Л.Г. Ишкова, преподаватель спецдисциплин по специальности 090203 ГБПОУ «Тверской колледж имени А.Н.Коняева».

**Использование метода модерации при проведении лабораторной работы на тему: «Отладка программ».**

**Moderare –** в переводе с латинского – приводить в равновесие, управлять, регулировать.

Эффективность модерации определяется тем, что используемые приёмы, методы и формы организации познавательной деятельности направлены на активизацию аналитической и рефлексивной деятельности обучающихся, развитие исследовательских и проектировочных умений, развитие коммуникативных способностей и навыков работы в команде. Методы модерации отличаются от автократичных дидактических способов обучения. Преподаватель и студенты являются равноправными участниками образовательного процесса. От каждого из них в равной мере зависит успех обучения. Такой подход формирует у обучающихся самостоятельность в выработке и принятии решений, готовность нести ответственность за свои действия, вырабатывать уверенность в себе, целеустремлённость. При использовании технологии модерации принципиально меняется и роль преподавателя. Он становиться консультантом, наставником, старшим партнёром, из «контролирующего органа» педагог превращается в более опытного товарища, играющего в одной команде с обучающимся. Цели применения модерации – эффективное управление группой в процессе урока, максимально полное вовлечение всех участников в образовательный процесс, поддержание высокой познавательной активности обучающихся на протяжении всего занятия, гарантированное достижение целей урока.

**Принципы и фазы модерации**

Для обеспечения эффективности управления образовательным мероприятием и гарантированного достижения запланированных целей урока в основу технологии модерации положены следующие ключевые принципы:

- структурированность (все содержание урока рационально делится на четко определенные части);

- систематичность (отдельные части урока взаимосвязаны и логически следует одна за другой, создавая полноценное содержание урока);

- комплексность (содержание каждой части урока, и организуемые процессы нацелены на обучение, воспитание, развитие и социализацию обучающихся);

- прозрачность (деятельность каждого обучающегося видна учителю, всем участникам ясно виден ход образовательного процесса, его промежуточные и итоговые результаты).

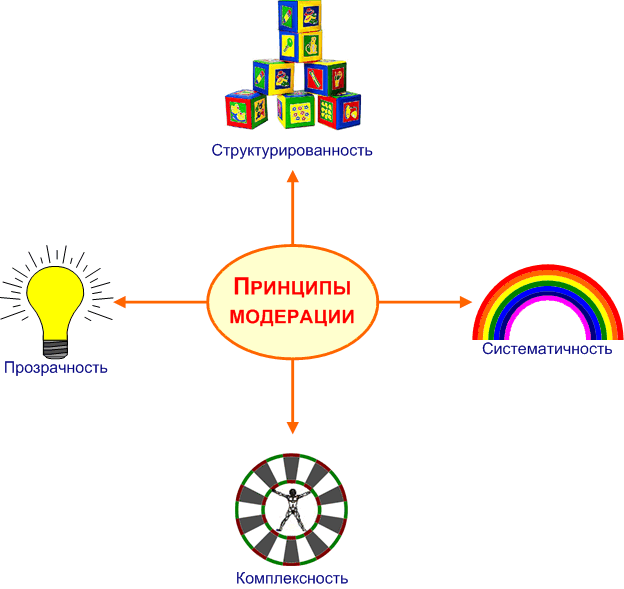


Рисунок 1. Принципы модерации.

Структурированность образовательного процесса обеспечивается разделением урока на определенные взаимосвязанные фазы (этапы, части), каждая из которых имеет свои цели, задачи и методы, позволяет создать ясный и четкий план образовательного мероприятия, задать направленное поступательное движение к поставленным целям урока, обеспечить методичную проработку каждой фазы и последовательность переходов от одной фазы урока к другой, осуществлять эффективный мониторинг хода и результатов образовательного процесса. Фазы (этапы) модерации:

- инициация (начало урока, знакомство);

- вхождение или погружение в тему (сообщение целей урока);

- формирование ожиданий учеников (планирование эффектов урока);

- интерактивная лекция (передача и объяснение информации);

- проработка содержания темы (групповая работа обучающихся);

- подведение итогов (рефлексия, оценка урока);

- эмоциональная разрядка (разминки).

