**КУРСОВОЙ ПРОЕКТ**

**ПО ДИСЦИПЛИНЕ**

**«Технология программирования»**

**НА ТЕМУ:**

**«Анализ треугольника»**

Ставрополь 2011

**Введение**

В связи с тем, что сегодня уровень сложности программного обеспечения очень высок, разработка приложений Windows с использованием только одного какого-либо языка (например, языка С) значительно затрудняется. Программист должен затратить массу времени на решение стандартных задач по созданию оконного интерфейса. Реализация технологии связывания и встраивания объектов L потребует от программиста еще более сложной работы. Чтобы облегчить работу программиста, все современные компиляторы языка С++ содержат специальные библиотеки классов. Такие библиотеки включают в себя практически весь программный интерфейс Windows и позволяют пользоваться при программировании средствами более высокого уровня, чем обычные вызовы функций. За счет этого значительно упрощается разработка приложений, имеющих сложный интерфейс пользователя, облегчается поддержка технологии L и взаимодействие с базами данных. Кроме того, стало проще производить отладку программ, написанных на С++. Вместо одной огромной монотонной программы мы имеем дело с набором аккуратных и удобных в обращении частей. Современные интегрированные средства разработки приложений Windows позволяют автоматизировать процесс создания приложения. Для этого используются генераторы приложений. Программист отвечает на вопросы генератора приложений и определяет свойства приложения – поддерживает ли оно многооконный режим, технологию L, справочную систему. Генератор приложений создаст приложение, отвечающее требованиям, и предоставит исходные тексты. Пользуясь им как шаблоном, программист сможет быстро разрабатывать свои приложения. Подобные средства автоматизированного создания приложений фирма Mirsft включила в продукт Visul ++, разработав библиотеку MF (Mirsft Fundtin lsss). MF – замечательный пакет, состоящий из заранее написанного и готового к работе кода. Заполнив несколько диалоговых панелей, можно указать характеристики приложения его тексты, снабженные обширными комментариями. MF Wizrd позволяет создавать однооконные и многооконные приложения, а также приложения, не имеющие главного окна, – вместо него используется диалоговая панель. Прикладную часть приложения программисту придется создавать самостоятельно. Исходный текст приложения станет только основой, к которой нужно подключить основное.

**1. Теоретические основы курсового проекта**

**1.1 Теоретические основы разработки Windows-приложений с использованием библиотеки MF**

Библиотека **MF (Mirsft Fundtin lsss)** – это набор классов **++** и глобальных функций, созданных для быстрой разработки **Windows**-приложений. Эта библиотека упрощает программирование под **Windows** и сокращает время разработки. Код, созданный с ее использованием, легко переносить на другие платформы. Библиотека **MF** упрощает решение таких сложных задач, как создание приложений, ориентированных на технологию **tiv** или на Интернет. С ее помощью легко программировать элементы интерфейса – предварительный просмотр документов перед печатью, страницы свойств, контекстные меню и настраиваемые панели инструментов с всплывающими подсказками.

Общий для всех 32-разрядных **Windows**-платформ ***интерфейс прикладного программирования*** называется **Win32**. **Win32 I** – это набор функций, структур, сообщений, макросов и интерфейсов, с помощью которых можно создавать программы для **Win32**. Так как **Win32 I** – это набор **С**-функций, их можно использовать в своей программе, просто включив в код соответствующий заголовочный файл. До появления **MF** **Windows**-программы применяли только средства **Win32 I**, так что разработка занимала много времени. Опытные программисты старались ускорить ее, создавая библиотеки с кодом основных, часто используемых функций. С появлением **MF** такие библиотеки стали доступны всем, кто пишет программы на **++**.

Классы **MF**, инкапсулирующие основные функции **Win32 I**, упрощают и ускоряют разработку **Windows**-приложений. Эти классы реализуют сложные функции, скрывающиеся за понятным и компактным интерфейсом.

