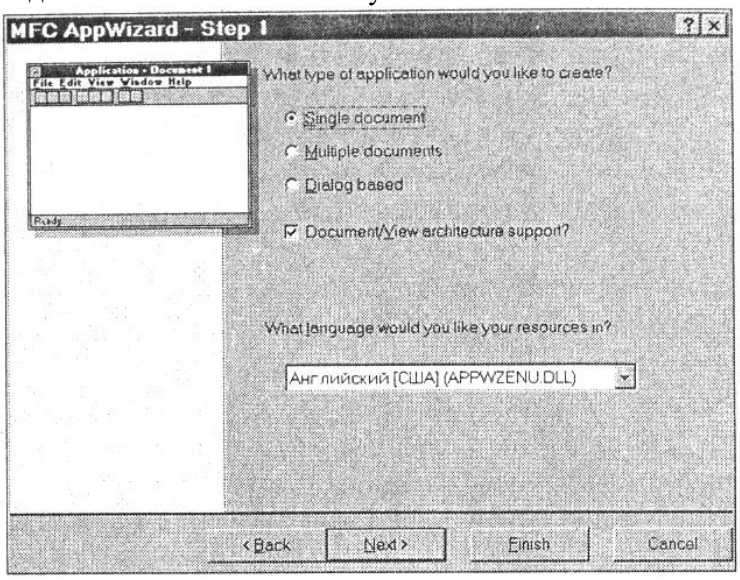

Рис.107. Создание приложения для обработки документа

Однако на шестом шаге, перед тем, как нажать кнопку Finish измените Base class (базовый класс) на CEditView, как показано на рис.108.

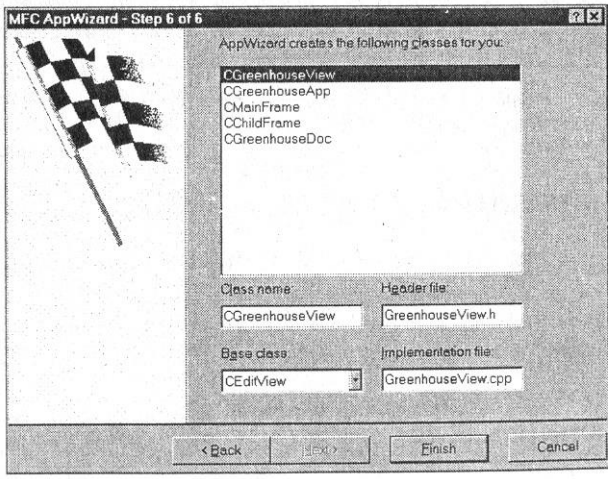

Рис.108. Установка базового класса для шаблона приложения

После завершения всех процессов программа возвратится в окно выбора проекта (рис. 109), где необходимо выбрать только что созданный проект.

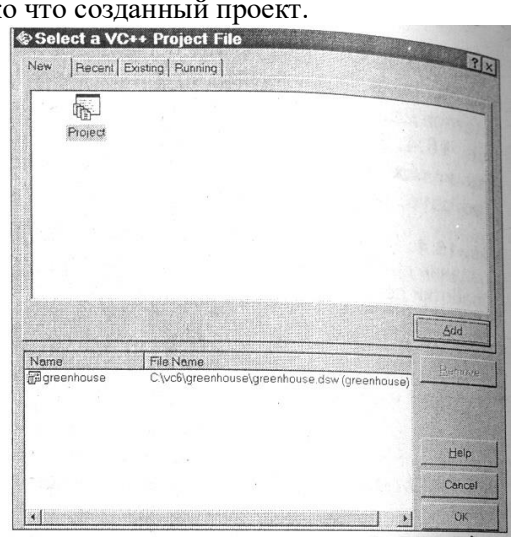

Рис.109. Окно выбора проекта после создания проекта greenhouse

Назначение классов в проект

Нажмите ОК, и проект Visual C++ готов. Теперь можно назначить классы в проект, название которого появилось под надписью VC++ в окне Component Assignment tool.

Для этого перетаскиваем мышкой необходимые классы в проект. Для того чтобы получить первый вариант работающего приложения, пока включите в проект только классы устройств. Постепенно мы будем расширять проект, но пока для быстрой отладки будем работать только с классами устройств.

Импорт библиотеки MFC

Для того чтобы можно было в дальнейшем работать с классами MFC, необходимо импортировать эту библиотеку классов в модель. Выберите Menu =>Tools=>Visual C++=>Quick Import MFC 6.0. После этого появится сообщение о загрузке классов в модель.

Загрузка созданных классов в модель

Теперь в модель Rational Rose Можно загрузить классы, созданные Visual C++. Это классы, о которых мы уже упоминали:

- главный класс приложения CGreenhouseApp ;
- класс документа CGreenhouseDoc;
- класс просмотра CGreenhouseView;
- класс для окна «О программе» CAboutDlg;
- класс основного окна программы CMainFrame;

Для того чтобы они появились в проекте, необходимо обновить проект по готовому коду при помощи следующей последовательности действий: Menu=>Tools=>Visual C++=>Update Model From Code.

Сейчас это делать не обязательно, потому что в код необходимо внести некоторые изменения для нормальной компиляции, однако, для того чтобы посмотреть, как будут отображены классы в проекте Rational Rose, вы можете обновить модель из кода.

После обновления Rational Rose автоматически создаст диаграмму, которая отражает структуру классов, наследуемых из классов библиотеки MFC.

Главное при этом не удаляйте классы, которые Rational Rose не найдет в полученном коде. Их там еще нет, и для того чтобы они появились необходимо обновить код по модели: Menu=>Tools=>Visual C++=>Update Code.

После того как обновление прошло, мы получим одинаковые классы в модели Rational Rose и в проекте VC++.