Каждая фаза – это полноценный раздел образовательного мероприятия. Объём и содержание раздела определяется темой и целями урока. Достижение эффективности и качества образовательного процесса при использовании технологии модерации, получение запланированных результатов обучения, обеспечивается организацией следующих ключевых процессов:

* эффективное взаимодействие (интеракция) участников группового процесса;
* упорядоченный обмен информацией (**к**оммуникация) между всеми участниками образовательного процесса;
* обеспечение наглядности хода и результатов образовательного процесса (**в**изуализация);
* **м**отивация всех участников образовательного процесса;
* **м**ониторинг образовательного процесса;
* рефлексия педагога и обучающихся;
* анализ деятельности участников и оценка результатов.

**Процесс модерации (начало) – «*Интеракция»***

Для организации эффективного взаимодействия обучающихся необходимо сформировать малые группы, в которых в дальнейшем пойдёт вся основная работа. В процессе совместной работы обучающихся будут осуществляться различные интеракции: учитель-ученик, ученик-ученик, учитель-малая группа, учитель-класс, малая группа-малая группа, малая группа-класс, ученик-малая группа-ученик-класс.

**Процесс модерации – «*Коммуникация»***

Коммуникация представляет собой средство для обмена информацией, знаниями, а также ожиданиями. Настроениями, чувствами, которые передаются партнёру в разговоре или при невербальном общении. Целью построения коммуникативного процесса должен стать эффективный обмен информацией между его участниками.

**Процесс модерации – «*Визуализация»***

 Для эффективной реализации такого принципа модерации как прозрачность, который позволяет педагогу и обучающимся ясно видеть весь ход образовательного процесса, его промежуточные и итоговые результаты, необходимо спланировать и осуществить еще один ключевой процесс – визуализацию. Визуализация подаваемого материала обеспечивает наглядность, чёткое восприятие и понимание, возможность многократного обращения к представленной информации, возможность сравнения предыдущей и последующей информацией.

В процессе представления и обсуждения материала урока учителю очень хорошо видно, как обучающиеся усвоили материал, всё ли правильно они поняли, что позволяет педагогу прямо в процессе обсуждения внести при необходимости коррективы в понимание новой темы, дополнительно объяснить проявившиеся сложные моменты темы.

**Процесс модерации – «*Мотивация»***

Поддержание тонуса, развитие уверенности в своих силах и позитивного настроя всех участников образовательного процесса, стимулирование познавательной активности и творческого процесса, раскрытие и развитие способностей обучающихся, содействие их эффективному сотрудничеству – катализаторам этих процессов является мотивация. Постановка, в соответствии с технологией модерации, ясных и понятных целей урока, выяснение ожидания обучающихся, учёт этих ожиданий делает процесс обучения понятным и желанным для обучающихся. Поставленные цели сами по себе обладают притягивающим, мотивирующим действием. Признание равноправной роли ученика в образовательном процессе несет мощнейший мотивационный потенциал. Работа в команде – ещё один мощный мотивирующий фактор.

**Процесс модерации – «*Мониторинг образовательного процесса»***

 Проведение мониторинга включает в себя отслеживание и сверку получаемых результатов каждого раздела урока с запланированными, а также формальное и неформальное фиксирование хода и результатов образовательного процесса и внесение, при необходимости, корректив по ходу урока. В процессе осуществления модерации очень важно обеспечить мониторинг выполнения плана урока и групповой динамики. Для усиления структурированности урока и упорядоченности формирующихся знаний обучающихся, важно каждый этап урока завершить визуализацией результата.

**Процесс модерации – «*Рефлексия»***

 Рефлексия – осмысление новых знаний, умений, качеств и ценностей, критический анализ информации, генерация ответов на вызовы окружения, а также самооценка себя, своего поведения, своей роли, своего вклада в процессе групповой работы, корректировка своей деятельности на основе этой оценки и потребностей группы – обязательный атрибут современного образования.

Урок закончен, надо подвести итоги, обеспечить обучающихся обратной связью, создать у всех участников ощущение завершённости процесса.

По итогам работы надо обязательно четко сформулировать полученные результаты и отметить их как важные шаги на пути к поставленным целям. Достижение результата – это успех, который придает уверенности в своих силах, настраивает на следующие достижения. Завершение урока положительной оценкой его результатов – это мотивирующий переход к следующим урокам, к следующим классам.

Далее рассмотрим применение метода модерации при проведении лабораторной работы по дисциплине «Инструментальные средства разработки программного обеспечения» на тему «Отладка программ».