Например, концепция окна представлена классом **MF Wnd**. Другими словами, класс **Wnd** инкапсулирует описатель **HWND**, определенный в **Win32** тип данных, содержащий сведения об окне. Это значит, что в классе **Wnd** есть переменная-член типа **HWND**, а его функции-члены замещают те функции **Win32 I**, которым **HWND** передается в качестве параметра. Например, такая функция **Win32 I**:

BL ShwWindw (HWND hWnd, int nmdShw);

заменена в **MF** на

BL Wnd: ShwWindw (int nmdShw);

Функции-члены классов библиотеки **MF** и заменяемые ими функции **Win32 I** обычно называются одинаково.

**MF** скрывает некоторые низкоуровневые элементы, но не запрещает их использовать там, где это необходимо. Инкапсуляция обеспечивает также защиту значений указателей, например описателей файлов и окон, от повреждения или от неправильного их использования.

Библиотека **MF** подходит для создания любых приложений, кроме самых простых. Однако даже для таких программ весьма полезны некоторые классы этой библиотеки, например строковые или классы наборов.

В **MF** представлены не все функции **Win32 I**, поэтому иногда требуется их прямой вызов. Это необходимо только при низкоуровневом доступе к системе. Например, если Вы разрабатываете утилиту, изменяющую учетную запись пользователя в сети, Вам придется применять сетевые функции **Win32 I**, так как в **MF** нет классов, обеспечивающих доступ к таким функциям.

Библиотека MF содержит большую иерархию классов, написанных на С++. В ее вершине находится класс bjt, который содержит различные функции, используемые во время выполнения программы и предназначенные, в частности, для предоставления информации о текущем типе во время выполнения, для диагностики, и для сериализации.

Если указатель или ссылка ссылается на объект, производный от класса bjt, то в этом случае предусмотрен механизм определения реального типа объекта с помощью макроса RUNTIM\_LSS(). Хотя в С++ имеется механизм RTTI, механизм, реализованный в MF, намного более эффективен по производительности. Каждый класс, производный от bjt, может по запросу проверить свое внутреннее состояние и выдать диагностическую информацию. Это интенсивно используется в MF при отладке.

Сериализация – это механизм, позволяющий преобразовать текущее состояние объекта в последовательный поток байт, который обычно затем записывается на диск, и восстановить состояние объекта из последовательного потока, обычно при чтении с диска. Это позволяет сохранять текущее состояние приложения на диске, и восстанавливать его при последующем запуске. Некоторые классы порождаются непосредственно от bjt. Наиболее широко используемыми среди них являются mdTrgt, Fil, D, GDIbjt и Mnu. Класс mdTrgt предназначен для обработки сообщений. Класс Fil предназначен для работы с файлами. Класс D обеспечивает поддержку контекстов устройств. О контекстах устройств мы будем говорить несколько позднее. В этот класс включены практически все функции графики GDI. GDIbjt является базовым классом для различных DGI-объектов, таких как перья, кисти, шрифты и другие. Класс Mnu предназначен для манипуляций с меню. От класса mdTrgt порождается очень важный класс Wnd. Он является базовым для создания всех типов окон, включая масштабируемые («обычные») и диалоговые, а также различные элементы управления. Наиболее широко используемым производным классом является FrmWnd. Как Вы увидите в дальнейшем, в большинстве программ главное окно создается с помощью именно этого класса. От класса mdTrgt, через класс WinThrd, порождается, наверное, единственный из наиболее важных классов, обращение к которому в MF-программах происходит напрямую: Win. Это один из фундаментальных классов, поскольку предназначен для создания самого приложения.

**1.1.1 I функции**

Большинство функций, вызываемых в MF-программе, являются членами одного из классов, определенных в библиотеке. Большинство функций I доступны через функции-члены MF. Тем не менее, всегда можно обращаться к функциям I напрямую. Иногда это бывает необходимым, но все же в большинстве случаев удобнее использовать функции-члены MF.

**1.1.2 Глобальные функции F**

В библиотеке есть ряд глобальных функций. Все они начинаются с префикса f. Когда MF только разрабатывалась, то проект назывался F, litin Frmwrk. После ряда существенных изменений F была переработана в MF, но прежнее название сохранилось во многих идентификаторах библиотеки и в названиях файлов. Например, очень часто используется функция fMssgB(), отображающая заранее определенное окно сообщения. Но есть и член-функция MssgB(). Таким образом, часто глобальные функции перекрываются функциями-членами.

