**Урок 1**

**Начало работы с визуальными компонентами в Microsoft Visual Studio 2010.**

**Программа 1 – «Форма и компоненты Label, Textbox, Button»**

Загружаем среду Microsoft Visual Studio 2010. Чтобы создать новое приложение с визуальными компонентами, выполняем команду главного меню File\New\Project. Открывается диалоговое окно New Project для выбора типа приложения. В этом окне последовательно выполняем следующие действия:

1) в левом поле в разделе Installed Templates выбираем шаблон «Visual C++» -> «CLR»;

2) в среднем поле выбираем шаблон «Windows Forms Application»;

3) в поле Name вводим имя проекта, например, «prog1»;

4) с помощью кнопки Browse находим папку, в которой планируем размещать проект;

5) нажимаем кнопку OK.

В результате на рабочем столе в окне проектирования появится объект Form1, который называется формой (рисунок 1).

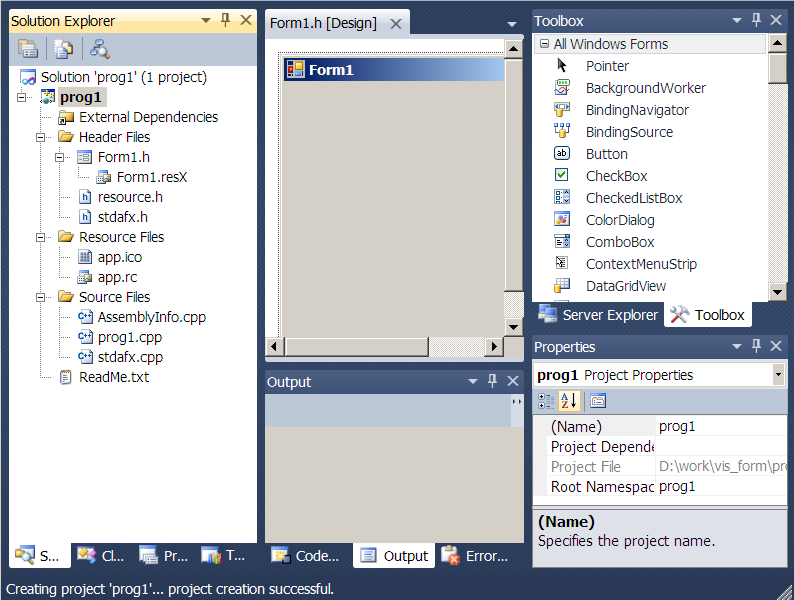


Рисунок 1 – Загрузка формы на рабочий стол

Эта форма представлена в режиме дизайнера форм, на что указывает вкладка «Form1.h [Design]» в верхней части окна проектирования. В режиме дизайнера программист видит текущий внешний вид разрабатываемого приложения и может этот внешний вид редактировать.

Слева от окна проектирования – окно «Solution Explorer». Справа от окна проектирования – окна «Toolbox» и «Properties».

В окне «Solution Explorer» отображается структура проекта. Основная программа находится в файле prog1.cpp. Чтобы увидеть содержимое этого файла, надо в структуре проекта сделать двойной клик мышкой по имени файла, в нашем случае по имени prog1.cpp. Но практически на начальном этапе работы делать это не имеет смысла, так как при создании простых проектов программист в основную программу обычно никакие изменения не вносит.

Итак, мы еще не написали ни строчки кода, а система уже автоматически создала работающее приложение. Компилируем, собираем, запускаем – клавиша [F5]. В качестве работающего приложения видим окно, вид которого отображался в режиме дизайнера. Это окно имеет основные характеристики Widows-окон. В верхней строке заголовка отображается название Form1 и располагаются три основные кнопки управления окном «Свернуть», «Изменить размер», «Закрыть». Итак, запустите приложение, потом закройте его, и Вы вернетесь в режим дизайнера.

Как увидеть код (программный модуль - заготовку), который автоматически создала система вместе с появлением формы на экране? Для этого нужно открыть контекстное меню формы (с помощью правой кнопки мыши) и выполнить опцию View Code. В результате в верхней части окна проектирования, где находится форма, и которое уже содержит вкладку Form1.h[Design], появится новая вкладка Form1.h. Когда эта вкладка открыта, на экране отображается соответствующий программный модуль (рисунок 2).