В таблице представлен сценарий лабораторной работы:

Таблица 1. Сценарий лабораторной работы.

|  |  |
| --- | --- |
| **Объявление темы урока**: « *Средства отладки программ*».  **Постановка цели урока**: *Ознакомиться с инструментами, приёмами выявления и исправления ошибок в программе. Приобрести умения и навыки отладки программ.*  **Формирование ожиданий учеников.**  **Ознакомление с теоретическими основами.** *Ознакомление с окном вывода ошибок. Переход от сообщения об ошибке к месту в программе. Трассировка программы. Установка точек останова* **Выполнение всей группой задания по образцу. Мониторинг.**  **Визуализация результатов.**  **Рефлексия. Осмысление приобретённых знаний.** *Ответы на контрольные вопросы.*  **Разбиение группы на подгруппы. Р***азбить на 4 группы.*  **Объявление задания для самостоятельного выполнения каждой подгруппе.** *Каждой подгруппе предложено исправить по одной ошибке.*  **Выполнение задания подгруппами. Мониторинг.**  **Визуализация результатов каждой подгруппой.**  **Рефлексия – осмысление приобретённых знаний.**  **Подведение итогов.** | **2мин.** |
| **3мин.** |
| **3мин.** |
| **18мин.** |
| **18мин.** |
| **5мин.** |
| **7мин.** |
| **2мин.** |
| **2мин.** |
| **18мин.** |
| **7мин.** |
| **3мин.** |
| **2мин.** |
| **Итого:** | **90мин.** |

**Конспект лабораторной работы**

**Цель лабораторной работы**: *Ознакомиться с инструментами, приёмами выявления и исправления ошибок в программе. Приобрести умения и навыки отладки программ в среде TURBO C++.*

**Оборудование:** *ПК платформы IBM PC.*

**Теоретическое введение**

При разработке программ, особенно в первое время, возможны ошибки как синтаксические, так и логические. Чем сложнее программа, тем труднее их обнаружить. Дня облегчения обнаружения и устранения ошибок в среде разработки Turbo C++Builder существует встроенный отладчик программ. Он позволяет выполнять программу по шагам, устанавливать точки останова, просматривать переменные и т. п.

После разработки любого приложения необходимо выполнить его компиляцию, компоновку и тестирование. Эти операции выполняются автоматически каждый раз при выполнении команды **Run**из главного меню среды разработки Turbo C++ Builder . Однако данные процедуры можно выполнять по отдельности с целью сокращения времени на отладку всего проекта в целом. В нашем очередном примере мы будем пользоваться различными командами трансляции (компиляции и компоновки) проекта. Рассмотрим, в чем состоит отличие между этими командами. Например, компиляцию отдельного модуля программы можно выполнить с помощью команды главного меню Project **Compile.** При этом не будет затрачено время на компиляцию и компоновку всего проекта в целом. Команда главного меню **Project Make** позволяет выполнить компиляцию только тех модулей, которые были отредактированы, после чего выполняет компоновку проекта с созданием исполняемого модуля с расширением ехе. И только команда **Project Build** компилирует все модули независимо от того, редактировались они или нет. А затем компонует проект и готовит исполняемый файл с расширением ехе. Данная команда необходима для переработки всего проекта после изменения настроек компилятора или среды разработки в целом. Команда**Project Run** в отличие от Project Build автоматически запускает исполняемый файл.

 На этапе компиляции происходит автоматический поиск синтаксических ошибок, неправильного объявления или использования переменных и т. п. В инспекторе кодов будет выведено соответствующее сообщение об ошибке или предупреждение. Предупреждение в отличие от ошибки не блокирует выполнение программы, но не гарантирует корректность ее работы.

При компоновке программы также выявляются некоторые виды ошибок, связанные с отсутствием объявления переменных, функций, библиотек и т. п., и выполняется объединение всех объектных модулей и библиотек при необходимости в один исполняемый файл с расширением ехе.