Все MF-программы включают заголовочный файл FWIN.H. В нем, а также в различных вспомогательных файлах, содержатся описания классов, структур, переменных и других объектов MF. Он автоматически подключает большинство заголовочных файлов, относящихся к MF, в том числе и WINDOWS.H, в котором определены все функции Windows I и другие объекты, которые используются при традиционном программировании на С и «чистом» I.

Windows взаимодействует с программой, посылая ей сообщения.

программа треугольник площадь прямоугольный

**1.1.3 Цикл сообщений**

Все приложения Windows должны организовать так называемый *цикл* *сообщений* (обычно внутри функции WinMin()). В этом цикле каждое необработанное сообщение должно быть извлечено из очереди сообщений данного приложения и передано назад в Windows, которая затем вызывает функцию окна программы с данным сообщением в качестве аргумента. В традиционных Windows-программах необходимо самостоятельно создавать и активизировать такой цикл. При использовании MF это также выполняется автоматически. Однако важно помнить, что цикл сообщений все же существует. Он является неотъемлемой частью любого приложения Windows.

**1.1.4 Создание приложения с помощью Visul ++**

О принципах устройства приложения рассказывалось выше. Теперь рассмотрим, как оно создается с помощью Visul ++. Сначала разберем одно важное понятие **–** проект. До сих пор приложение рассматривалось, как только как совокупность объектов базовых и производных классов. Но для обеспечения работы приложения требуется нечто большее – наряду с описанием классов необходимо описание ресурсов, связанных с приложением, нужна справочная система и т.п. Термин «проект» как раз и используется, когда имеется в виду такой общий взгляд на приложение.

В среде Visul ++ можно строить различные типы проектов. Такие проекты после их создания можно компилировать и запускать на исполнение. Фирма Mirsft разработала специальный инструментарий, облегчающий и ускоряющий создание проектов в среде Visul ++. Например, мастер MF Wizrd () позволяет создать проект, Windows-приложения которое имеет однодокументный, многодокументный или диалоговый интерфейс и использует библиотеку MF.

**1.1.5 Инструментальное средство – lssWizrd (мастер классов)**

Создаваемый остов приложения составлен так, что в дальнейшей работе с проектом можно использовать другое инструментальное средство – lssWizrd (мастер классов), предназначенное для создания остовов новых производных классов. Еще одно основное назначение lssWizrd в том, что он создает остовы для переопределяемых методов. Он позволяет показать все сообщения, приходящие классу, и создать остов обработчика любого из этих сообщений. Это только две основные функции lssWizrd. Он не всесилен, но его возможности довольно велики.

Средство lssWizrd предоставляет широкий спектр услуг. Он позволяет не только добавлять к существующему классу новые методы и данные.

Создание нового класса.

При помощи lssWizrd можно добавить новый класс, созданный на основе базовых классов. В качестве базового класса можно использовать классы, наследованные от класса mdTrgt или класса Rrdst. Для наследования классов от других базовых классов использовать средства lssWizrd нельзя. Такие классы надо создавать вручную, непосредственно в текстовом редакторе.

Полученная заготовка класса полностью работоспособна. Ее можно дополнить по своему усмотрению новыми методами и данными. Эту работу можно выполнить вручную, но гораздо лучше и проще воспользоваться услугами lssWizrd. За счет использования lssWizrd процедура создания собственного класса значительно ускоряется и уменьшается вероятность совершить ошибку во время объявления методов.

Включение в класс новых методов.

Очень удобно использовать lssWizrd для включения в состав класса новых методов. Можно добавлять к классу методы, служащие для обработки сообщений Windows и команд от объектов, а также методы, переопределяющие виртуальные методы базовых классов.

lssWizrd не только позволяет добавить в класс новые методы, но и удалить их. lssWizrd самостоятельно удалит объявление метода из класса.

