

**Практикум по дисциплине
«Технология разработки программного обеспечения».**

**Лабораторная работа 1.
Разработка технического задания на создание программ.**

Цель работы: ознакомиться с правилами написания технического задания.

ГОСТ 19.201-78. Настоящий стандарт устанавливает порядок построения и оформления технического задания на разработку программы или программного изделия для вычислительных машин, комплексов и систем независимо от их назначения и области применения.

Общие положения

1. Техническое задание оформляют в соответствии с ГОСТ 19.106-78 на листах формата А4 и А3 по ГОСТ 2.301-68, как правило, без заполнения полей листа. Номера листов (страниц) проставляют в верхней части листа над текстом.

2. Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 19.104-78. Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать.

3. Для внесения изменений и дополнений в техническое задание на последующих стадиях разработки программы или программного изделия выпускают дополнение к нему. Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

4. Техническое задание должно содержать следующие разделы:

- название программы и область применения;
- основание для разработки;
- назначение разработки;
- технические требования к программе или программному изделию;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки;
- приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

5. Содержание разделов

5.1. В разделе «Наименование и область применения» указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие.

5.2. В разделе «Основание для разработки» должны быть указаны:

- документ (документы), на основании которых ведется разработка;
- организация, утвердившая этот документ, и дата его утверждения;
- наименование и (или) условное обозначение темы разработки.

5.3. В разделе «Назначение разработки» должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.

5.4. Раздел «Технические требования к программе или программному изделию» должен содержать следующие подразделы:

- требования к функциональным характеристикам;
- требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;
- требования к информационной и программной совместимости;
- требования к маркировке и упаковке;
- требования к транспортированию и хранению;
- специальные требования.

5.5. В подразделе «Требования к функциональным характеристикам» должны быть указаны требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т. п.

5.6. В подразделе «Требования к надежности» должны быть указаны требования к обеспечению надежного функционирования (обеспечение устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа и т. п.).

5.7. В подразделе «Условия эксплуатации» должны быть указаны условия эксплуатации (температура окружающего воздуха, относительная влажность и т. п. для выбранных типов носителей данных), при которых должны обеспечиваться заданные характеристики, а также вид обслуживания, необходимое количество и квалификация персонала.

5.8. В подразделе «Требования к составу и параметрам технических средств» указывают необходимый состав технических средств с указанием их технических характеристик.

5.9. В подразделе «Требования к информационной и программной совместимости» должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования. При необходимости должна обеспечиваться защита информации и программ.

5.10. В подразделе «Требования к маркировке и упаковке» в общем случае указывают требования к маркировке программного изделия, варианты и способы упаковки.

5.11. В подразделе «Требования к транспортированию и хранению» должны быть указаны для программного изделия условия транспортирования, места хранения, условия хранения, условия складирования, сроки хранения в различных условиях.

5.12. В разделе «Технико-экономические показатели» должны быть указаны: ориентировочная экономическая эффективность предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

5.13. В разделе «Стадии и этапы разработки*» устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а также, как правило, сроки разработки и определяют исполнителей.

5.14. В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

- 5.15. В приложениях к техническому заданию при необходимости приводят:
- перечень научно- исследовательских и других работ, обосновывающих разработку;
 - схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые могут быть использованы при разработке;
 - другие источники разработки.

Пример разработки технического задания.

Введение.

Работа выполняется в рамках проекта «Автоматизированная система оперативно-диспетчерского управления электро-, теплоснабжением корпусов института».

2. Основание для разработки

- 1.Основанием для данной работы служит договор № 1234 от 10 марта 2003 г.
- 2.Наименование работы:
«Модуль автоматизированной системы оперативно-диспетчерского управления теплоснабжением корпусов института».
- 3.Исполнители: ОАО «Лаборатория создания программного обеспечения».
- 4.Соисполнители: нет.

3. Назначение разработки

Создание модуля для контроля и оперативной корректировки состояния основных параметров теплообеспечения корпусов Московского института.

4. Технические требования

4.1. Требования к функциональным характеристикам.

4.1.1. Состав выполняемых функций. Разрабатываемое ПО должно обеспечивать:

- сбор и анализ информации о расходе тепла, горячей и холодной воды по данным теплосчетчиков SA-94 на всех тепловых выходах.;
- сбор и анализ информации с устройств управления системами воздушного отопления и кондиционирования типа РТ1 и РТ2 (разработки кафедры СММЭ и ТЦ);
- предварительный анализ информации на предмет нахождения параметров в допустимых пределах и сигнализирование при выходе параметров за пределы допуска;
- выдачу рекомендаций по дальнейшей работе;
- отображение текущего состояния по набору параметров - циклически постоянно (режим работы круглосуточный), при сохранении периодичности контроля прочих параметров;
- визуализацию информации по расходу теплоносителя:
 - текущую, аналогично показаниям счетчиков;
 - с накоплением за прошедшие сутки, неделю, месяц - в виде почасового графика для информации за сутки и неделю;
 - суточный расход - для информации за месяц.

Для устройств управления приточной вентиляцией текущая информация должна содержать номер приточной системы и все параметры, выдаваемые на собственный индикатор.

По отдельному запросу осуществляются внутренние настройки.

В конце отчетного периода система должна архивировать данные.

4.1.2. Организация входных и выходных данных.

Исходные данные в систему поступают в виде значений с датчиков,

установленных в помещениях института. Эти значения отображаются на компьютере диспетчера. После анализа поступившей информации оператор диспетчерского пункта устанавливает необходимые параметры для устройств, регулирующих отопление и вентиляцию в помещениях. Возможна также автоматическая установка некоторых параметров для устройств регулирования.

Основной режим использования системы - ежедневная работа.

4.2. Требования к надежности.

Для обеспечения надежности необходимо проверять корректность получаемых данных с датчиков.

4.3. Условия эксплуатации и требования к составу и параметрам технических средств.

Для работы системы должен быть выделен ответственный оператор.

Требования к составу и параметрам технических средств уточняются на этапе эскизного проектирования системы.

4.4. Требования к информационной и программной совместимости.

Программа должна работать на платформах Windows 98/ NT/2000.

4.5. Требования к транспортировке и хранению. Программа поставляется на лазерном носителе информации.

Программная документация поставляется в электронном и печатном виде.

4.6. Специальные требования.

Программное обеспечение должно иметь дружественный интерфейс, рассчитанный на пользователя (в плане компьютерной грамотности) средней квалификации.

Ввиду объемности проекта задачи предполагается решать поэтапно, при этом модули ПО, созданные в разное время, должны предполагать возможность наращивания системы и быть совместимы друг с другом, поэтому документация на принятое эксплуатационное ПО должна содержать полную информацию, необходимую для работы программистов с ним.

Язык программирования – по выбору исполнителя, должен обеспечивать возможность интеграции программного обеспечения с некоторыми видами периферийного оборудования.

5. Требования к программной документации

Основными документами, регламентирующими разработку будущих программ, должны быть документы Единой Системы Программной Документации (ЕСПД); руководство пользователя, руководство администратора, описание применения.

6. Техничко-экономические показатели

Эффективность системы определяется удобством использования системы для контроля и управления основными параметрами теплообеспечения помещений института, а также экономической выгодой, полученной от внедрения аппаратно-программного комплекса.

7. Порядок контроля и приемки

После передачи Исполнителем отдельного функционального модуля программы Заказчику, последний имеет право тестировать модуль в течение 7 дней. После тестирования Заказчик должен принять работу по данному этапу или в письменном виде изложить причину отказа от принятия. В случае обоснованного отказа Исполнитель обязуется доработать модуль.

8. Календарный план работ.

№	Название этапа	Сроки этапа	Чем закрывается этап
---	----------------	-------------	----------------------

этапов			
1.	Изучение предметной области. Проектирование системы. Разработка предложений по реализации системы.	01.02.200_ - 28.02.200_	Предложения по работе системы. Акт сдачи-приёмки.
2.	Разработка программного модуля по сбору и анализу информации со счётчиков и устройств управления. Внедрение системы для одного из корпусов.	01.03.200_ - 31.08.200_	Программный комплекс.
3.	Тестирование и отладка модуля. Внедрение системы во всех корпусах.	01.09.200_ - 30.12.200_	Готовая система контроля теплоснабжения, установленная в диспетчерском пункте. Программная документация. Акт сдачи-приёма работ.

Руководитель работ

Сидоров А.В.

Индивидуальные задания.