# Инструменты отладчика

Основными инструментами отладки любой программы являются команды выполнения программы по шагам и установки точек останова. Команды пошагового выполнения программы вызываются из главного меню в группе Run или с помощью горячих клавиш. Список этих команд, их соответствие горячим клавишам и описание назначения приведен в **Таблице 1.**

Таблица 2. Список команд пошагового выполнения программы.

|  |  |  |
| --- | --- | --- |
| **Команда** | **Горячие клавиши** | **Назначение** |
| **Step Over** | <F8> | Пошаговое выполнение программы без захода в вызываемые функции и подпрограммы |
| **Trace Into** | <F7> | Пошаговое выполнение программы с заходом в вызываемые функции и подпрограммы |
| **Trace to Next Source Line** | <Shift>+<F7> | Перейти  к следующей исполняемой строке программы |
| **Run to Cursor** | <F4> | Выполнить    команды    до    места расположения курсора |
| **Run Until Return** | <Shift>+<F8> | Выполнить команды до выхода из выполняемой функции и остановиться на операторе, следующем за этой функцией |

Команды установки точек останова и некоторые другие вызываются из контекстного меню инспектора кода через раскрывающийся список команды **Run** (Рисунок 5.), т.е. необходимо выполнить команду: Run -> Add Breakpoint -> Source BreakPoint

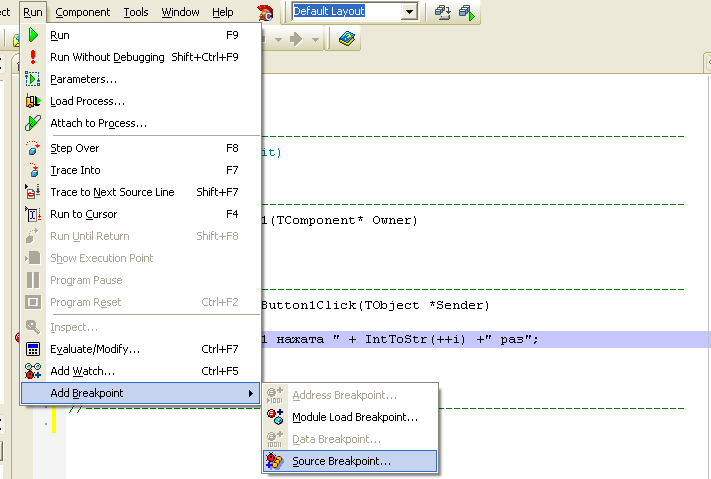


Рисунок 1. Установка точки останова

В появившемся окне нажать ОК.

**Используемые источники**

1. Никита Культин. Основы Программирования в Turbo C++ (+ дистрибутив на CD).-СПб.:БХВ-Петербург,2013.-456 с.

2. <http://radio-hobby.org/modules/instruction/page.php?id=483>

**Контрольные вопросы**

1. Какие ошибки могут возникать при отладке программы?
2. Что выполняет команда Project **Compile**?
3. Что выполняет команда **Project Make** ?
4. Что выполняет команда **Project Build** ?
5. Чем отличается команда **Project Run**  от Project Build ?
6. Какие ошибки обнаруживаются при компиляции?
7. Какие ошибки обнаруживаются при компоновке?

**Порядок выполнения работы**

1. Внимательно прочитать тему занятия и цель, записать в тетрадь.
2. Ознакомиться с теоретическим введением.
3. Выполнить Задание 1, 2 экспериментальной части, показать преподавателю.
4. Выполнить Задание 3. Записать типы ошибок и причины их возникновения.

**Оформление результата**

Записать тему, цель занятия, приложить листинг исходной программы и результат работы. Ответить на контрольные вопросы.

**Экспериментальная часть**

**Задание 1. Создайте небольшое приложение на форме Form1 с двумя кнопками  Button1, Button2 и одним элементом надписи Label1. Это приложение должно сообщать о количестве нажатий на первую кнопку. При нажатии на вторую кнопку приложение** **должно закрываться**.

Разместите эти элементы на форме **Form1**и измените размер формы в соответствии с  **рисунком ( Рисунок 2)**.

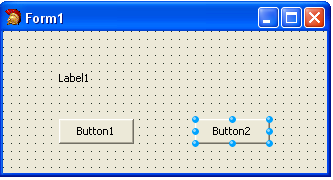


Рисунок 2.