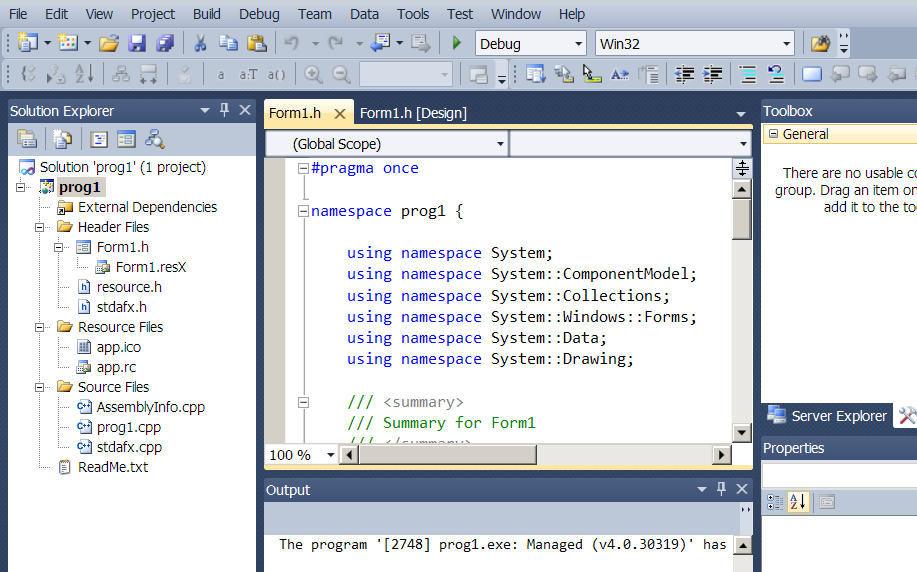


Рисунок 2 – Программный модуль Form1.h

Из содержимого этого программного модуля следует, что экземпляр класса Form, который виден на рабочем столе, действительно является наследником класса Form и получил имя Form1. Форма – это главный объект при создании проекта с визуальными компонентами. В форму, в соответствии с алгоритмом задачи, программист помещает необходимые компоненты из палитры (набора) компонентов среды (они расположены в окне вкладки Toolbox). Программист придает свойствам компонентов необходимые значения (свойства отображаются в специальном окне Properties) и определяет реакции на события компонентов (вкладка Events содержит список возможных событий, которые могут происходить с компонентом). Эти реакции программист задает в программах, которые называются обработчики событий. Все программы-обработчики событий компонентов, расположенных в данной форме, помещаются в тот же программный модуль, который создается вместе с появлением формы на экране (Form1.h, например).

Итак, создав приложение с визуальным интерфейсом, получили на экране пустую форму. Чтобы изменить размеры формы, надо в режиме дизайнера протянуть ее за анкерные точки – небольшие квадратики, выделенные по углам и сторонам формы.

Для каждого компонента есть набор свойств. Активный компонент – это тот, на который в данный момент произведен щелчок мыши. Все свойства активного компонента отображаются в специальном окне под названием Properties (свойства). Это окно может быть видимо или нет. Сделать окно видимым можно, открыв контекстное меню любого компонента и выполнив в нем опцию Properties.

Одно из самых распространенных свойств многих компонентов – свойство Text. В этом свойстве задают название компонента (его заголовок).

Давайте изменим заголовок нашей формы со стандартного заголовка Form1 на любой другой (можно – и на русском языке). Итак, делаем активной нашу форму (делаем на ней щелчок мыши). Нажимаем правую кнопку мыши и открываем контекстное меню. Выбираем опцию Properties. Внизу справа в одноименном окне видим свойства формы. Первое свойство – Text. Правее указано его текущее значение Form1, которое установила система (рисунок 3).