Первый запуск приложения

Запустите Visual Studio (если эта среда разработчика еще не запущена) и перейдите в проект greenhouse.

Нажмите F7 для создания EXE файла. В полученный при помощи Rational Rose код необходимо внести небольшие изменения для создания выполняемого файла. Изменения внесенные один раз, запоминаются, и при обновлении модели нет необходимости вносить их повторно.

Загляните в протокол. Самой распространенной ошибкой является «fatal error C1010: unexpected end of file while looking for precompiled header directive». Это означает, что необходимо включить файл “stdafx.h” во все файлы, где возникла такая ошибка.

Откройте эти файлы и добавьте строку #include “stdafx.h” перед другими операторами #include. После завершения этой работы опять запустите генерацию.

К сожалению, генератор кода Visual C++ в Rational Rose не поддерживает ключевые слова typedef и enum, поэтому необходимо определить данный тип непосредственно в исходном коде, например, создать включаемый файл с определение типов.

Создадим непосредственно в проекте VC++ файл defs.h: Project=>Add to project=>New (рис.110).

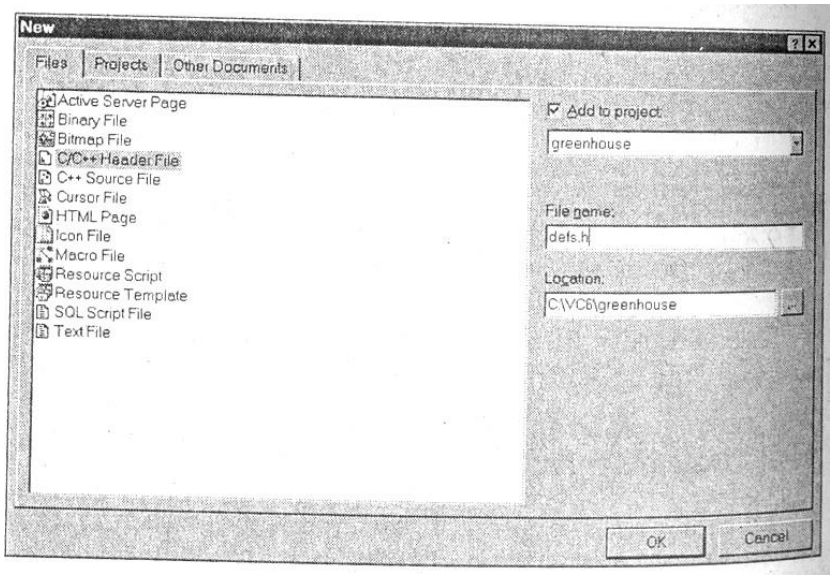

Рис.110.Добавление нового заголовочного файла в проект ‘IC++

В редакторе откроется окно для ввода, в которое введем следующие определения типов.

```
#pragma once //для того, чтобы файл включался один раз
//состояние устройства включено или выключено
eEnum DeviceState{Off,On};
// тип температуры лучше сделать простым типом, а не классом
typedef float Temperature;
//время дневное или ночное
 enum Lights{day,night};
//тип для описания pH
typedef float pH;
```

//типы для описания текущего времени, часа и минуты
typedef unsigned int Day;
typedef unsigned int Hour;
typedef unsigned int Minute;

Конечно, можно обойтись и без переопределений. Ведь для компилятора слово typedef обозначает только синоним стандартных типов. Но для написания программ определение типов конкретной предметной области удобно для дальнейшего сопровождения программы да и при разработке удобно иметь дело не с безличным float, а с конкретным Temperature.

После того как файл определений создав, его можно включить в модули, где используются эти определения. Так как мы создали определение температуры, то класс Temperature нам больше не нужен, и его можно удалить из модели.

Теперь в некоторых классах есть операции, которые возвращают значения. Пока для них созданы только шаблоны, поэтому пока текста нет, в такие операции поставим с любым значением подходящего типа.

После завершения этих предварительных операций еще раз запустим компиляцию. Теперь она должна пройти без ошибок. Запускаем полученный файл (CTRL+F5) и видим изображение, показанное на рис.111.

Что мы получили при первом запуске программы? Мы получили Windows приложение, которое пока не выполняет необходимых нам функций. Однако это уже полноценное приложение, включающее шаблоны разработанных нами классов, которые осталось только довести до рабочего состояния

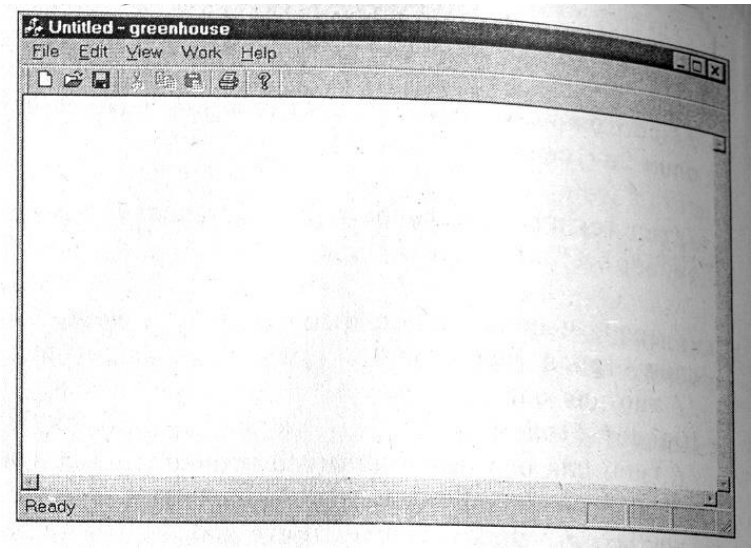

Рис. 111.Экран программы GreenHouse.exe