Лабораторная работа № 13
Создание кода класса на Microsoft Visual C++

Вступительные замечания

Мы ознакомились с возможностями создания кода класса на языке C++, отвлеченном от конкретного компилятора. Как обычно, более универсальный подход является и более сложным для создания необходимого кода приходится оперировать большим количеством установок.

Более простой путь для создания приложения в использовании возможности Rational Rose — создавать код класса на основе библиотеки классов фирмы Microsoft — MFC. Для этого нет необходимости вручную оперировать значительным количеством установок, так как в пакет встроен модуль Model Assistant, который позволяет изменять все необходимые установки при помощи визуальных средств.

Возможности создания кода класса

Класс в Rational Rose— это описание общей структуры (данных и связей) для дальнейшего создания объектов. для того чтобы генератор Rational Rose имел возможность создавать на основе описанной модели программный код, для каждого класса необходимо указать язык, для которого будет создаваться код. Также необходимо определить компонент, в котором этот класс будет храниться. Если в качестве языка для создания кода указан VC++, то пользователь получает доступ ко всей иерархии классов библиотеки MFC при помощи визуальных средств Model Assistant.

При создании класса необходимо указать стереотип, который влияет на получаемый исходный код класса. Так, при изменении стереотипа на struct или union, будут созданы указанные типы данных.

Как вы уже заметили, Rational Rose поддерживает обычные для классов C++ обозначения области видимости, такие как public, private, protected.

Таким образом, каждый атрибут или операция в спецификации классов при создании заголовочного файла класса будут определены в одну из секций public, private или protected. Также имеется возможность не создавать программный код для определенных классов.

Структура создаваемого кода классов

Для каждого создаваемого класса Rational Rose создает следующую структуру кода:

- директивы include, которые создаются исходя из необходимости включения атрибутов и связей классов;
- декларация класса, имя класса, тип, наследование;
- переменные Data members, которые создаются по описанию атрибутов класса и его связей;
- декларация методов класса и скелет этих методов для дальнейшего наполнения каждой операции, заданной в описании класса;
- документация для каждого создаваемого класса, переменных, методов, заданная в описании модели;
- идентификатор ID модели, который включается в код как комментарий для каждого создаваемого класса, атрибута или метода, заданных в текущей модели.

Ассоциация класса с языком VC++

Для того чтобы использовать класс в программном проекте, необходимо его ассоциировать с выбранным языком, в нашем случае с VC++. Для этого проделаем следующее Menu:Tools=>Visual C++=>Component Assigned Tools. Получаем окно, показанное на рис. 94.

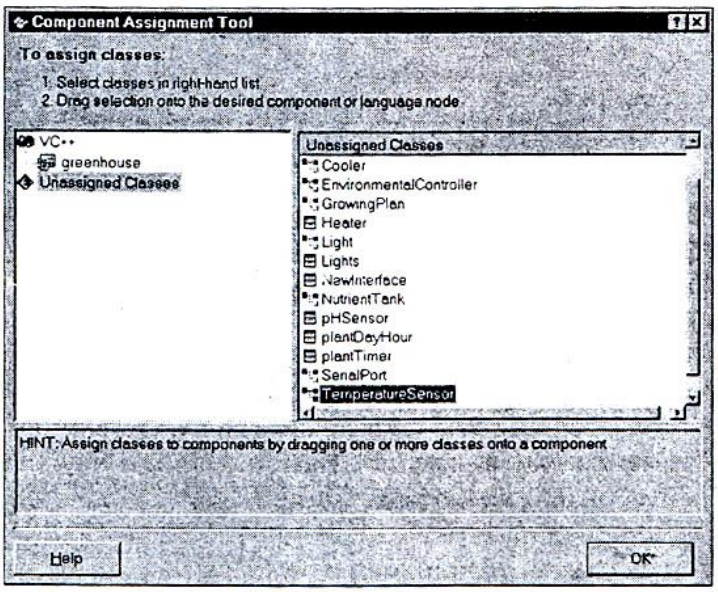

Рис. 94. Назначение класса в VC++ проект

В появившемся окне выбираем класс и перетаскиваем его на значок VC++. На вопрос, желаем ли мы создать VC++ компонент и ассоциировать его с классом, отвечаем Yes и попадаем в окно выбора проекта VC++ (рис. 95). Здесь можно создать проект или выбрать из уже имеющихся для помещения в него нового класса. Нажмите Add и OK.