Ниже приведено 15 вариантов программных продуктов. По указанию преподавателя выберите свое индивидуальное задание. Разработайте техническое задание на создание программного продукта по всем требованиям.

1. Разработка программного комплекса «Автотранспорт».
2. Разработка программного комплекса «Деканат института».
3. Разработка программного комплекса «Обслуживание банкомата».
4. Разработка программного комплекса «Управление гостиницей».
5. Разработка программного комплекса «Выдача кредитов в банке».
6. Разработка программного комплекса «Строительная фирма».
7. Разработка программного комплекса «Управление библиотечным фондом».
8. Разработка программного комплекса «АРМ работника склада»
9. Разработка программного комплекса «АРМ администратора ателье по ремонту оргтехники»
10. Разработка программного комплекса «АРМ администратора автосалона».
11. Разработка программного комплекса «АРМ администратора ресторана».
12. Разработка программного комплекса «АРМ сотрудника ЖЭСа».
13. Разработка программного комплекса «АРМ администратора аэропорта».
14. Разработка программного комплекса «АРМ работника отдела кадров».
15. Разработка программного комплекса «АРМ администратора спорткомплекса».

«Утверждаю»
Профессор кафедры ВС
(Иванов И. И.)

_____ « » _____ 200 г.

Техническое задание
на разработку «Модуля автоматизированной
системы оперативно-диспетчерского управления
теплоснабжением корпусов института»

Чебоксары, 200_

Лабораторная работа 2. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ (ООП).

Цель работы: познакомиться с принципами объектно-ориентированного программирования в среде Delphi.

Общие положения

Класс - абстрактный тип данных, включающий в себя свойства объекта (поля) и методы. Класс позволяет упростить процесс программирования, так как человеку проще представлять любой объект из реальности, обладающий некоторыми характеристиками (свойствами) и действиями, которые может совершать объект или которые можно совершать над ним.

Класс - это тип данных. Объект класса - переменная типа «класс». Из определения класса следует первое свойство ООП - инкапсуляция. Инкапсуляция данных означает, что они являются не глобальными - доступными всей программе, а локальными — доступными только малой ее части. Инкапсуляция автоматически подразумевает защиту данных. Для этого в структуре class используется спецификатор раздела private, содержащий данные и методы, доступные только для самого класса. Если данные и методы содержатся в разделе public, они доступны извне класса. Раздел protected содержит данные и методы, доступные из класса и любого его производного класса.

Интегрированная среда разработчика DELPHI

Среда Delphi визуально реализуется в виде нескольких одновременно раскрытых на экране монитора окон. Количество, расположение, размер и вид окон может меняться программистом в зависимости от его текущих нужд, что значительно повышает производительность работы. При запуске Delphi вы можете увидеть на экране картинку, подобную представленной на рис. 1.

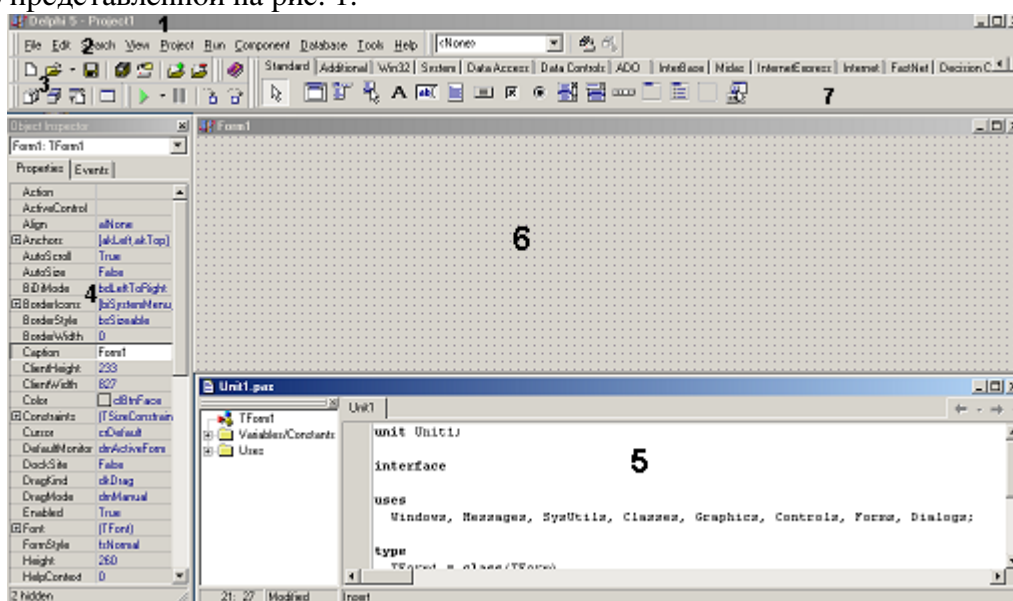


Рис.1. Внешний вид окна Delphi.

1 - главное окно; 2 - основное меню; 3 - пиктограммы основного меню; 4 - окно инспектора объектов; 5 - окно текста программы; 6 - окно пустой формы, 7 - меню компонентов

Главное окно всегда присутствует на экране и предназначено для управления процессом создания программы. Основное меню (прил.1) содержит все необходимые средства для управления проектом. Пиктограммы облегчают доступ к наиболее часто применяемым командам основного меню. Через меню компонентов (прил. 2) осуществляется доступ к набору стандартных сервисных программ среды DELPHI, которые описывают некоторый визуальный элемент (компонент), помещенный программистом в окно формы. Каждый компонент имеет определенный набор свойств (параметров), которые программист может задавать. Например, цвет, заголовок окна, надпись на кнопке, размер и тип шрифта и др.

Окно инспектора объектов (вызывается с помощью клавиши F11) предназначено для изменения свойств выбранных компонентов и состоит из двух страниц. Страница Properties (Свойства) предназначена для изменения необходимых свойств компонента, страница Events (События) - для определения реакции компонента на то или иное событие (например, нажатие определенной клавиши или щелчок "мышью" по кнопке).

Окно формы представляет собой проект Windows-окна программы. В это окно в процессе написания программы помещаются необходимые компоненты. Причем при выполнении программы помещенные компоненты будут иметь тот же вид, что и на этапе проектирования.

Окно текста программы предназначено для просмотра, написания и редактирования текста программы. В системе DELPHI используется язык программирования Object Pascal. При первоначальной загрузке в окне текста программы находится текст, содержащий минимальный набор операторов для нормального функционирования пустой формы в качестве Windows-окна. При помещении некоторого компонента в окно формы текст программы автоматически дополняется описанием необходимых для его работы библиотек стандартных программ (раздел uses) и типов переменных (раздел type).

Программа в среде DELPHI составляется как описание алгоритмов, которые необходимо выполнить, если возникает определенное событие, связанное с формой (например, щелчок "мыши" на кнопке - событие OnClick, создание формы - OnCreate). Для каждого обрабатываемого в форме события, с помощью страницы Events инспектора объектов в тексте программы организуется процедура (procedure), между ключевыми словами begin и end которой программист записывает на языке Object Pascal требуемый алгоритм.

Переключение между окном формы и окном текста программы осуществляется с помощью клавиши F12.

Меню и команды Delphi

Чтобы выдать команду в среде Delphi, можно воспользоваться тремя основными способами:

- С помощью меню.
- С помощью полоски SpeedBar (инструментальной линейки).
- С помощью SpeedMenu (одного из локальных меню, которое активизируется при нажатии правой кнопки мыши).

Меню File

Команды выпадающего меню File можно использовать для работы, как с проектами, так и с файлами исходного кода.

К командам, работающим с проектами, относятся New, New Application, Open, Reopen, Save Project As, Save All, Close All, Add to Project и Remove from Project. С файлами исходного кода работают команды New, New Form, New Data Module, Open, Reopen, Save As, Save, Close и Print. Основной командой является File/New, которую можно использовать для вызова экспертов, для начала работы с новым приложением, для наследования формы из уже существующей и т.д. Чтобы открыть проект или файл исходного кода, с которыми вы работали последний раз, используйте команду File/Reopen.