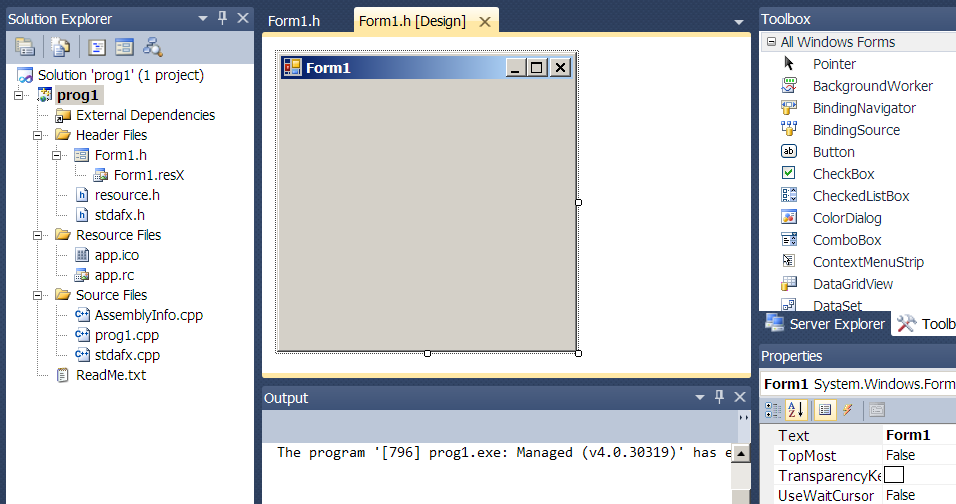


Рисунок 3 – Свойства формы

Подводим курсор мыши и делаем щелчок после слова Form1 в строке со свойством Text, в результате там появится курсор. Удаляем старое имя Form1, вводим новое, например, “Программа 1», нажимаем клавишу [Enter]. Сразу же в дизайнере формы изменится заголовок формы. В файле Form1.h также произойдут соответствующие изменения (рисунок 4).

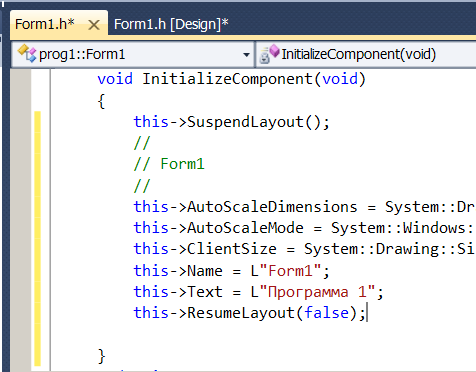


Рисунок 4 – Фрагмент кода с описанием главной формы

Обратите внимание в коде на имя Name и текст Text. Name – это имя компонента в программе, то есть это имя переменной. Естественно, оно может состоять только из латинских букв. Text – это название компонента, которое отображается в дизайнере и в работающем приложении, то есть это название «для пользователя». Оно может быть записано и латинским, и русским шрифтом. Для каждого компонента, кроме главной формы, программист может установить свое имя переменной Name и свой текст-заголовок Text, если он предусмотрен для этого компонента. Для главной формы имя Form1 изменять нельзя.

Изменять свойства компонента можно и через окно Properties, и непосредственно в коде. Проще и безопаснее, конечно, использовать окно Properties.

Теперь приступим к разработке нашего первого приложения. Суть работы этого приложения состоит в следующем: в специальное окно пользователь может вводить любой текст, а при щелчке мышкой по специальной кнопке текст из окна удаляется. Для реализации будем использовать три наиболее распространенных компонента:

1) Label (метка) – для надписи над окном ввода текста;

2) Textbox (окно для ввода-вывода строчных данных) – для окна, в который вводится текст;

3) Button (кнопка) – для кнопки, по которой можно щелкать мышкой.

В режиме дизайнера смотрим на вкладку Toolbox. Находим в ней необходимые нам компоненты и перетаскиваем их в нашу форму. Для переноса компонента в форму существует два способа:

1) дважды щелкнуть на нужном компоненте на вкладке; в результате компонент отобразится в форме и будет активным, далее перетаскиваем его мышкой в нужное место на форме;

2) щелкнуть на нужном компоненте на вкладке один раз, перевести указатель мыши в нужное место окна дизайнера форм и снова щелкнуть мышью (то есть выделить компонент указателем мыши и «тянуть» в форму).

Итак, перетаскиваем в форму компоненты Label, Textbox, Button. Получим примерно следующий вид формы – рисунок 5.