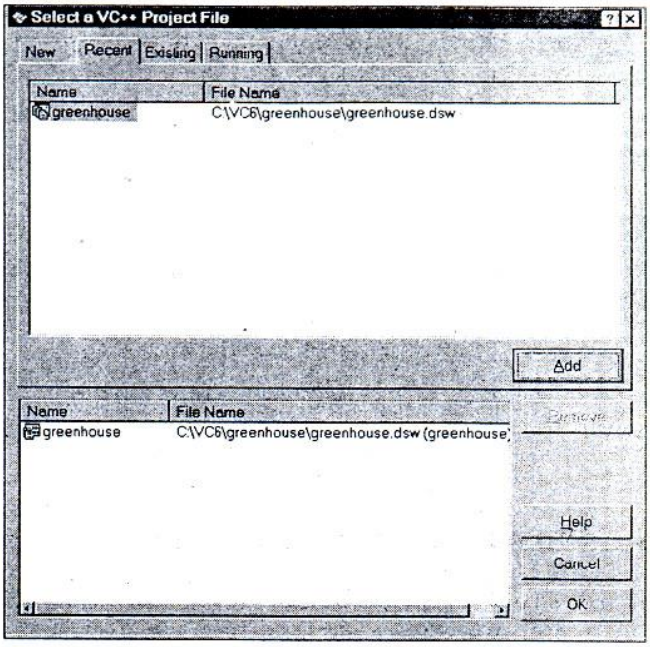

Рис.95.Выбор проекта

Вы у себя можете создать проект с именем greenhouse при помощи MFC App Wizard exe (мастер создания исполняемого приложения) с типом Single document (однооконный документ).

По большому счету, класс может быть ассоциирован с любым языком, поддерживаемым генератором кода Rational Rose, и в результате ассоциации будет создан код программы на том языке, с которым ассоциирован класс, причем от конкретного языка зависят некоторые свойства класса. Поэтому неплохой идеей будет изначальная ассоциация классов с определенным языком программирования.

Меню инструментов Visual C++

После ассоциации класса с языком программирования вы можете воспользоваться пунктом главного меню Tools для Visual C++, показанном на рис. 96.

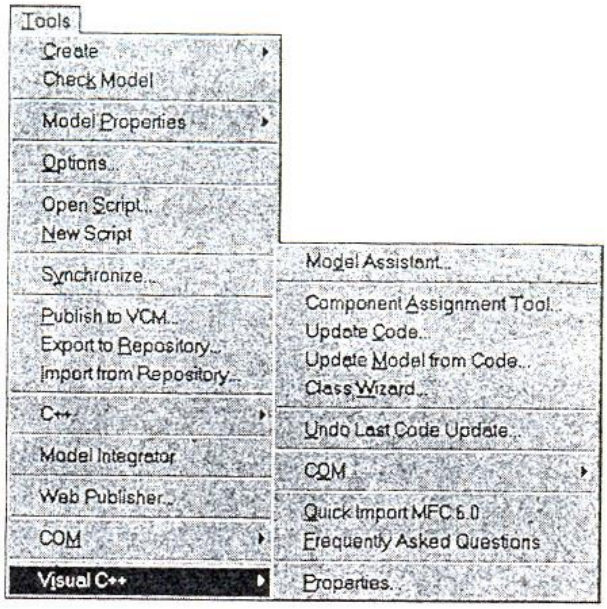

Рис.96. Меню инструментов Visual C++

Model Assistant

Model Assistant позволяет обновлять и конкретизировать классы в модели, используя дополнительные ключевые слова C++ для необходимой генерации кода. Model Assistant представляет собой окно, позволяющее создавать атрибуты и операции и изменять их свойства (рис. 97), причем значительно проще и нагляднее, чем это происходит для C++.