Включение в класс новых элементов данных.

lssWizrd позволяет включать в класс не только новые методы, но и элементы данных, связанные с полями диалоговых панелей, форм просмотра и форм для просмотра записей баз данных и полей наборов записей. lssWizrd использует специальные процедуры, чтобы привязать созданные им элементы данных к класса к полям диалоговых панелей. Эти процедуры носят названия «обмен данными диалоговой панели» и «проверка данных диалоговой панели» (Dilg Dt hng nd Dilg Dt Vlidtin – DD/DDV). Чтобы привязать поля из наборов записей к переменным, используется процедура обмена данными с полями записей (Rrd Fild hng – RF).

Процедуры DD/DDV и RF.

Значительно упрощают программисту работу с диалоговыми панелями. Они позволяют связать поля диалоговых панелей и переменные. Когда пользователь редактирует поля диалоговых панелей, процедуры DDV проверяют введенные значения и блокируют ввод запрещенных значений. Затем процедуры DD автоматически копируют содержимое полей диалоговых панелей в привязанные к ним элементы данных класса. И наоборот, когда приложение изменяет элементы данных класса, привязанные к полям диалоговой панели, процедуры DD могут сразу отобразить новые значения полей на экране компьютера. Процесс получения и обработки сообщений может показаться чересчур сложным, но, тем не менее, ему должны следовать все Windows-программы. К счастью, при использовании библиотеки MF большинство частных деталей скрыты от программиста, хотя и продолжают неявно присутствовать в программе. В простейшем случае программа, написанная с помощью MF, содержит два класса, порождаемые от классов иерархии библиотеки: класс, предназначенный для создания приложения, и класс, предназначенный для создания окна. Другими словами, для создания минимальной программы необходимо породить один класс от Win, а другой – от FrmWnd. Эти два класса обязательны для любой программы. Кроме создания вышеупомянутых классов, в программе также должна быть организована обработка всех сообщений, поступающих от Windows. В данном примере программа еще ничего полезного не делает, поэтому отвечать на каждое сообщение не нужно. MF обработает все сообщения, которые нас не интересуют. Тем не менее, в этом примере присутствует карта откликов на сообщения, или просто карта сообщений. Позже мы рассмотрим ее подробнее. Как уже упоминалось, MF – это базовый набор (библиотека) классов, написанных на языке С++ и предназначенных для упрощения и ускорения процесса программирования для Windows. Библиотека содержит многоуровневую иерархию классов, насчитывающую около 200 членов. Они дают возможность создавать Windows-приложения на базе объектно-ориентированного подхода. С точки зрения программиста, MF представляет собой каркас, на основе которого можно писать программы для Windows.

**1.1.6 Преимущество MF библиотек**

Библиотека MF разрабатывалась для упрощения задач, стоящих перед программистом. Как известно, традиционный метод программирования под Windows требует написания достаточно длинных и сложных программ, имеющих ряд специфических особенностей. В частности, для создания только каркаса программы таким методом понадобится около 75 строк кода. По мере же увеличения сложности программы ее код может достигать поистине невероятных размеров. Однако та же самая программа, написанная с использованием MF, будет примерно в три раза меньше, поскольку большинство частных деталей скрыто от программиста.

Одним из основных преимуществ работы с MF является возможность многократного использования одного и того же кода. Так как библиотека содержит много элементов, общих для всех Windows-приложений, нет необходимости каждый раз писать их заново. Вместо этого их можно просто наследовать (говоря языком объектно-ориентированного программирования). Кроме того, интерфейс, обеспечиваемый библиотекой, практически независим от конкретных деталей, его реализующих. Поэтому программы, написанные на основе MF, могут быть легко адаптированы к новым версиям Windows (в отличие от большинства программ, написанных обычными методами).