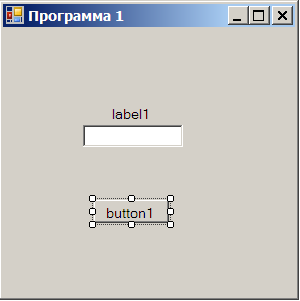


Рисунок 5 – Форма с 3 компонентами label1, textbox1, button1 на промежуточном этапе 1

Один из компонентов формы активен. На нашем рисунке это компонент button1. Активный компонент можно перемещать по форме и изменять его размер. Чтобы сделать активным другой компонент, надо щелкнуть по нему мышкой.

Запустите полученное приложение (клавиша [F5]). Вы уже сможете вводить текст в окно и щелкать мышкой по кнопке (рисунок 6). Естественно, при этом щелчке текст из окна удаляться пока не будет, так как для этого надо написать специальную программу – обработчик события «Щелчок мышью на кнопке».

Обязательно посмотрите, какой код на этом этапе работы автоматически создала система в Form.1. Фрагмент кода с описанием компонентов приведен на рисунке 7. Обратите внимание, как удобно работать с хорошо прокомментированным кодом. Давайте, например, посмотрим на описание одного из компонентов - метки label1. Система установила имя соответствующей переменной в программе (Name) как label1 и установила заголовок компонента (Тext) также как label1. Для последующих меток система будет устанавливать имена label2, label3 и т.д. Естественно, что однотипные стандартные заголовки компонентов (Text) надо изменить на такие, которые будут отражать суть использования компонентов для нашей задачи. Что касается имен переменных, казалось бы, можно оставить стандартные имена (label1, label2 и т.д.), так как пользователь их не видит. Но для программиста это неудобно. Представьте, что у Вас порядка 30 компонентов - меток и Вы в программе для работы с ними используете однотипные имена переменных, лишенные смысловой нагрузки. Вы запутаетесь в коде. Поэтому настоятельно рекомендуется устанавливать имена переменных, адекватно характеризующие сущность каждой переменной. И, конечно, рекомендуется, чтобы имя переменной и соответствующий текст были максимально похожи.

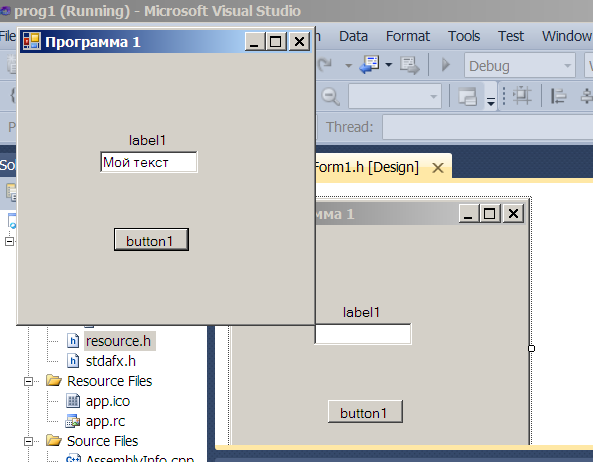


Рисунок 6 – Вид работающего приложения на промежуточном этапе 1

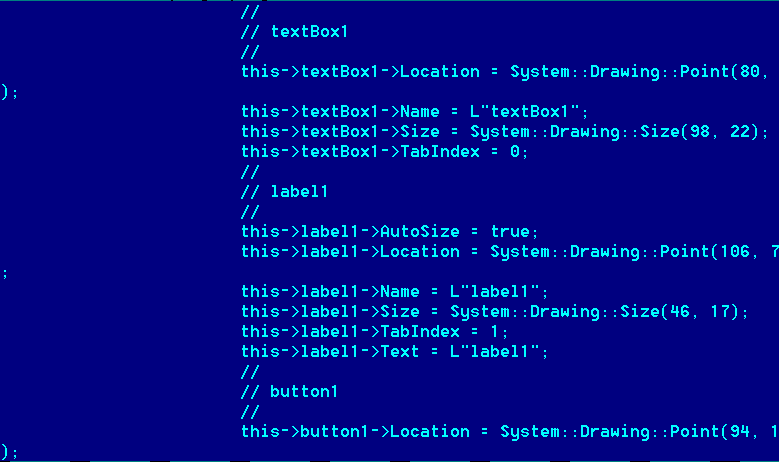


Рисунок 7 – Фрагмент описания компонентов в Form1.h на промежуточном этапе 1

Итак, переходим к работе со свойствами компонентов в дизайнере форм.