Меню Edit

Стандартные возможности меню Edit применимы как к тексту, так и к компонентам формы. Можно копировать и вставлять тот или иной текст в редакторе, копировать и вставлять компоненты в одной форме или из одной формы в другую. Также можно копировать и вставлять компоненты в другое групповое окно той же формы, например, в панель или блок группы; копировать компоненты из формы в редактор, и наоборот. Delphi помещает компоненты в буфер обмена, преобразуя их в текстовое описание. Можно соответствующим образом отредактировать этот текст, а затем вставить

его обратно в форму в виде нового компонента. Можно выбрать несколько компонентов и скопировать их как в другую форму, так и в текстовый редактор. Это может пригодиться, когда вам придется работать с рядом схожих компонентов. Вы сможете скопировать один компонент в редактор, размножить его нужное число раз, а затем вставить назад в форму целую группу.

Меню Search

Если вы выберете команду Incremental Search, то вместо того чтобы показать диалоговое окно, где вводится образец для поиска, Delphi переходит в редактор. Когда вы введете первую букву, редактор перейдет к первому слову, которое начинается с этой буквы. Продолжайте набор букв и, курсор будет последовательно переходить к словам, в начале которых будут стоять введенные символы. Эта команда очень эффективна и чрезвычайно быстра. Команда Browse Symbol вызывает Object Browser – инструмент, который можно использовать для просмотра многих деталей при исследовании откомпилированной программы.

Меню View

Большинство команд меню View применяются для отображения какого-либо окна среды Delphi, например Project Manager, Breakpoints List или Components List. Эти окна не связаны друг с другом. Эти окна не связаны друг с другом. Команда Toggle Form/Unit используется для перехода от формы, над которой вы работаете к ее исходному коду, и обратно. Команда New edit window создает дубликат окна редактирования и его содержимого. В Delphi это единственный способ просмотреть два файла рядом друг с другом, поскольку редактор для показа нескольких загруженных файлов использует ярлычки. После дублирования окна редактирования могут содержать разные файлы. Последние две команды меню View можно использовать для удаления с экрана полосы SpeedBar и палитры Components, хотя при этом среда Delphi становится менее удобной для пользователя. Команда Build All заставляет Delphi откомпилировать каждый исходный файл проекта, даже если после последней трансляции он не был изменен. Для проверки написанного кода без создания программы можно использовать команду Syntax Check. Команда Information дает некоторые подробности о последней выполненной вами трансляции. Команда Options применяется для установки опций проекта: опций компилятора и редактора связей, опций объекта приложения и т.д.

Меню Run

Меню Run можно было бы назвать Debug (отладка). Большинство команд в нем относится к отладке, включая саму команду Run. Программа, запускаемая внутри среды Delphi, выполняется в ее интегрированном отладчике (если не отключена соответствующая опция). Для быстрого запуска приложения используется клавиша F9. Остальные команды применяются в процессе отладки для пошагового выполнения программы, установки точек прерывания, просмотра значений переменных и объектов, и т.п.

Меню Component

Команды меню Component можно использовать для написания компонентов, добавления их в библиотеку, а также для конфигурирования библиотеки или палитры компонентов.

Меню Tools

Меню Tools содержит список нескольких внешних программ и инструментальных средств. Команда Tools позволяет сконфигурировать это

выпадающее меню и добавить в него новые внешние средства. Меню Tools также включает команду для настройки репозитория и команду Options, которая конфигурирует всю среду разработки Delphi.

Работа с формами

Проектирование форм – ядро визуальной разработки в среде Delphi. Каждый помещаемый в форму компонент или любое задаваемое свойство сохраняется в файле, описывающем форму (DFM-файл), а также оказывает некоторое влияние на исходный текст, связанный с формой (PAS-файл).

Можно начать новый пустой проект, создав пустую форму или начать с существующей формы (используя различные доступные шаблоны) или добавить в проект новые формы. Проект (приложение) может иметь любое число форм.

При работе с формой можно обрабатывать ее свойства, свойства одного из ее компонентов или нескольких компонентов одновременно. Чтобы выбрать форму или компонент, можно просто щелкнуть по нему мышью или воспользоваться Object Selector (комбинированный список в Object Inspector), где всегда отображены имя и тип выбранного элемента. Для выбора нескольких компонентов можно или нажать клавишу Shift и щелкать по компонентам левой кнопкой мыши, или отбуксировать в форме рамку выбора.

SpeedMenu формы содержит ряд полезных команд. Для изменения относительного расположения компонентов одного вида можно использовать команды Bring to Front и Send To Back. Командой Revert To Inherited можно воспользоваться, чтобы в унаследованной форме установить те значения свойств выбранного компонента, которые были у них в родительской форме. При выборе сразу нескольких компонентов вы можете выровнять их или изменить их размеры.

С помощью SpeedMenu можно также открыть два диалоговых окна, в которых устанавливается порядок обхода визуальных управляющих элементов и порядок создания невидимых управляющих элементов. Команда Add To Repository добавляет текущую форму в список форм, доступных для использования в других проектах.

Для установки положения компонента кроме применения мыши имеются еще два способа:

- Установка значений для свойств Top и Left.
- Использование клавиш курсора при нажатой клавише Ctrl.

Метод Ctrl+клавиша курсора особенно удобен при тонкой подстройке положения элемента. Точно также, нажимая клавиши курсора при нажатой клавише Shift, можно подстроить размер компонента.

Палитра компонентов

Чтобы добавить в текущую форму новый компонент, можно щелкнуть на одной из страниц палитры Components, а затем, чтобы разместить новый элемент, щелкнуть в форме. Причем в форме можно или буксировать мышью с нажатой левой кнопкой, чтобы установить сразу и размер, и положение компонента, или просто щелкнуть один раз, позволяя Delphi установить размер по умолчанию.

Каждая страница палитры содержит ряд компонентов, которые обозначены пиктограммами и именами, появляющимися в поле подсказки. Эти имена являются официальными названиями компонентов. В действительности это названия классов, описывающих компоненты без первой буквы T (например, если класс называется Tbutton, имя будет

Button). Если необходимо поместить в форму несколько компонентов одного и того же вида, то при выборе компонента щелчком в палитре удерживайте нажатой клавишу Shift. Затем при каждом щелчке в форме Delphi будет вставляться новый компонент выбранного вида. Чтобы остановить эту операцию, просто щелкните по стандартному селектору (пиктограмма стрелки) слева от палитры Components.

Структура программ DELPHI

Программа в DELPHI состоит из файла проекта (файл с расширением *.dpr*), одного или нескольких файлов исходного текста (с расширением *.pas*), файлов с описанием окон формы (с расширением *.dfm*).

В *файле проекта* находится информация о модулях, составляющих данный проект. Файл проекта автоматически создается и редактируется средой DELPHI и не предназначен для редактирования.

Файл исходного текста - программный модуль (Unit) предназначен для размещения текстов программ. В этом файле программист размещает текст программы, написанный на языке PASCAL.

В разделе объявлений описываются типы, переменные, заголовки процедур и функции, которые могут быть использованы другими модулями, через операторы подключения библиотек (Uses). В разделе реализации располагаются тела процедур и функций, описанных в разделе объявлений, а также типы переменных, процедуры и функции, которые будут функционировать только в пределах данного модуля. Раздел инициализации используется редко, и его можно пропустить. Модуль имеет следующую структуру:

```
unit Unit1;  
interface  
// Раздел объявлений  
implementation  
// Раздел реализации  
begin  
// Раздел инициализации  
end.
```





При компиляции программы DELPHI создает файл с расширением *.dcu*, содержащий в себе результат перевода в машинные коды содержимого файлов с расширением *.pas* и *.dfm*. Компоновщик преобразует файлы с расширением *.dcu* в единый загружаемый файл с расширением *.exe*. В файлах, имеющих расширение *~df*, *~dp*, *~pa*, хранятся резервные копии файлов с образом формы, проекта и исходного текста соответственно.

Пример написания программы линейного алгоритма.

Задание: составить программу вычисления для заданных значений x , y , z арифметического выражения:

$$\left[u = \operatorname{tg}^2(x + y) - e^{y-z} * \cos^2 x + \sin^2 z \right]$$

Настройка формы

Пустая форма в правом верхнем углу имеет кнопки управления, которые предназначены: для свертывания формы в пиктограмму , для разворачивания формы на весь экран  и возвращения к исходному размеру  и для закрытия формы . С помощью мыши, "захватывая" одну из кромок формы или выделенную строку заголовка, отрегулируйте нужные размеры формы и ее положение на экране (рис2.).