Измените свойство Caption кнопок **Button1, Button2 соответственно на Кнопка1 и Выход .**

Щелкните дважды левой кнопкой мыши по созданной кнопке **Выход** и в отрывшемся окне инспектора кодов впишите между фигурными скобками заготовки обработчика события, команду закрытия приложения Form1->Close();. Вернитесь к окну формы и щелкните дважды левой кнопкой мыши по кнопке с названием Кнопка 1. В отрывшемся окне инспектора кодов впишите между фигурными скобками следующую строку:

    Label1->Caption="Kнonкa 1 нажата " + IntToStr(++i) +" раз";

 В этой строке выполняется присвоение (знак "равно") свойству Caption, элемента надписи Label1, текстовой строчки, состоящей из трех слагаемых частей. Поскольку свойство Caption элемента надписи Label1 предназначено для отображения текста, мы должны присваивать этому свойству только текстовые (строковые) значения. В языке C++ такие строковые значения заключаются в кавычки. Сохраните проект под именем butct.bdsproj, а программный модуль под именем Ubutct.cpp. Не допускается использование в этих именах русских букв. И не изменяйте расширения файлов.

Теперь попробуем скомпилировать, скомпоновать и выполнить данное приложение. Выполните команду **Compile**из группы **Project**главного меню или нажмите комбинацию клавиш для компиляции программы <Alt>+<F9>. Как видим, в программе есть одна ошибка. Нажмите кнопку **ОК**— и перед вами окажется окно инспектора кода с выделенной строкой, имеющей ошибку, и сообщением о типе ошибки (Рисунок 3.).

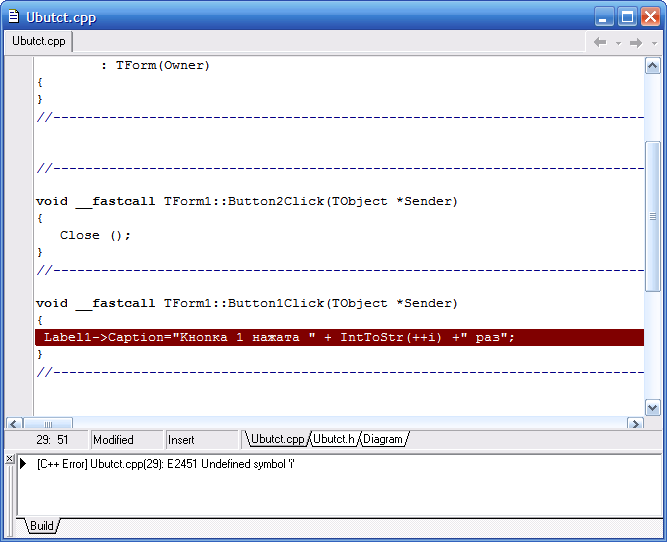


Рисунок 3. Окно инспектора кода.

Строка [C++ Error] Ubutct.pp(29): E2451 Undefined symbol 'i' сообщает о том, что компилятор C++ обнаружил ошибку в 29-й строке модуля Ubutct.cpp. Код ошибки Е2451 означает, что обнаружен необъявленный символ. В нашем случае это переменная с именем **i.** Действительно, мы не объявили ее в программе. Переменная **i** является целой, (тип  **int,** от integer, целый)то есть принимающей только целочисленные значения. Можно сделать такое объявление непосредственно в тексте программы обработчика событий, но тогда эта переменная будет недоступна в других функциях обработчика. Поэтому объявите ее в блоке **public**файла описания заголовков Ubutct.h (Рисунок 4).

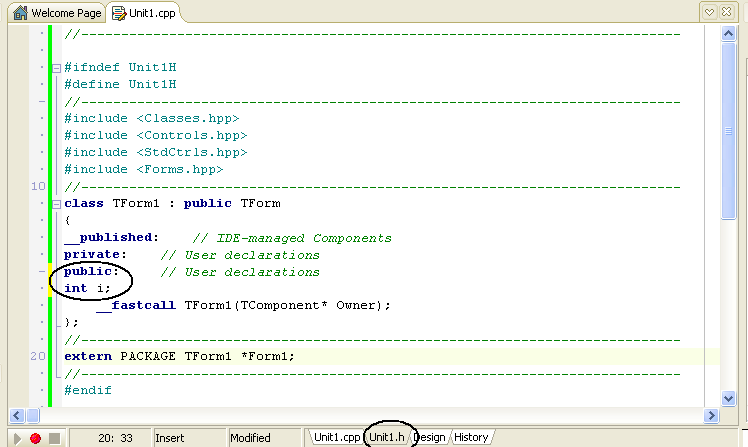


Рисунок 4.