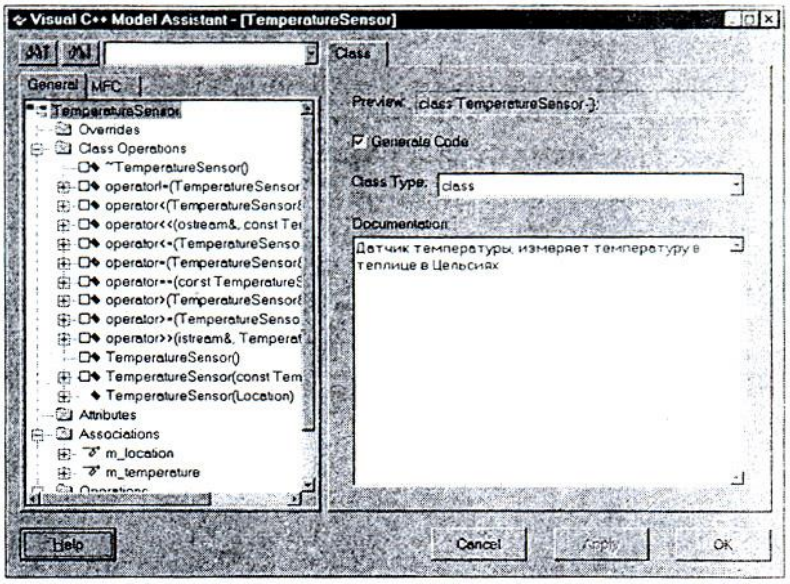

Рис.97. Model Assistant для класса датчика температуры

В этом окне вы видите следующие поля:

- Preview (предварительный просмотр) показывает описание класса так, как оно определено в текущий момент;
- Code Name (имя программы) показывает имя программного файла для данного класса;
- Generate Code (создавать исходный текст) — ключ, определяющий необходимо ли создавать для данного класса исходный текст на языке VC++. Если ключ снят, то

генерация кода не будет происходить и этот класс не будет показываться в списке классов для обновления кода в окне Code Update (обновление исходного текста);

- Class Type (тип класса) позволяет установить тип класса, такой как «class», «struct» или «union»;
- Documentation (документация) позволяет задавать произвольные комментарии для класса. Если это поле заполнено, то при генерации исходного кода текст из него будет включен в программу как комментарии. Это чрезвычайно удобно при просмотре программистом кода, созданного при помощи генератора Rational Rose. По крайней мере, данная возможность позволяет создавать документацию к программе непосредственно в момент создания класса.

Rational Rose позволяет создавать комментарии еще на этапе проектирования, пока не написано ни строки кода, что довольно удобно, потому что именно в момент проектирования класса еще не забыты те цели и ограничения, с которыми создается проектируемый класс или метод.

Значительные возможности предоставляет Rational Rose по интерактивной установке свойств методов класса. Рассмотрим свойства операции calibrate, для чего активизируем строку calibrate в окне Model Assistant. При этом программа активизирует диалоговое окно, показанное на рис. 98. Это окно позволяет устанавливать и изменять атрибуты для операции, а также перегружать определенные в классе операции.

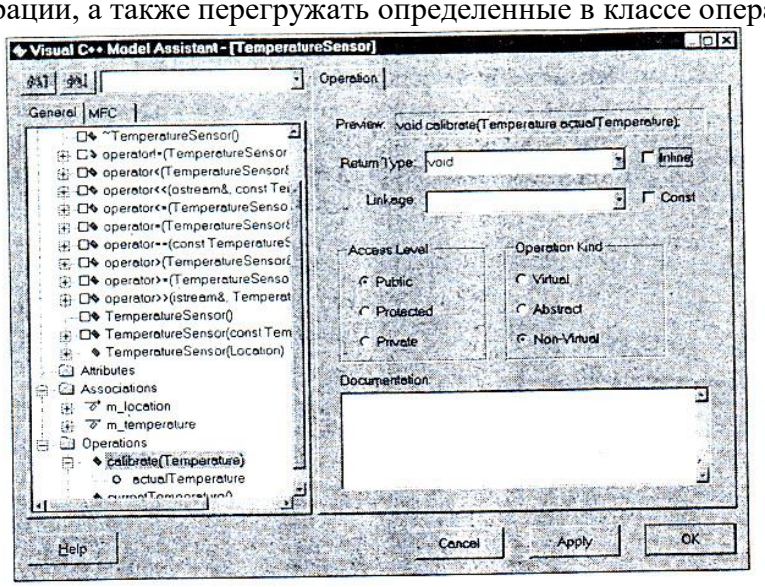

Рис. 98. Свойства операции calibrate

Назначение полей а окне следующее:

- Preview (предварительный просмотр) показывает описание операции таким образом, как оно было определено в текущий момент;
- Code Name (наименование кода) показывает имя кода для операции. Может быть скрыто, если такое имя не задано;
- Return Type (тип) позволяет выбрать из списка тип, возвращаемый операцией;
- Linkage (присоединение) позволяет установить тип операции, который может быть Friend или Static:
    1. Static обозначает, что к данной операции можно обращаться еще до создания объекта класса;
    2. Friend определяет, что данная функция хоть и не является членом класса, но имеет доступ к его защищенным и собственным компонентам. Таким образом, определяя операцию как дружественную, мы тем самым удаляем ее из методов класса и подразумеваем, что данная функция будет описана вне класса.