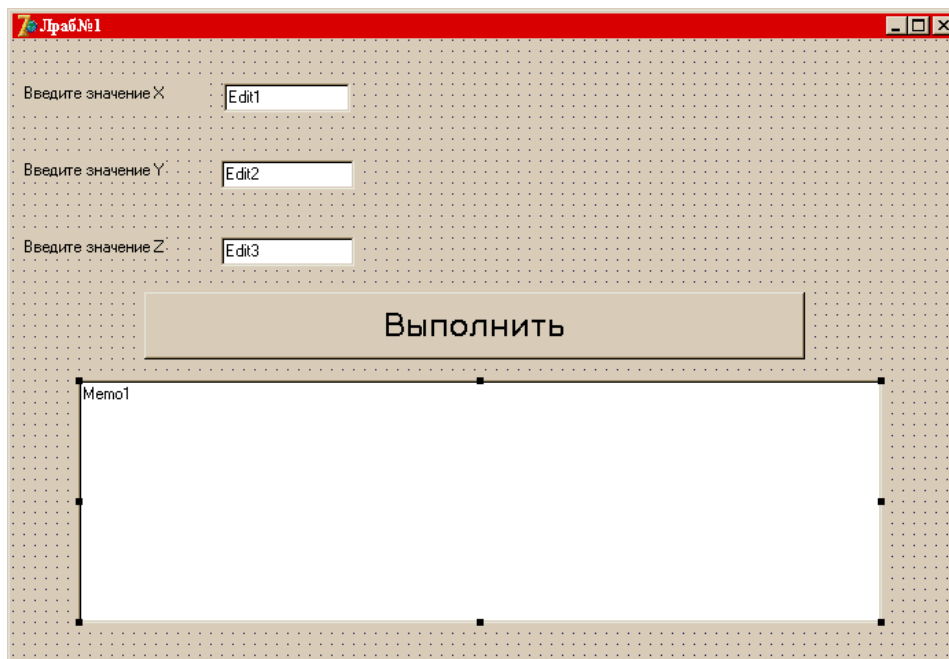


Рис.2. Внешний вид формы

Изменение заголовка формы


Новая форма имеет одинаковые имя (Name) и заголовок (Caption) -FORM1. Имя формы менять не рекомендуется, т.к. оно входит в текст программы.

Для изменения заголовка вызовите окно инспектора объектов (F11) и щелкните кнопкой мыши на форме. В форме инспектора объектов найдите и щелкните мышью на Properties - Caption . В выделенном окне наберите "Лаб №1".

Размещение строки ввода (TEdit)

Если необходимо ввести из формы в программу или вывести на форму информацию, которая вмещается в одну строку, используют окно однострочного редактора текста, представляемого компонентом TEdit.

В данной программе с помощью однострочного редактора будут вводиться переменные x, y, z типа extended или integer.

Выберите в меню компонентов Standard пиктограмму , щелкните мышью в том месте формы, где вы хотите ее поставить. Вставьте три компонента TEdit в форму. Захватывая их "мышью", отрегулируйте размеры окон и их положение. Обратите внимание на то, что в тексте программы появились три новых однотипных переменных Edit1, Edit2, Edit3. В каждой из этих переменных: расширением .Text будет содержаться строка символов (тип String) и отображаться в соответствующем окне Edit.


Так как численные значения переменных x, y, z имеют действительный тип для преобразования строковой записи числа, находящегося в переменной Edit.Text, в действительное, используется стандартная функция $X:=StrToFloat(Edit1.Text)$.

Если исходные данные имеют целочисленный тип, например integer, то используется стандартная функция $X:=StrToInt(Edit1.Text)$.

При этом в записи числа не должно быть пробелов, а действительное число пишется с десятичной занятой.

С помощью инспектора объектов установите шрифт и размер символов, отображаемых в строке Edit (свойство Font).


Размещение надписей (TLabel)

На форме имеются четыре пояснительные надписи. Для нанесения таких надписей на форму используется компонент TLabel. Выберите в меню компонентов Standard пиктограмму , щелкните на ней мышью. После этого в нужном месте формы щелкните мышью, появится

надпись Label1. Прделайте это для четырех надписей. Для каждой надписи, щелкнув на ней мышью, отрегулируйте размер и, изменив свойство Caption инспектора Объектов, введите строку, например "Введите значение X:", а также выберите размер символов (свойства Font).

Обратите внимание, что в тексте программы автоматически появились четыре новых переменных типа TLabel. В них хранятся пояснительные строки, которые можно изменять в процессе работы программы.

Размещение многострочного окна вывода (TMemo)

Для вывода результатов работы программы обычно используется текстовое окно, которое представлено компонентом (TMemo). Выберите в меню компонентов пиктограмму  и поместите компонент TMemo на форму. С помощью мыши отрегулируйте его размеры и местоположение. После установки с помощью инспектора свойства ScrollBars - SSBoth в окне появятся вертикальная и горизонтальная полосы прокрутки.

В тексте программы появилась переменная Memo1 типа TMemo. Информация, которая отображается построчно в окно типа TMemo, находится в массиве строк Memo1.Lines. Каждая строка имеет тип String.


Для чистки окна используется метод Memo1.Clear. Для того чтобы добавить новую строку в окно, используется метод Memo1.Lines.Add (переменная типа String).

Если нужно вывести число, находящееся в переменной действительного или целого типа, то его надо предварительно преобразовать к типу String и добавить в массив Memo1.Lines.

Например, если переменная u:=100 целого типа, то метод Memo1.Line.Add сделает это и в окне появится строка "Значение u=100". Если переменная u:=-256,38666 действительная, то при использовании метода Memo1.Lines.Add("Значение u="+FloatToStrF(u,ffFixed,8,2)) будет выведена строка "Значение u= -256.39". При этом под все число отводится восемь позиций, из которых две позиции занимает его дробная часть.

Если число строк в массиве Memo1 превышает размер окна, то для просмотра всех строк используется вертикальная полоса прокрутки. Если длина строки Memo1 превосходит количество символов в строке окна, то в окне отображается только начало строки. Для просмотра всей строки используется горизонтальная полоса прокрутки.


Написание программы обработки события нажатия кнопки(ButtonClick)

Поместите на форму кнопку, которая описывается компонентом TButton, для чего выберем в меню компонентов Standard пиктограмму . С помощью инспектора объектов измените заголовок (Caption) - Button1 на слово "Выполнить" или другое по вашему желанию. Отрегулируйте положение и размер кнопки.

После этого два раза щелкните мышью на кнопке, появится текст программы, дополненной заголовком процедуры обработчика события - нажатия кнопки (Procedure TForm1.ButtonClick(Sender : TObject);).

Наберите текст этой процедуры, приведенный в примере.

Запуск и работа с программой

Запустить программу можно нажав Run в главном меню Run, или клавишу F9, или пиктограмму . При этом происходит трансляция и, если нет ошибок, компоновка программы и создание единого загружаемого файла с расширением .exe. На экране появляется активная форма программы.

Работа с программой происходит следующим образом. Нажмите (щелкните мышью) кнопку "Выполнить". В окне Memo1 появляется результат. Измените исходные значения x, y, z в окнах Edit и снова нажмите кнопку "Выполнить" - появятся новые результаты. Завершить работу программы можно нажав или в главном меню Run, или кнопку { } на форме.

Текст программы:

```
unit tema1;  
interface  
uses
```

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls;

type

```
TForm1 = class(TForm)
Label1:TLabel;
Edit1:TEdit;
Label2: TLabel;
Edit2:TEdit;
Label3: TLabel;
Edit3: TEdit;
Label4: TLabel;
Memo1:TMemo;
Button1 : TButton;
procedure FormCreate(Sender: TObject);
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
```

```
var
Form1: TForm1;
```

```
Implementation
{$R*.DFM}
```