Начинаем работать с меткой label1. Делаем ее активной, выделяя мышкой. Правой кнопкой мышки открываем контекстное меню и открываем свойства метки (выбираем опцию Properties). Значение свойства Text изменяем на, например, «Введите текст». Как только Вы нажмете клавишу [Enter], на форме этот новый текст установится. Если необходимо, переместите метку на экране в нужное Вам место. Значение свойства Name (оно указано вверху списка свойств) заменяем, например, на «label\_EnterText».

Начинаем работать с окном ввода-вывода textbox. Делаем это окно активным, выделяя мышкой. Растягиваем его в длину, как считаем нужным, протягивая за анкерные точки. Нажимаем, а потом отпускаем (не щелкаем) кнопку мыши на специальной кнопке с изображением маленькой стрелочки над правым верхним углом окна. В результате открывается вспомогательное меню на самом компоненте. Включаем свойство многострочности ввода-вывода, поставив галочку в Mulline. После этого окно можно растянуть и в высоту. Перемещаем окно на экране в нужное Вам место.

Правой кнопкой мышки открываем контекстное меню и открываем свойства окна ввода-вывода (выбираем опцию Properties). Значение свойства Name (оно указано вверху списка свойств) заменяем, например, на «textBox\_EnterText».

Начинаем работать с кнопкой butten. Делаем ее активной, выделяя мышкой. Правой кнопкой мышки открываем контекстное меню и открываем свойства кнопки (выбираем опцию Properties). Значение свойства Text изменяем на, например, «Удалить текст». Как только Вы нажмете клавишу [Enter], на форме этот новый текст установится в кнопке. Если необходимо, на экране растяните кнопку и переместите ее в нужное Вам место. Значение свойства Name (оно указано вверху списка свойств) заменяем, например, на «button\_DeleteText».

Полученный в итоге внешний вид формы представлен на рисунке 8. Запустите приложение для текущего контроля (рисунок 9). Естественно, при щелчке по кнопке текст из окна удаляться пока не будет, так как для этого надо написать специальную программу – обработчик события «Щелчок мышью на кнопке».

Обязательно проанализируйте полученный код в Form1.h (рисунок 10) и сравните его с предыдущей версией (рисунок 7). Посмотрите, например, на описание метки и там, и там.