- Inline позволяет указать в операции ключевое слово inline, то есть операция будет создана как inline подстановка. В этом случае компилятор при создании объектного кода будет стараться подставить в текст программы код операторов ее тела. Таким образом, при многократных вызовах подставляемой функции размеры программы могут увеличиваться, однако исключатся, затраты на передачу управления вызываемой функции и возвраты из нее. Как отмечает проект стандарта C++, кроме экономии времени при выполнении программы, подстановка функции позволяет проводить оптимизацию ее кода в контексте, окружающем вызов, что в ином случае невозможно Наиболее удобны для подстановки функции, состоящие всего из нескольких строк;
- Const определяет тип операции как Const.
- Access Level (уровень доступа) указывает доступ к операции и может быть Public, Protected или Private;
- Operation Kind — тип операции Virtual, Abstract или Non-Virtual.

К механизму виртуальных функций обращаются в тех случаях, когда необходимо в базовый класс поместить функцию, которая должна по-разному выполняться в производных классах. Например, базовый класс может описывать фигуру на экране без конкретизации ее вида, а производные классы уже описывать реализацию конкретных треугольников, эллипсов, квадратов и т.д. При этом класс, который содержит хотя бы одну виртуальную функцию, называется абстрактным, в данном случае нет разницы между установкой пункта Virtual или Abstract. И в том, и в другом случае будет создана функция с ключевым словом Virtual, которая потребует переопределения в производных классах или как минимум создания дочерних классов для класса, имеющего виртуальную функцию.

Если щелкнуть по атрибуту класса, то открывается окно для редактирования свойств этих атрибутов, показанное на рис.99, в котором можно установить основные атрибуты класса, не выходя из Model Assistant.

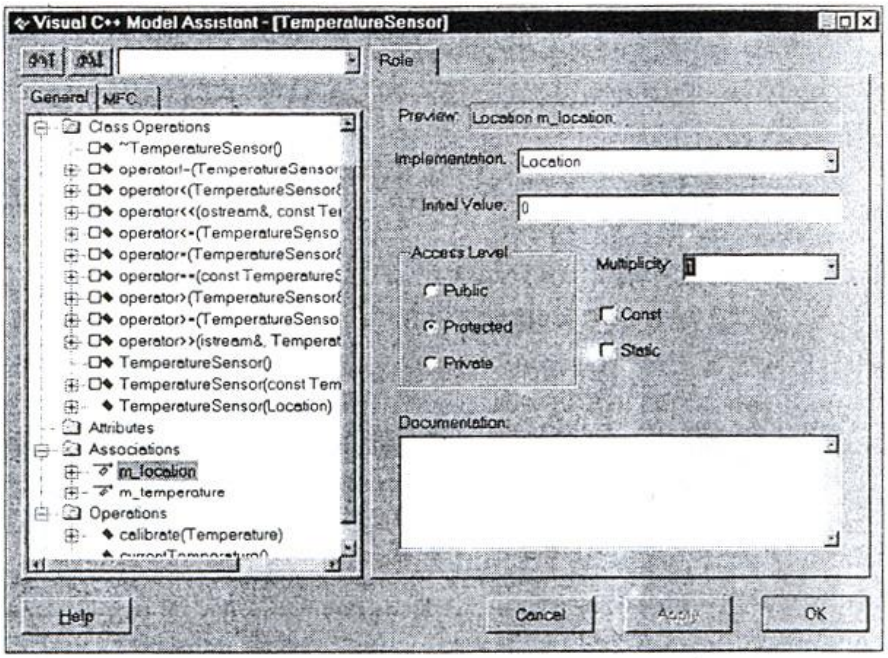

Рис.99. Редактирование атрибутов класса при помощи Model Assistant

В данном окне имеются следующие поля:

- Preview (предварительны просмотр) показывает описание атрибута таким образом, как оно было определено в текущий момент;
- Type (тип) позволяет выбрать из списка тип атрибута;

- Initial Value (инициализация) позволяет устанавливать значение для работать именно в том средстве, которое позволяет наиболее быстро получит необходимый результат;
- Access Level (уровень доступа) указывает доступа к атрибуту и определяет секцию, в которой будет создан атрибут: Public, Protected или Private;
- Static обозначает, что атрибут является статичным (общим для всех объектов класса);
- Derived обозначает, что данный атрибут является производным;
- Documentation позволяет редактировать описание атрибута.