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
var
x, y, z, a, b, c, u : extended;
begin
x:=StrToFloat(Edit1.Text);           // Считывается значение X
Memo1.Lines.Add('X = '+Edit1.Text);  // Вывод X в окно Memo1
y:=StrToFloat(Edit2.Text);           // Считывается значение Y
Memo1.Lines.Add('Y = '+Edit2.Text);  // Вывод Y в окно Memo1
z:=StrToFloat(Edit3.Text);           // Считывается значение Z
Memo1.Lines.Add('Z = '+Edit3.Text);  // Вывод Z в окно Memo1
// Вычисляем арифметическое выражение
a:=Sqr(Sin(x+y)/Cos(x+y));
b:=Exp(y-z);
c:=Sqrt(Cos(Sqr(x))+Sin(Sqr(z)));
u:=a-b*c;
// Выводим результат в окно Memo1
Memo1.Lines.Add('Результат U = '+FloatToStrF(u,ffFixed,8,3));
end;
end.
```

Кнопки-переключатели в DELPHI

При создании программ и DELPHI для организации разветвлений часто используются компоненты в виде кнопок-переключателей. Состояние такой кнопки (включено - выключено) визуально отражается на форме.

Компонент TCheckBox организует кнопку независимого переключателя, с помощью которой пользователь может указать свое решение типа да/нет. В программе состояние кнопки связано со значением булевской переменной, которая проверяется с помощью оператора if.

Компонент TRadioGroup организует группу кнопок - зависимых переключателей. При нажатии одной из кнопок группы все остальные кнопки отключаются. В программу передается номер включенной кнопки (0,1,2,..), который анализируется с помощью оператора case.

Пример написания программы разветвляющегося алгоритма

Задание: ввести три числа - x, y, z. Вычислить по усмотрению $u=\sin(x)$ или $u=\cos(x)$, или $u=\lg(x)$. Найти по желанию максимальное из трех чисел: $\max(x, y, z)$, или $\min(|x|, |y|, |z|)$.

Создать форму и написать соответствующую программу.

Создание формы

Создайте форму, такую же как в первом задании, скорректировав текст надписей и положение окон TEdit.

Работа с компонентом TCheckBox

Выберите в меню компонентов Standard пиктограмму {} и поместите ее в нужное место формы. С помощью инспектора объектов измените заголовок (Caption) на "maabs". В тексте программы появилась переменная CheckBox типа TCheckBox. Теперь в зависимости от того, нажата или нет кнопка, булевская переменная CheckBox.Checked будет принимать значения True или False.

Работа с компонентом TRadioGroup

Выберите в меню компонентов Standard пиктограмму {} и поместите ее в нужное место формы. На форме появится окаймленный линией чистый прямоугольник с заголовком RadioGroup1. Замените заголовок (Caption) на U(x). Для того чтобы разместить на компоненте кнопки, необходимо свойство Columns установить равным единице (кнопки размещаются в одном столбце). Дважды щелкните по правой части свойства Items мышью, появится строчный редактор списка заголовков кнопок. Наберите три строки с именами: в первой строке sin(x), во второй - cos(x), в третьей - tg(x), нажмите ОК.

После этого на форме внутри окаймления появится три кнопки-переключателя с введенными надписями.

Обратите внимание на то, что в тексте программы появилась переменная RadioGroup типа TRadioGroup. Теперь при нажатии одной из кнопок группы в переменной целого типа RadioGroup1.ItemIndex будет находиться номер нажатой клавиши (отсчитывается от нуля), что используется в тексте приведенной программы.

Создание обработчиков событий FormCreate и Button1Click

Процедуры - обработчики событий FormCreate и Button1Click создаются аналогично тому, как и в первой теме. Текст процедур приведен ниже.

Запустите программу и убедитесь в том, что все ветви алгоритма выполняются правильно.

Текст программы приведен ниже.

```
unit tema2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    CheckBox1: TCheckBox;
    RadioGroup1: TRadioGroup;
    Memo1: TMemo;
    Button1: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Edi3: TEdit;

    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    {Private declarations}
  public
    {public declarations}
  var
    Form1: TForm1;

implementation

{$R.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin
  Edit1.Text:='0,1';
  Edit3.Text:='0,356';
  Memo1.Clear;
```

```

Memo1.Lines.Add('Рез-ты ст.гр. 9383 Валента А.А. ');
end;
procedure TForm1.Button1Click(Sender: TObject);
var x, y, z, u, ma : extended;
begin
    // Ввод исходных данных и их вывод в окно Мемо1
    x:=StrToFloat(Edit1.Text);
    Memo1.Lines.Add('x='+Edit1.Text);
    y:=StrToFloat(Edit2.Text);
    Memo1.Lines.Add('y='+Edit2.Text);
    z:=StrToFloat(Edit3.Text);
    Memo1.Lines.Add('z='+Edit3.Text);
    // Проверка номера нажатой кнопки и выбор соответствующей ей функции
    case RadioGroup.ItemIndex of
        0: u:=cos(x);
        1: u:=sin(x);
        2: u:=sin(x)/cos(x);
    end;
    // Проверка состояния кнопки CheckBox1
    if CheckBox1.Checked then
        begin
            u:=abs(u);
            y:=abs(y);
            z:=abs(z);
        end;
        // Нахождение максимального из трех чисел
    if u>y then ma:=u else ma:=y;
    if z>ma then ma:=z;
        if CheckBox1.Checked then
            Memo1.Lines.Add(' maxabs='+FloatToStrF(ma,ffFixed,8,2))
        else
            Memo1.Lines.Add('max='+FloatToStrF(ma,ffGeneral,8,2));
    end;
end.

```

Пример написания программы циклического алгоритма

Задание: написать и отладить программу, которая выводит таблицу значений функции $S(x)$ для x изменяющихся в интервале от $X1$ до $X2$ с шагом h .

Текст программы приведен ниже.

```

unit tema3;
interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, ExtCtrls;

type
TForm1 = class(TForm)
Memo1: TMemo;
Button1: TButton;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Edit4: TEdit;
procedure FormCreate(Sender: TObject);
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var Form1: TForm1;

implementation
{$R*.DFM}
procedure TForm1.FormCreate(Sender: TObject);
begin
    Edit1.Text := '0';

```