Еще одним существенным преимуществом MF является упрощение взаимодействия с прикладным программным интерфейсом (I) Windows. Любое приложение взаимодействует с Windows через I, которое содержит несколько сот функций. Внушительный размер I затрудняет попытки понять и изучить его целиком. Зачастую, даже сложно проследить, как отдельные части I связанны друг с другом! Но поскольку библиотека MF объединяет (путем инкапсуляции) функции I в логически организованное множество классов, интерфейсом становится значительно легче управлять. Поскольку MF представляет собой набор классов, написанных на языке С++, поэтому программы, написанные с использованием MF, должны быть в то же время программами на С++. Для этого необходимо владеть соответствующими знаниями. Для начала необходимо уметь создавать собственные классы, понимать принципы наследования и уметь переопределять виртуальные функции. Хотя программы, использующие библиотеку MF, обычно не содержат слишком специфических элементов из арсенала С++, для их написания тем не менее требуются солидные знания в данной области [7].

**2. Описание программного продукта**

**2.1 Общие сведения**

Данная программа написана на языке Visul ++ 6.0. После запуска программа предлагает ввести длины сторон треугольника и рассчитать некоторые его характеристики.

**2.2** **Функциональное назначение**

Данная программа предназначена для анализа треугольника на наличие в нем прямого угла, а также для расчета периметра, площади, радиуса вписанной и описанной окружности.

**2.3 Спецификация программы**

Программа называется Tringl. В программе были разработаны три класса и их методы. Интерфейс и все вычисления были выполнены в одном классе, описывающем главное диалоговое окно (рисунок 2.3.1). Класс называется TringlDlg. Он наследован от класса Dilg.

Рисунок 2.3.1 – Главное окно программы

Ввод длин сторон треугольника осуществляется через элементы dit. С ними связаны атрибуты класса m\_lin\_1, m\_lin\_2, m\_lin\_3. Эти атрибуты имеют тип int. Минимальным значение длины стороны является 1. У пользователя имеется возможность ввести значения вручную или использовать для этого элементы Sin. В программе реализована обработка события UD\_DLTS данных элементов. Функции, реализующие обработку данных событий, называются nDltsSin1, nDltsSin2, nDltsSin3. В данные функции передается указатель NM\_UDWN\* NMUDwn. Через него можно получить значение атрибута iDlt (значение, на которое следует увеличить или уменьшить связанное с элементом число).

Если пользователь попытается ввести число, не соответствующее целому типу, программа выдаст соответствующее сообщение об этом.

После ввода длин сторон пользователю необходимо нажать на кнопку «Рассчитать». С этой кнопкой связана функция nlult().

Сначала функция проверяет, правильно ли задан треугольник. Это происходит следующим образом. По известной формуле Герона рассчитывается площадь треугольника.

,

где ,

длины сторон треугольника

В случае, значение выражения под корнем равно нулю или меньше нуля, считается, что такого треугольника не существует. В этом случае программа выдает соответствующее сообщение (рисунок 2.3.2).

2.3.2. Сообщение о некорректном вводе

Если треугольник задан правильно, программа вычисляет, является ли он прямоугольным. Для этого использована теорема Пифагора:



Программа находит наибольшую сторону (предполагаемую гипотенузу) и проверяет на истинность условие (2). Если оно истинно, то выдается сообщение «Треугольник прямоугольный». В противном случае, «Треугольник не прямоугольный» (рисунок 2.3.3).

Рисунок 2.3.3. Анализ треугольника на наличие прямого угла

Далее функция рассчитывает периметр треугольника (3), радиус описанной окружности (4) и радиус вписанной окружности (5).







После получения результатов, функция преобразует их в текстовый вид. Для этого используется String m\_rsult и функция Frmt. Переменная m\_rsult связана с элементом Stti. Результат вычислений появится на форме.

Рисунок 2.3.4. Результат вычислений

Таблица 2.1 – Спецификация разработанных классов

|  |  |  |  |
| --- | --- | --- | --- |
| Имя класса | Член класса | Способа доступа | Назначение |
| TringlDlg | BL nInitilDilg() | защищенный | Выставляет значения длин сторон |
| TringlDlg | vid nlult() | защищенный | Производит все вычисления |

**2.4 Требования по установке**

Для того чтобы создать приложение, просмотреть текст программы или запустить её на выполнение – необходимо, чтобы на компьютере был установлена среда Mirsft Visul ++. Чтобы просмотреть текст программы Funtin или запустить её на выполнение – надо скопировать с носителя папку с проектом на жёсткий диск компьютера. Если надо просмотреть текст программы, шаблоны диалоговых окон и др., то следует открыть файл Funtin.dsw, являющийся файлом проекта.