В этом окне есть вкладка MFC, которая предназначена для классов, наследуемых из базовых классов MFC. Так как датчик температуры не наследуется ни из каких классов, то эта вкладка нам пока не понадобится, мы рассмотрим ее свойства, когда будем создавать приложение на основе классов MFC.

Component Assignment Tool

Активизирует диалоговое окно назначения классов в компоненты и назначения языка для класса. Это окно предоставляет возможность создания новых компонентов в модели, ассоциации компонентов с проектами на конкретных языках программирования и назначения классов в компоненты. Для того чтобы получить преимущества использования данного инструмента, необходимо создавать компоненты здесь, а не через окно Browser или в диаграмме компонентов. При этом созданные компоненты будут содержать необходимую информацию для генерации кода на выбранном языке программирования. Данное средство позволяет просмотреть классы, которые еще не назначены в компоненты, что уменьшает вероятность ошибки.

Component Assignment Tool может быть открыт как посредством Tools, так и из контекстного меню компонента в окне Code Update Tool.

Update Code/ Update Model (обновить код/модель)

Данная возможность создать проект Visual C++ по разработанной модели обновить модель по уже готовому проекту, созданному при помощи MFC.

Причем, что очень приятно, есть возможность не просто закачать уже готовый код в программу Rational Rose, как это предусмотрено пунктом меню C++ Reverse Engineering, а поддерживать обмен постоянно, то есть работать именно в том средстве, которое позволяет наиболее быстро получить необходимый результат.

Например, необходимо быстро подправить что-либо в уже готовой, работающей программе, причем, как всегда бывает, это необходимо было сделать «еще вчера». Программист быстро «подкручивает» что-то в исходном коде добавляет или изменяет методы и атрибуты и быстро сдает работающую программу. Затем просто выбирает пункт Update Model from code, и эти изменения тут же попадают на рабочий стол Rational Rose. Теперь с ними можно нормально работать и сопровождать.

Ведь не секрет, что сопровождение — это кошмарный сон программиста. Часто проще написать код заново, чем разбираться в многочисленных « заплатках», сделанных в разное время и иногда даже разными людьми.

Здесь же «заплаток» нет принципиально. Вся программа составляет одну обозримую модель.

Допустим, что у нас уже создан проект greenhouse (у вас он должен уже быть, если все сделано правильно), и мы знаем, что после того как последний раз изменялась модель, исходный код класса датчика температуры был исправлен, и его необходимо обновить.

Выберем пункт Update Model from code. Причем в контекстном меню класса, ассоциированного с VC++, также есть этот пункт, только он называется Update Model, что, впрочем, не имеет никакого значения, так как действия, выполняемые этими пунктами,

одинаковы. После выбора появится окно с описанием дальнейших действий, его можно погасить, выбрав кнопку Next (дальше), и, к тому же, еще предотвратить его назойливое появление в дальнейшем путем установки флажка в поле Don't show this page in the future (больше не показывать эту страницу). После этого появится окно, показанное на рис. 100.