Для этого необходимо щелкнуть левой кнопкой мыши по вкладке **Ubutct.h** инспектора кодов и вписать строку объявления переменной сразу же после строчки public.

**Задание 2. Примените команды выполнения программы по шагам и установки точек** **останова на практике, используя программу, созданную при выполнении Задания 1.**

Поместите курсор на строку:

Labell->Caption="KHonKa 1 нажата "  +  IntToStr(++i)   +" раз";  
и нажмите клавишу <F5>. При этом строка окрасится в красный цвет, что свидетельствует о том, что на ней установили точку останова. Теперь при выполнении программы произойдет автоматический останов на этой строке. Проверьте это, выполнив команду **Run**с помощью клавиши <F9>. Программа запустится и на экране появится ее форма. Но стоит щелкнуть левой кнопкой мыши по кнопке с названием **Кнопка 1**, программа остановится и откроется окно (Рисунок 5) Инспектора кода с зеленой стрелкой напротив строки, на которой установлена точка останова.

Теперь, нажимая клавишу <F8>, можно продолжить выполнение программы по шагам. Для прерывания работы программы необходимо нажать комбинацию клавиш <Ctrl>+<F2>. Можно продолжить выполнение программы с помощью клавиши <F9>, а затем щелкнуть кнопку **Выход**в окне программы для ее завершения. Попробуйте на практике выполнить и другие команды отладчика, для того чтобы понять их назначение.

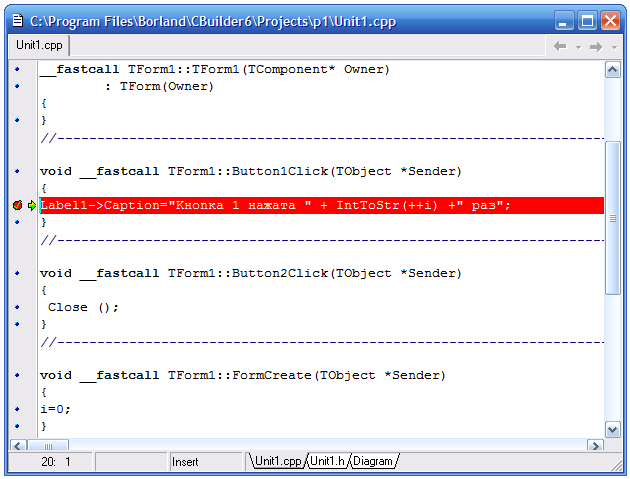


Рисунок 5. Окно инспектора кода

**Задание 3. Примените навыки справления ошибок для отладки приложения.**

Создайте небольшое приложение на форме **Form1** с кнопкой  **Button1**. Это приложение должно сообщать о количестве нажатий на кнопку. Щелкните дважды левой кнопкой мыши по кнопке. В отрывшемся окне инспектора кодов впишите следующий код:

   int i=0;

Label->Caption="Kнonкa 1 нажата " + IntToStr(++i) +" раз"

Теперь попробуйте откомпилировать, скомпоновать и выполнить данное приложение.

Необходимо исправить ошибки, которые могут возникнуть в процессе компиляции (Рисунок 6):

1. **E2451 Undenfined symbol “Label”;**
2. **E2121 Function call missing;**
3. **E2379 Statement missing;**

а также

**Логическую ошибку (**количество нажатий на кнопку вычисляется неверно), которая становиться очевидной после исправления ошибок компиляции.

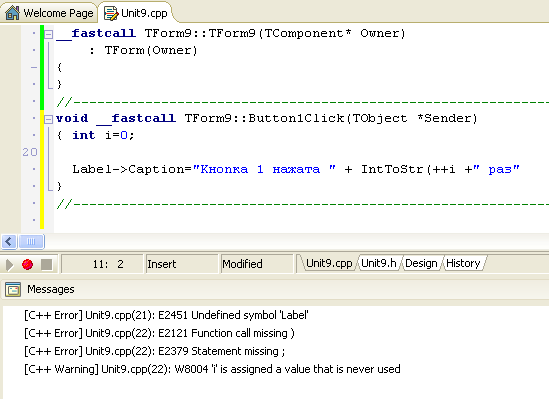
[Введите цитату из документа или краткое описание интересного события. Надпись можно поместить в любое место документа. Для изменения форматирования надписи, содержащей броские цитаты, используйте вкладку "Работа с надписями".]

Рисунок 6. Ошибки компиляции.

.