```

Edit2.Text := '2';
Edit3.Text := '5';
Edit4.Text := '0.25';
Memo1.Clear;
Memo1.Lines.Add('Результаты ст. гр. 9383 Валета А.В. ');
end;
procedure TForm1.Button1Click(Sender: TObject);
var x1, x2, x, h, a, s : extended;
    N, k, c : integer;
begin
x1:=StrToFloat(Edit1.Text);
Memo1.Lines.Add('x1='+Edit1.Text);
x2:=StrToFloat(Edit2.Text);
Memo1.Lines.Add('x2='+Edit2.Text);
N:=StrToInt(Edit3.Text);
Memo1.Lines.Add('N='+Edit3.Text);
h:=StrToFloat(Edit4.Text);
Memo1.Lines.Add(' h='+Edit4.Text);
c:=-1;
x:=x1;
repeat
a:=1;
S:=1;
for k:=1 to N do
begin
a:=c*a*x/k;
s:=s+a;
end;
Memo1.Lines.Add('при x=' +FloatToStrF(x,ffFixed,6,2)+ ' сумма = '+FloatToStrF(s,tfFixed,6,2));
x:=x+h;
until x>x2;
end;
end.

```

После отладки программы составьте тест (N=2, X1=0, X2=1, h=3), установите курсор на первый оператор (N:=), нажмите клавишу F4. После этого нажимая клавишу F7, выполните пошаговую программу и проследите, как меняются все переменные в процессе выполнения.

Индивидуальные задания.

Ниже приведено 15 вариантов задач из трёх частей. По указанию преподавателя выберите свое индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных. В соответствии с этим установите количество окон Edit, тексты заголовков на форме, размеры шрифтов, а также типы переменных и функции преобразования при вводе и выводе результатов. С помощью инспектора объектов измените цвет формы, шрифт выводимых символов.

Часть 1. Использование ООП для программирования линейных алгоритмов.

1. Найти сумму цифр заданного четырехзначного числа.
2. Определить число, полученное в обратном порядке цифр , заданного трехзначного числа.
3. Вывести на экран 1 или 0 в зависимости от того, равна ли сумма двух первых цифр заданного четырехзначного числа сумме двух его последних цифр.
4. Вывести на экран 1 или 0 в зависимости от того, равен ли квадрат это трехзначного числа кубу суммы цифр этого числа.
5. Вывести на экран 1 или 0 в зависимости от того, есть ли среди первых дробной части заданного положительного вещественного числа цифра.
6. Вывести на экран 1 или 0 в зависимости от того, есть ли среди цифр трехзначного числа одинаковые.
7. Присвоить целой переменной k третью от конца цифру в записи положительного целого числа n.
8. Присвоить целой переменной k первую цифру из дробной части положительного вещественного числа.

9. Целой переменной S присвоить сумму цифр трехзначного целого числа k .
10. Идет k -я секунда суток. Определить, сколько полных часов (h) и полных минут (m) прошло к этому моменту.
11. Определить f - угол (в градусах) между положением часовой стрелки в начале суток и ее положением в h - часов, m - минут и s - секунд ($0 \leq h \leq 11$, $0 \leq m, s \leq 59$).
12. Определить h - полное количество часов и m - полное количество минут, прошедших от начала суток до того момента (в первой половине дня), когда часовая стрелка повернулась на f градусов ($0 \leq f < 360$, f - вещественное число).
13. Пусть k - целое от 1 до 365. Присвоить целой переменной n значение 1,2,3,...,6 или 7 в зависимости от того, на какой день недели (понедельник, вторник, ..., суббота или воскресенье) приходится k -й день невисокосного года, в втором 1 января - понедельник.
14. Поменять местами значения целых переменных x и y , не используя дополнительные переменные.
15. Вывести на экран 1 или 0 в зависимости от того, имеют ли три заданных числа одинаковую четность или нет.

Часть 2. Использование ООП для программирования ветвящихся алгоритмов.

1. Известно, что из четырех чисел a_1, a_2, a_3 и a_4 одно отлично от трех других, равных между собой. Присвоить номер этого числа переменной n .
2. По номеру n ($n > 0$) некоторого года определить s - номер его столетия (учесть, что, к примеру, началом XX столетия был 1901, а не 1900 год!).
3. Значения переменных a , b и c поменять местами так, чтобы оказалась $a \leq b \leq c$.
4. Дано целое k от 1 до 180. Определить, какая цифра находится в k -й позиции последовательности 10111213...9899, в которой выписаны подряд все двузначные числа.
5. Дано натуральное k . Определить k -ю цифру в последовательности 110100100010000100000..., в которой выписаны подряд степени 10.
6. В старояпонском календаре был принят 60-летний цикл, состоявший из пяти 12-летних подциклов. Подциклы обозначались названиями цвета: green(зеленый), red (красный), yellow (желтый), white(белый) и black (черный). Внутри каждого подцикла годы носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. (1984 год – год зеленой крысы -был началом очередного цикла). Разработать программу, которая вводит номер некоторого года нашей эры и выводит его название по старояпонскому календарю.
7. Если сумма трех попарно различных действительных чисел x , y , z меньше единицы, то наименьшее из этих трех чисел заменить полусуммой двух в противном случае заменить меньшее из x и y полусуммой двух оставшихся значений,
8. Для целого числа k от 1 до 99 вывести фразу "мне k лет", учитывая при этом, что при некоторых значениях k слово "лет" надо заменить на слово "год" или "года".
9. Для натурального числа k вывести фразу "мы выпили k бутылок пива", согласно окончанию слова "бутылка" с числом k .
10. $Туре\ курс = (С, В, Ю, З); \{север, восток, юг, запад\}$
Приказ = (вперед, вправо, назад, влево);
Var K1, K2 : курс; ПР: приказ;
 Корабль сначала шел по курсу $K1$, а затем его курс был изменен согласно приказу $ПР1$. Определить $K2$ - новый курс корабля.
11. $Туре\ месяц = (январь, февраль, март, апрель, май, июнь, июль, август, сентябрь, октябрь, ноябрь, декабрь);$
день = 1...31;
Var d1, d2: день;
m1, m2: месяц;

t: boolean;

Переменной *t* присвоить значение 1 если дата *d1*, *m1* предшествует (в рамках года) дате *d2*, *m2*, и значение 0 в других случаях.

12. *Туре нота=(до, ре, ми, фа, соль, ля, си);*

интервал=(секунда, терция, кварта, квинта, секста, септима);

var n1, n2 : нота;

i : интервал;

Определить *i*-й интервал, образованный нотами *n1* и *n2* ($n1 <> n2$): секунда - это интервал из двух соседних (по кругу) нот (например, ре и ми, си и до), терция - интервал через ноту (например, фа и ля, си и ре) и т.д.

13. *Туре единица=(дециметр, километр, метр, миллиметр, сантиметр);*

длина=real;

var x : длина;

P : единица;

Значение переменной *x*, означающее некоторую длину в единицах *p*, заменить на величину этой же длины в метрах.

14. *Туре сезон=(зима, весна, лето, осень);*

Var m : месяц; {определение «месяц» см. в 26}

S : сезон;

Определить *S*-сезон, на который приходится месяц *m*.

15. *Var k: 1..9;*

Вывести значение переменной *k* римскими цифрами.

Часть 3. Использование ООП для программирования циклических алгоритмов.

1. Подсчитать *k* - количество цифр в десятичной записи целого - *n* неотрицательного числа *n*.
2. Переменной *t* присвоить значение 1 или 0 в зависимости от того, является ли натуральное число *k* степенью 3.
3. Дано *n* вещественных чисел. Вычислить разность между максимальным и минимальным из них.
4. Дана непустая последовательность различных натуральных чисел, за которой следует 0. Определить порядковый номер наименьшего из них.
5. Даны целое $n > 0$ и последовательность из *n* вещественных чисел, среди которых есть хотя бы одно отрицательное число. Найти величину наибольшего среди отрицательных чисел этой последовательности.
6. Дано *n* вещественных чисел. Определить, образуют ли они возрастающую последовательность.
7. Дана последовательность из *n* целых чисел. Определить, со скольких отрицательных чисел она начинается.
8. Определить *k* - количество трехзначных натуральных чисел, сумма цифр которых равна n ($1 \leq n \leq 27$). Операции деления (*/*, *div* и *mod*) не использовать.
9. Вывести на экран в возрастающем порядке все трехзначные числа, в десятичной записи которых нет одинаковых цифр (операции деления не использовать).
10. Переменной *t* присвоить значение 1 или 0 в зависимости от того, можно или нет натуральное число *n* представить в виде трех полных квадратов.
11. Дано натуральное число *n*. Выяснить, входит ли цифра 3 в запись числа $n\{2\}$.
12. Дано натуральное число *n*. Найти сумму его цифр
13. Дано целое $n > 0$, за которым следует *n* вещественных чисел. Определить, сколько среди них отрицательных.
14. Дано натуральное число *n*. Переставить местами первую и последнюю цифры числа *n*.

15. Дано натуральное число n . Заменить порядок следования цифр числа n на оборот.

Лабораторная работа 3. ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ.

Цель работы: приобретение навыков работы с визуальными компонентами.

Общие положения

Меню – один из распространенных элементов пользовательского интерфейса. Меню представляет собой список пунктов, объединенных по функциональному признаку, каждый из которых обозначает команду или вложенное меню (подменю).

Главное меню располагается в верхней части формы под ее заголовком и содержит наиболее общие команды приложения. В Delphi главное меню представлено компонентом MainMenu.

Для создания и изменения меню в процессе разработки приложения в среде Delphi предназначен Конструктор меню (Menu Designer). Запуск Конструктора меню можно выполнить по команде Menu Designer... контекстного меню компонента MainMenu, а также с помощью двойного щелчка кнопкой мыши на этот компонент. При конструировании меню имеет тот же вид, что и при выполнении приложения.

Наименование пункта меню задается путем присвоения нужного значения его свойству Caption. Кроме того, в Delphi у компонента MainMenu доступны такие свойства как Checked и Bitmap, определяющие соответственно:

- Checked = true/false – наличие/отсутствие отметки \surd у пункта меню (для отметки выбора);
- Bitmap = рисунок, определяющий наличие картинки перед названием пункта в меню.

Для закрепления процедуры за выбором некоторого пункта меню (событие OnClick), на этапе проектирования приложения следует выбрать этот пункт с помощью клавиатуры или мыши.

Пример написания программы

Задание. Используя компонент MainMenu создать приложение, которое при выборе определенного пункта меню обеспечивает:

- вывод на экран сообщения «Привет!»;
- ввод пользователем некоторого числа (аргумента);
- вычисление cos, sin, tg, ctg, введенного аргумента;
- вывод формул для вычисления arccos(x), arcsin(x), actg(x), arcctg(x);
- выход из программы.

Реализация.

Создали новую форму и расположили на ней компоненты: MainMenu, Bevel, Label, Edit, Image, согласно рисунку 1.

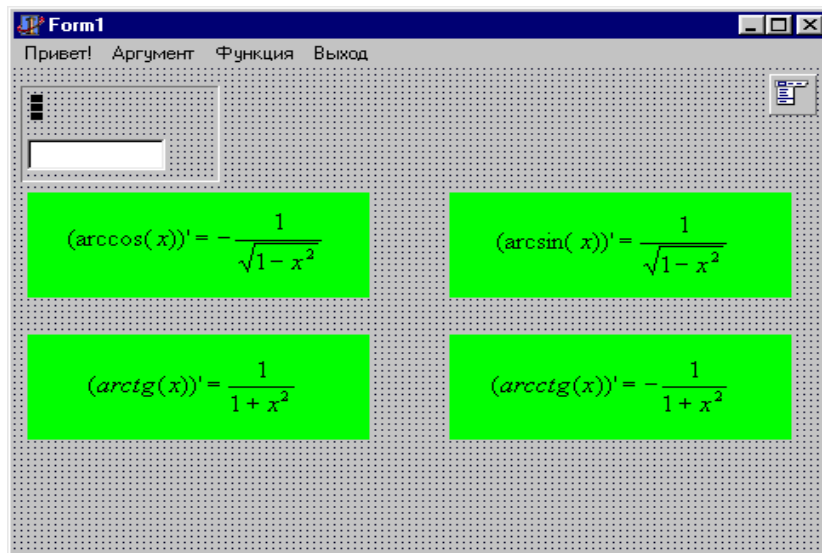


Рис 1. Расположение компонент на форме

Компоненты Bevel, Label и Edit располагаем на форме в следующей последовательности: сначала Bevel, затем Label и Edit – так как Bevel должен оказаться «снизу» Label’а и Edit’а.

Четыре компонента Image располагаем на форме произвольным образом. Свойству Picture компонента Image1 (Image1.Picture) присваиваем значение arccos.bmp (загружаем рисунок из каталога e:\5381\Урок\Menu\). Соответственно в Image2.Picture загружаем arcsin.bmp, в Image3.Picture - arctg.bmp, в Image4.Picture - arcctg.bmp.

Создание главного меню происходит в следующей последовательности:

1. Помещаем на форму компонент MainMenu, двойным щелчком на компоненте вызываем Дизайнер Меню;
2. Проектируем меню согласно рисунку 2 (вставка разделителя осуществляется вводом символа '-' в поле Caption):

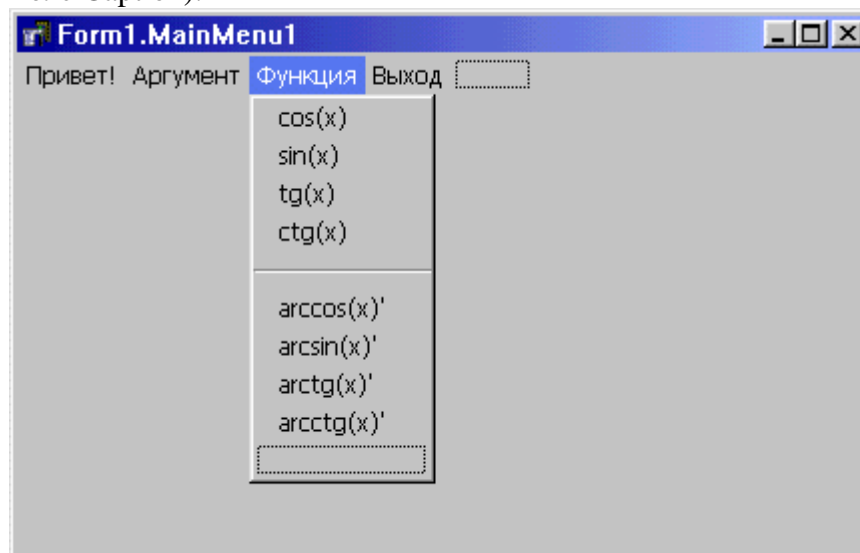


Рис 2. Проектирование главного меню программы

3. Кроме свойств Caption элементов Главного Меню, меняем свойство Name подпунктов меню «Функция» следующим образом:

Пункт подменю «Функция» (сверху вниз)	Свойство Caption	Свойство Name
Первый	Cos(x)	cosx1
Второй	Sin(x)	sinx1

Третий	$Tg(x)$	tgx1
Четвертый	$Ctg(x)$	ctgx1
Пятый – разделитель	-	Не меняем
Шестой	$Arccos(x)'$	arccosx1
Седьмой	$Arcsin(x)'$	arcsinx1
Восьмой	$Arctg(x)'$	arctgx1
Девятый	$Arcctg(x)'$	arcctg1

4. Закрываем Дизайнер Меню и теперь за каждым пунктом меню закрепляем процедуру программы. Для этого просто щелкаем мышью на нужном пункте главного меню (оно уже появилось в форме).

В программу необходимо добавить следующие процедуры (закрепить за ...):

- за пунктом меню (п. Меню) «Привет!»:

```
//выдача сообщения при выборе п.Меню "привет"
procedure TForm1.N4Click(Sender: TObject);
begin
  ShowMessage('Привет мир, Windows и пользователь!ж:)+'+#13#10+'Это мы из МГВРК!');
end;
#13#10 – символ конца строки. С его помощью осуществляется переход на другую строку в окне выдачи сообщения. ' ж:)' = –
символ "смешной рожицы с челочкой"
```

- за п. Меню «Аргумент»:

```
//выбор пункта меню "Аргумент"
procedure TForm1.N7Click(Sender: TObject);
begin
  Label1.Caption:='Введите значение x: ';
  edit1.Visible:=true;
  form1.activecontrol:=Edit1;
end;
```

- за п. Меню «Выход»:

```
//выбор п.Меню "Выход" - выход из приложения
procedure TForm1.N2Click(Sender: TObject);
begin
  close;
end;
```

- за пунктами Меню $\cos(x)$, $\sin(x)$, $tg(x)$, $ctg(x)$:

```
//обработка выборов п.п.Меню "Функция" до разделителя:
//cos(x), sin(x), tg(x), ctg(x) осуществляется ОДНОЙ процедурой
procedure TForm1.cosx1Click(Sender: TObject);
var
  cod:Integer;
  ws, resstr:String;
begin
  ws:=Edit1.text;
  val(ws,x,cod);
  try //начало блока try...except..end
  with Sender As TMenuItem do //будем обрабатывать Sender как пункт Меню
  begin
    //объект Sender вызвал эту процедуру
    if (Sender = cosx1) then //если sender – это cosx1 (т.е.cos(x)), тогда:
    begin
      str(cos(x):1:4,ws);//сразу преобразуем значение функции в строку
      Resstr:='Cos(x) = ';//и определим начало результирующей строки
    end;
    if (Sender = sinx1) then
    begin
      str(sin(x):1:4,ws);
      Resstr:='Sin(x) = ';
    end;
    if (Sender = tgx1)then
    begin
      str(tan(x):1:4,ws);
      Resstr:='tg(x) = ';
    end;
    if (Sender = ctgx1)then
    begin
      str(cotan(x):1:4,ws);
      Resstr:='Ctg(x) = ';
    end;
  end;
end;
```

```

except //если в блоке try...except возникла ошибка, она обрабатывается ЗДЕСЬ
  ShowMessage('А существует ли значение выбранной функции для Вашего аргумента?');
  exit; //а мы просто о ней сообщим
end;
ShowMessage(Resstr + ws);
end;

```

- за событием OnKeyPress элемента Edit1:

```

//выполняется при вводе символов в Edit
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  if not (key in ['0'..'9','-']) then
    key:=#0;
end;

```

- за пунктом меню arccos(x):

```

//выбор п.Меню arccos(x)
procedure TForm1.arccosx1Click(Sender: TObject);
begin //прямое преобразование Sender в элемент меню TMenuItem
  TMenuItem(Sender).checked:= not (TMenuItem(Sender).checked);
  if (TMenuItem(Sender).checked) then
    image1.visible:=true
  else
    image1.visible:=false;
end;

```

- за пунктом меню arcsin(x):

```

//выбор п.Меню arcsin(x)
procedure TForm1.arcsinx1Click(Sender: TObject);
begin
  TMenuItem(Sender).checked:= not (TMenuItem(Sender).checked);
  if (TMenuItem(Sender).checked) then
    image2.visible:=true
  else
    image2.visible:=false;
end;

```

- за пунктом меню arctg(x):

```

//выбор п.Меню arctg(x)
procedure TForm1.arctgx1Click(Sender: TObject);
begin
  TMenuItem(Sender).checked:= not (TMenuItem(Sender).checked);
  if (TMenuItem(Sender).checked) then
    image3.visible:=true
  else
    image3.visible:=false;
end;

```

- за пунктом меню arcctg(x):