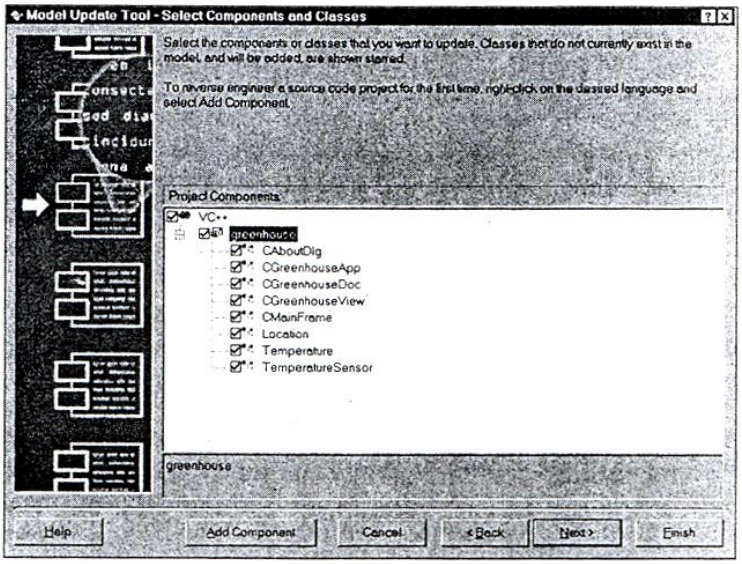

Рис.100. Окно Update Model Tool

Здесь можно обновить как все классы модели, так и отдельные классы, при помощи установки и снятия отметок с определенных классов.

Если классы модели еще не ассоциированы ни с одним проектом VC++, то при помощи кнопки Add Component (добавить компонент) это можно сделать прямо из данного окна.

Некоторые классы проекта могут иметь ошибки или недостаток данных при создании их в Rational Rose с последующей генерацией кода или же при переносе данных из готового кода в модель Rational Rose. Такие классы отмечаются знаком вопроса в ярко желтом кружочке. Если выбрать такой знак вопроса, то появится сообщение Rational Rose, которое указывает на возникшую проблему или ошибку в классе, и предлагает методы ее устранения.

Затем Rational Rose получает информацию из проекта Visual C++, для этого загружается Microsoft Visual Studio и активизируется нужный проект Visual C++. После того как обмен произошел, может быть активизировано окно удаления компонентов. Если вы проводили эксперименты с полученным кодом, а затем удалили некоторые классы, операции или атрибуты, то программа отследит, что элементы существуют в модели Rational Rose, но никак не отражены в исходном коде. Программа считает, что эти компоненты были удалены из исходного кода и предлагает их удалить и из модели (рис.101).

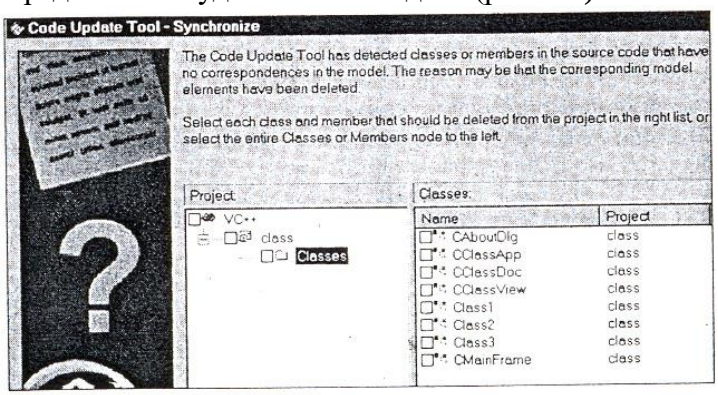

Рис.101. Синхронизация кода и модели

Внимательно отнеситесь к этому вопросу. Возможно в проект Visual C++ были добавлены классы, например, при помощи Class Wizard, и они еще не отражены в модели Rational Rose. В этом случает необходимо провести обновление кода при помощи функции Update Code (обновить код). У нас именно такой случай, и вы можете ничего не удалять, если не будете устанавливать флажки на предложенных компонентах.

После завершения обмена программой будет представлен отчет о том, как прошло обновление. В окне имеется две вкладки:

Summary — краткая общая информация и Log — полный отчет об обновленных классах (рис. 102). Если все прошло нормально, то ошибок и предупреждений быть не должно, как показано на рисунке. Фатальная ошибка будет возникать, если на компьютере установлен пакет Visual Studio 5, а не 6, с которым работает Rational Rose 98i-2000.