**2.5 Требования к техническому обеспечению**

Для корректной работы программы необходимы IBM совместимый компьютер, операционная система семейства Windows версии 95/98/NT/, оперативная память 128 Mb и выше, свободное место на диске 3153 Кбайт под размещение приложения.

**2.6 Вызов программы. Входные и выходные данные**

Программа представляет собой единственный файл-приложение Tringl., запускаемое в любой операционной системе семейства Windows.

Входными данными является введенные пользователем длины сторон треугольника. Результаты своих вычислений программа выводит на экран.

**2.7 Тестирование программы**

Программа тестировалась стохастическим методом. Тестировался запуск программы, задавались неправильно заданные треугольники, вводились числа большого размера, ответа и подсчет результата. Выявленные незначительные ошибки функционирования были устранены.

В общем, программа работает без сбоев и может широко использоваться при обучении математике школьников и студентов.

**Заключение**

В ходе выполнения курсового проекта было создано Windows-приложение, которое вычисляет различные характеристики треугольника, заданного пользователем. Данная программа была написана на Mirsft Visul ++ 6.0 с использованием библиотеки MF.

В ходе разработки курсового проекта была освоена работа с Mirsft Visul ++. В результате чего была написана программа. Была освоена работа с различными элементами интерфейса, работа с мастерами Visul ++ 6.0.

Такие простые приложения и ему подобные, могут широко внедряться в различных сферах человеческой деятельности. Можно написать подобные программы на языке программирования Mirsft Visul ++ с использованием библиотеки MF для создания других приложений.

**Список литературы**

1. «Основы программирования на Visul ++», методические указания по курсу «Технология программирования» для специальности 220200,071900
2. Кейт Грегорий «Использование Visul ++ 6.0», М, СПб, К.: издательский дом «Вильямс», 2000.
3. Черносвитов А., «Visul ++ и MF. Курс MSD для профессионалов.» – СПб: Издательство «Питер», 2000.
4. Паппас К., Мюррей У. «Полное руководство по Visul ++». Минск: ООО «Попурри», 1999.
5. Марк Луис. «Visul ++ 6». – М.: Лаборатория базовых знаний, 1999.
6. Гилберт Стивен, Маккарти Билл. «Программирование на Visul ++ 6. Этюды профессионалов» – К.: Издательство «ДиаСофт», 1999.
7. Баженова И.Ю. «Visul ++6.0 (VISUL STUDI 1998). Уроки программирования», М.: Диалог – МИФИ, 1999
8. А.В. Фролов, Г.В. Фролов «Mirsft Visul ++ и MF. Программирование для Windows 95 и Windows NT» в 2-х ч. (часть1) (библиотека системного программиста; т. 24) М.: Диалог – МИФИ, 1997
9. А.В. Фролов, Г.В. Фролов «Mirsft Visul ++ и MF. Программирование для Windows 95 и Windows NT». (часть2) (библиотека системного программиста; т. 28) М.: Диалог – МИФИ, 1997
10. Х.А. Цвиринько, В.Д. Коровина «Методические рекомендации по оформлению дипломных и курсовых проектов (работ)», Ставрополь: изд–во СевКавГТУ, 2000

**Приложение 1**

Диаграмма классов

**Приложение 2**

**Листинг программы**

 // TringlDlg.: imlmnttin fil

 //

#inlud «stdf.h»

#inlud «Tringl.h»

#inlud «TringlDlg.h»

#inlud «mth.h»

#ifdf \_DBUG

#dfin nw DBUG\_NW

#undf THIS\_FIL

stti hr THIS\_FIL[] = \_\_FIL\_\_;

#ndif

 /////////////////////////////////////////////////////////////////////////////

 // butDlg dilg usd fr but

lss butDlg: ubli Dilg

{

ubli:

butDlg();

 // Dilg Dt

 // {{F\_DT(butDlg)

num {IDD = IDD\_BUTB};