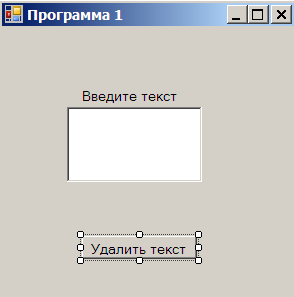


Рисунок 8 – Итоговый вид формы с 3 компонентами

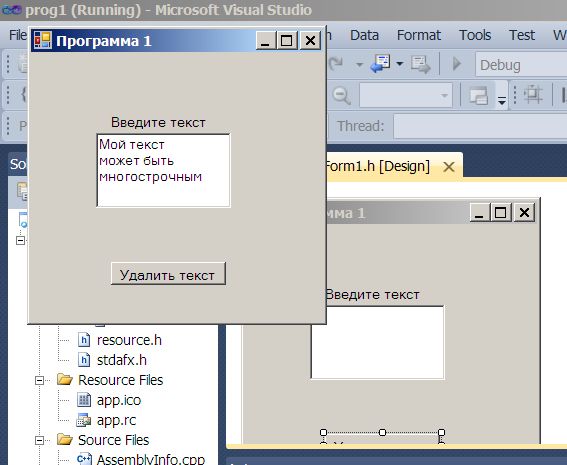


Рисунок 9 – Вид работающего приложения на промежуточном этапе 2

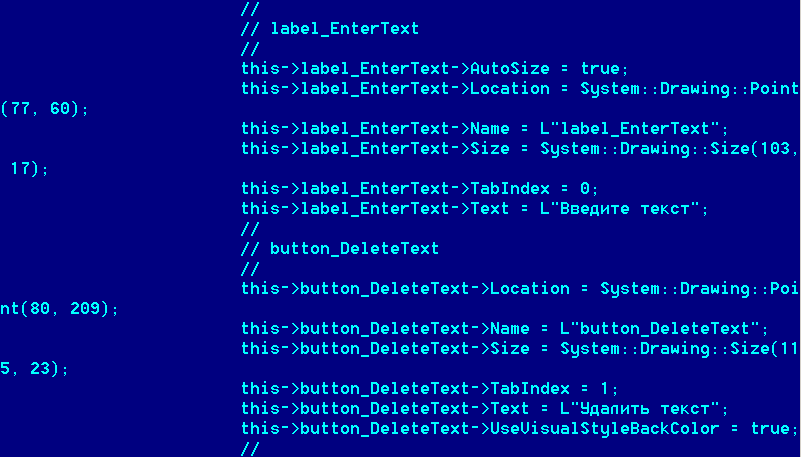


Рисунок 10 – Фрагмент описания компонентов в Form1.h на промежуточном этапе 2

Теперь приступим к написанию программы – обработчика события. Для создания такой функции надо провести следующие действия. Вначале устанавливаем как активный тот компонент, для которого планируем написать программу – обработчик события. Открываем контекстное меню для этого компонента, выбираем опцию Properties. В окне Properties имеется 5 вкладок. Центральная вкладка показывает перечень свойств компонента. Следующая, четвертая, вкладка (значок на ней похож на молнию) показывает перечень событий, связанных с компонентом. Переключаемся на эту вкладку и видим список возможных событий, которые могут происходить с этим компонентом. Выбираем нужное нам событие и в пустом поле правее имени этого события дважды щелкаем мышью. В результате система создаст в модуле формы, в которую помещен компонент (в нашем случае в файле Form1.h), программу-обработчик этого события. Это будет функция с заголовочной частью, но с пустым телом, то есть мы получим заготовку программы. В это пустое тело программист должен вписать свои команды, которые будут определять реакцию компонента на данное событие.

Для каждого компонента существует самое распространенное событие. Например, для компонента «кнопка» (butten) - это событие Click, которое возникает, когда на кнопке щелкают мышью. Создать заготовку программы-обработчика самого распространенного события можно проще и быстрее, чем то, как описано выше. Для этого в дизайнере формы надо сделать двойной щелчок мышью по тому компоненту, для которого планируете написать обработчик такого события.

Переходим к нашему приложению. Ниже в «разговорном» стиле приводится описание действий, очень удобных для программиста.

В программе предполагается одно событие: «Щелчок мышью на кнопке». Это событие - самое распространенное для кнопки. Поэтому в дизайнере форм делаем двойной щелчок мышью по нашей кнопке. Система сразу же открывает код Form1.h и устанавливает курсор в пустое тело созданного обработчика события. Вид этого пустого обработчика события будет таким:

private: System::Void button\_DeleteText\_Click(System::Object^ sender, System::EventArgs^ e) {

}

В фигурных скобках программисту надо написать тело функции. Что должно произойти в программе, если пользователь щелкнет мышью на кнопке? Текст из окна textbox\_EnterText должен исчезнуть, то есть программисту надо работать с компонентом textbox\_EnterText. Рекомендуется, по возможности, минимально писать вручную, а максимально использовать тест, созданный системой. Поэтому переходим в коде выше на описание компонента textbox\_EnterText. Копируем там строчку

this->textBox\_EnterText->Name = L"textBox\_EnterText";

и вставляем ее в свою программу-обработчик события. Получим следующую программу:

private: System::Void button\_DeleteText\_Click(System::Object^ sender, System::EventArgs^ e) {

this->textBox\_EnterText->Name = L"textBox\_EnterText";

}

В скопированной строке устанавливается имя Name компонента, а нам надо работать с текстом, отображаемым в этом компоненте, поэтому заменяем слово Name на слово Text. Нам нужно, чтобы в компоненте отображался «пустой» текст, поэтому убираем всю строку в двойных кавычках. Получаем в итоге:

private: System::Void button\_DeleteText\_Click(System::Object^ sender, System::EventArgs^ e) {

this->textBox\_EnterText->Text = L"";

}

Запускаем приложение. Теперь можно в окно вводить текст, а щелкнув по кнопке – очищать окно, удаляя текст (рисунок 11). Приложение готово!

|  |  |
| --- | --- |
| а) | б) |

Рисунок 11 – Приложение работает: а) ввели текст, б) щелкнули мышкой по кнопке