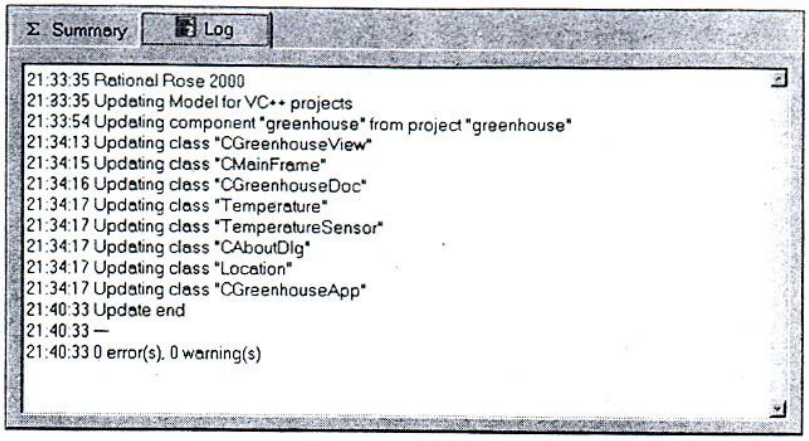

Рис.102. Отчет о проведенных обновлениях компонентов

При этом будет возникать ошибка обновления, и процесс завершится неудачно.

Процесс обновления кода по изменениям в модели происходит аналогично. Причем Rational Rose позволяет выбрать конкретные классы, которые необходимо обновить.

Class Wizard (мастер создания класса)

Class Wizard помогает создавать новые классы. Как и большинство мастеров, он посредством последовательно активизируемых окон ведет пользователя к созданию класса с необходимой информацией. Мастер предоставляет возможность устанавливать три различных варианта создания классов:

1. Создание нового, пустого класса.
2. Создание подкласса уже созданного класса.
3. Создание нового класса по шаблону уже существующего

Undo Last Code Update (отмена последнего обновления)

Пункт Undo Last Code Update позволяет активизировать диалоговое окно, где вы можете заменить полученный исходный код на его предыдущую версию (рис.103).

COM

Активизирует меню установок объектов COM. Мы не будем рассматривать это меню по причине того, что в нашей системе таких объектов нет.

Quick Import MFC 6.0

Позволяет импортировать классы библиотеки классов MFC в текущую модель, для того чтобы ими можно было воспользоваться при создании иерархии приложения.

Properties

Позволяет устанавливать свойства генератора Visual C++, которые влияют на создаваемый код. Мы не будем на них подробно останавливаться по причине того, что для нашей задачи достаточно установок по умолчанию.

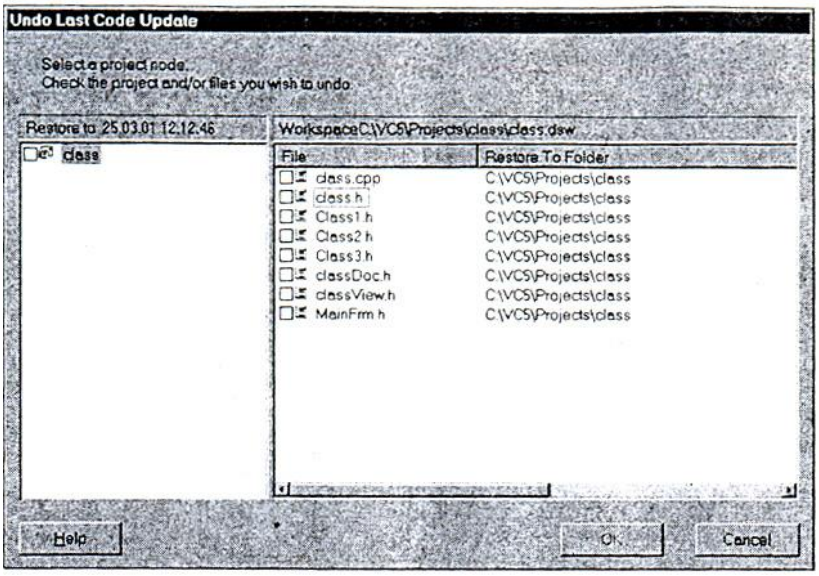

Рис.103. Диалог отмены последнего обновления

Создание кода класса

Как вы заметили, Visual Studio 6.0 с Rational Rose, Rational Rose является прекрасным дополнением к Visual Studio. Таким образом, для создания кода класса достаточно воспользоваться средством Model Assistant для установки необходимых свойств генерации и провести обновление кода при помощи команды Update Code. Приемы работы средством Model Assistant несколько отличаются от установки свойств при создании кода C++, поэтому будем рассматривать их не отвлеченно, а на примере создания кода гидропонной станции.

Процесс создания кода — итерационный процесс, в течение которого для получения работающего приложения нам придется вносить изменения как в модель, так и в полученный исходный код.