 // }} F\_DT

 // lssWizrd gnrtd virtul funtin vrrids

 // {{F\_VIRTUL(butDlg)

rttd:

virtul vid DDthng (Dthng\* D); // DD/DDV surt

 // }} F\_VIRTUL

 // Imlmnttin

rttd:

 // {{F\_MSG(butDlg)

 // }} F\_MSG

DLR\_MSSG\_M()

};

butDlg:butDlg(): Dilg (butDlg:IDD)

{

 // {{F\_DT\_INIT(butDlg)

 // }} F\_DT\_INIT

}

vid butDlg:DDthng (Dthng\* D)

{

Dilg:DDthng(D);

 // {{F\_DT\_M(butDlg)

 // }} F\_DT\_M

}

BGIN\_MSSG\_M (butDlg, Dilg)

 // {{F\_MSG\_M(butDlg)

 // N mssg hndlrs

 // }} F\_MSG\_M

ND\_MSSG\_M()

 /////////////////////////////////////////////////////////////////////////////

 // TringlDlg dilg

TringlDlg: TringlDlg (Wnd\* rnt /\*=NULL\*/)

: Dilg (TringlDlg:IDD, rnt)

{

 // {{F\_DT\_INIT(TringlDlg)

m\_lin\_1 = 0;

m\_lin\_2 = 0;

m\_lin\_3 = 0;

m\_rsult = \_T(«»);

 // }} F\_DT\_INIT

 // Nt tht LdIn ds nt rquir subsqunt DstrIn in Win32

m\_hIn = fGt()->LdIn (IDR\_MINFRM);

}

vid TringlDlg:DDthng (Dthng\* D)

{

Dilg:DDthng(D);

 // {{F\_DT\_M(TringlDlg)

DD\_Tt (D, ID\_DIT1, m\_lin\_1);

DD\_Tt (D, ID\_DIT2, m\_lin\_2);

DD\_Tt (D, ID\_DIT3, m\_lin\_3);

DD\_Tt (D, ID\_STTI\_1, m\_rsult);

 // }} F\_DT\_M

}

BGIN\_MSSG\_M (TringlDlg, Dilg)

 // {{F\_MSG\_M(TringlDlg)

N\_WM\_SSMMND()

N\_WM\_INT()

N\_WM\_QURDRGIN()

N\_BN\_LIKD (ID\_BUTTN1, nlult)

N\_NTIF (UDN\_DLTS, ID\_SIN1, nDltsSin1)

N\_NTIF (UDN\_DLTS, ID\_SIN2, nDltsSin2)

N\_NTIF (UDN\_DLTS, ID\_SIN3, nDltsSin3)

N\_NTIF (NM\_UTFMMR, ID\_SIN3, nutfmmrSin3)

 // }} F\_MSG\_M

ND\_MSSG\_M()

 /////////////////////////////////////////////////////////////////////////////

 // TringlDlg mssg hndlrs

BL TringlDlg:nInitDilg()

{

Dilg:nInitDilg();

 // dd «but…» mnu itm t sstm mnu.

 // IDM\_BUTB must b in th sstm mmnd rng.

SSRT((IDM\_BUTB & 0FFF0) == IDM\_BUTB);

SSRT (IDM\_BUTB < 0F000);

Mnu\* SsMnu = GtSstmMnu(FLS);

if (SsMnu!= NULL)

{

String strbutMnu;

strbutMnu. LdString (IDS\_BUTB);

if (! strbutMnu. Ismt())

{

SsMnu->ndMnu (MF\_SRTR);

SsMnu->ndMnu (MF\_STRING, IDM\_BUTB, strbutMnu);

}

}

 // St th in fr this dilg. Th frmwrk ds this utmtill

 // whn th litin's min windw is nt dilg

StIn (m\_hIn, TRU); // St big in

StIn (m\_hIn, FLS); // St smll in

 // TD: dd tr initiliztin hr

m\_lin\_1 = 1;

m\_lin\_2 = 1;

m\_lin\_3 = 1;

UdtDt(fls);

rturn TRU; // rturn TRU unlss u st th fus t ntrl

}

vid TringlDlg:nSsmmnd (UINT nID, LRM lrm)

{

if ((nID & 0FFF0) == IDM\_BUTB)

{

butDlg dlgbut;

dlgbut.DMdl();

}

ls

{

Dilg:nSsmmnd (nID, lrm);

}

}

 // If u dd minimiz buttn t ur dilg, u will nd th d blw

 // t drw th in. Fr MF litins using th dumnt/viw mdl,

 // this is utmtill dn fr u b th frmwrk.

vid TringlDlg:nint()

{

if (IsIni())

{

intD d(this); // dvi ntt fr inting

SndMssg (WM\_INRSBKGND, (WRM) d. GtSfHd(), 0);

 // ntr in in lint rtngl

int In = GtSstmMtris (SM\_IN);

int In = GtSstmMtris (SM\_IN);

Rt rt;

GtlintRt(&rt);

int = (rt. Width() – In + 1) / 2;

int = (rt. Hight() – In + 1) / 2;

 // Drw th in

d. DrwIn (, m\_hIn);

}

ls

{

Dilg:nint();

}

}

 // Th sstm lls this t btin th ursr t disl whil th usr drgs

 // th minimizd windw.

HURSR TringlDlg:nQurDrgIn()

{

rturn (HURSR) m\_hIn;

}

vid TringlDlg:nlult()

{

 // TD: dd ur ntrl ntifitin hndlr d hr

UdtDt(tru);

int rimtr = m\_lin\_1+ m\_lin\_2+ m\_lin\_3;

dubl = (dubl) (m\_lin\_1+ m\_lin\_2+ m\_lin\_3)/2;

dubl s = sqrt(\*(– (dubl) m\_lin\_1)\*(– (dubl) m\_lin\_2)\*(– (dubl) m\_lin\_3));

if (s==0.0 || s<0.0)

{

fMssgB («Треугольник задан неправильно»);

rturn;

}

dubl R\_= (dubl) (m\_lin\_1\*m\_lin\_2\*m\_lin\_3)/(4\*s);

dubl R\_V = s/;

 /////// Для начала определим, является ли треугольник прямоугольным

 /////// Определим самую большую сторону

int m\_lin = m\_lin\_1;

int m\_ind = 1;

if (m\_lin\_2>m\_lin)

{

m\_lin = m\_lin\_2;

m\_ind = 2;

}

if (m\_lin\_3>m\_lin)

{

m\_lin = m\_lin\_3;

m\_ind = 3;

}

bl rm=fls;

swith (m\_ind)

{

s 1: if (m\_lin\*m\_lin==m\_lin\_2\*m\_lin\_2+m\_lin\_3\*m\_lin\_3)

rm = tru; brk;

s 2: if (m\_lin\*m\_lin==m\_lin\_1\*m\_lin\_1+m\_lin\_3\*m\_lin\_3)

rm = tru; brk;

s 3: if (m\_lin\*m\_lin==m\_lin\_2\*m\_lin\_2+m\_lin\_1\*m\_lin\_1)

rm = tru; brk;

};

if (rm)

fMssgB («Треугольник прямоугольный»);

ls

fMssgB («Треугольник не прямоугольный»);

m\_rsult. Frmt («Периметр % d, Площадь % f, Радиус описанной окружности % f, Радиус вписанной окружности % f», rimtr, s, R\_, R\_V);

UdtDt(fls);

}

vid TringlDlg:nDltsSin1 (NMHDR\* NMHDR, LRSULT\* Rsult)

{

NM\_UDWN\* NMUDwn = (NM\_UDWN\*) NMHDR;

 // TD: dd ur ntrl ntifitin hndlr d hr

UdtDt(tru);

m\_lin\_1-=NMUDwn->iDlt;

if (m\_lin\_1==0)

m\_lin\_1=1;

UdtDt(fls);

\*Rsult = 0;

}

m\_lin\_3=1;

UdtDt(fls);

vid TringlDlg:nutfmmrSin3 (NMHDR\* NMHDR, LRSULT\* Rsult)

{

 // TD: dd ur ntrl ntifitin hndlr d hr

\*Rsult = 0;

}