```

//выбор п.Меню arcctg(x)
procedure TForm1.arcctgx1Click(Sender: TObject);
begin
  TMenuItem(Sender).checked:= not (TMenuItem(Sender).checked);
  if (TMenuItem(Sender).checked) then
    image4.visible:=true
  else
    image4.visible:=false;
end;

```

На рисунке 3 изображен вид работающей программы.

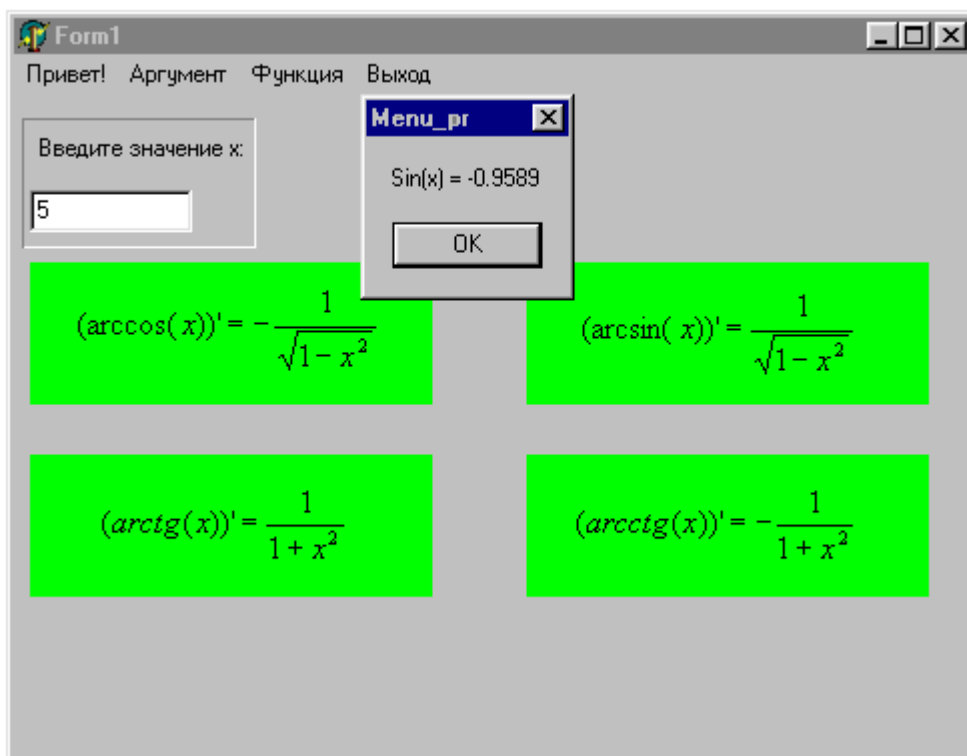


Рис. 3. Работающее программное приложение.

Индивидуальные задания.

Ниже приведено 15 вариантов задач. По указанию преподавателя выберите свое индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных. В соответствии с этим установите количество визуальных компонент в форме.

Во всех заданиях скалярные переменные вводить с помощью компонента TEdit с соответствующим пояснением в виде компонента TLabel. Скалярный результат выводить в виде компонента TLabel. Массивы представлять на форме в виде компонентов TStringGrid, в которых 0-й столбец и 0-ю строку использовать для отображения индексов массивов. Вычисления выполнять после нажатия соответствующих кнопок визуального компонента MainMenu.

1. Задана матрица размером $N \times M$. Получить массив B , присвоив его k -му элементу значение 0, если все элементы k -го столбца матрицы нулевые, и значение 1 в противном случае.
2. Задана матрица размером $N \times M$. Получить массив B , присвоив его k -му элементу значение 1, если элементы k -й строки матрицы упорядочены по убыванию, и значение 0 в противном случае.
3. Задана матрица размером $N \times M$. Получить массив B , присвоив его k -му элементу значение 1, если k -я строка матрицы симметрична, и значение 0 в противном случае.
4. Задана матрица размером $N \times M$. Определить k - количество "особых" элементов матрицы, считая элемент "особым", если он больше суммы остальных элементов своего столбца.
5. Задана матрица размером $N \times M$. Определить k - количество "особых" в матрицы, считая элемент "особым", если в его строке слева от него находятся элементы, меньшие его, а справа - большие.
6. Задана символьная матрица размером $N \times M$. Определить k - количество различных элементов матрицы (т.е. повторяющиеся элементы считать один раз).
7. Дана матрица размером $N \times M$. Упорядочить ее строки по неубыванию их первых элементов.
8. Дана матрица размером $N \times M$. Упорядочить ее строки по неубыванию суммы их элементов.
9. Дана матрица размером $N \times M$. Упорядочить ее строки по неубыванию их наибольших элементов.

10. Определить, является ли заданная квадратная матрица n -го порядка симметричной относительно побочной диагонали.
11. Для матрицы размером $N \times M$ вывести на экран все ее седловые точки. Элемент матрицы называется седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце, или наоборот.
12. В матрице n -го порядка переставить строки так, чтобы на главной диагонали матрицы были расположены элементы, наибольшие по абсолютной величине.
13. В матрице n -го порядка найти максимальный среди элементов, лежащих ниже побочной диагонали, и минимальный среди элементов, лежащих выше главной диагонали.
14. В матрице размером $N \times M$ поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением.
15. Из матрицы n -го порядка получить матрицу порядка $n-1$ путем удаления из исходной матрицы строки и столбца, на пересечении которых расположен элемент с наибольшим по модулю значением.

Лабораторная работа 4.

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИЙ OLE И COM В ВИЗУАЛЬНОМ ПРОГРАММИРОВАНИИ.

Цель работы: познакомиться на конкретных примерах с технологиями OLE и COM.

Основные понятия.

На протяжении многих лет программисты стремятся придумать способы, позволяющие использовать уже написанные коды. Например, если в программе нужна работа с таблицами, то зачем придумывать велосипед, если есть Microsoft Excel. Технология, позволяющая вызвать из вашей программы Excel или даже встроить это приложение в свою систему, называется OLE. Студентам предлагается познакомиться на лабораторном занятии с этой технологией.

Еще одним шагом в эволюции программирования стала технология COM. По сути, именно она лежит в основе технологии OLE, и именно с помощью нее реализуется сложное взаимодействие между приложениями, написанными разными программистами и на разных языках. На этом занятии предлагается рассмотреть использование COM-серверов Delphi для работы с Word и Excel.

Программа 1 (OLE технологии).

Шаг 1. Создайте новую форму.

Шаг 2. Расположите на ней компоненты: OLEContainer (закладка System) и кнопку (Button).

Шаг 3. Создайте файл Excel. Занесите в него свои ФИО. Сохраните его 'H:\1.xls'.

Шаг 4. Создайте процедуру обработки сообщения о нажатии на кнопку:

```
procedure TForm1.Button1Click(Sender: TObject); begin
OLEContainer1.CreateLinkToFile('H:\1.xls',FALSE); end;
```

Шаг 5. Скомпилируйте приложение. Нажмите на кнопку. Убедитесь, что в OLEContainer будет загружен созданный вами файл 1.xls.

Шаг 6. Щелкните 2 раза на OLEContainer, и вы увидите, что будет запущен Microsoft Excel с загруженным в него вашим файлом. Внесите в файл изменения. Сохраните файл и закройте Excel. Нажмите в своем приложении кнопку и убедитесь, что в OLEContainer отобразились изменения файла (рис. 1).

Шаг 7. Измените процедуру обработки сообщения о нажатии на кнопку следующим образом:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
OLEContainer1.CreateObjectFromFile('H:\1.xls', false);
end;
```

Шаг 8. Добавьте еще одну кнопку на форму и напишите для нее обработчик события нажатия на нее:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
OLEContainer1.Close;
end;
```

Шаг 9. Скомпилируйте проект. Нажмите на кнопку «Загрузить файл». В OLEContainer'e отобразится содержимое файла.

Щелкните 2 раза на OLEContainer. Вы получите результат, показанный на рис. 2.

Excel будет встроен в ваше приложение, и в нем будет открыт ваш файл. Для того чтобы выйти из режима редактирования, нажмите кнопку «Закрыть файл».

Шаг 10. Теперь снова поменяйте процедуру для первой кнопки:

```
procedure TForm1.Button1Click(Sender: TObject);
```

begin

```
OLEContainer1.InsertObjectDialog;
```

end;

Шаг 11. Скомпилируйте проект. Теперь при нажатии на кнопку «Загрузить файл» будет появляться стандартная форма, показанная на рис. 3. В ней показаны все программы, которые вы можете встраивать или связывать со своей программой.

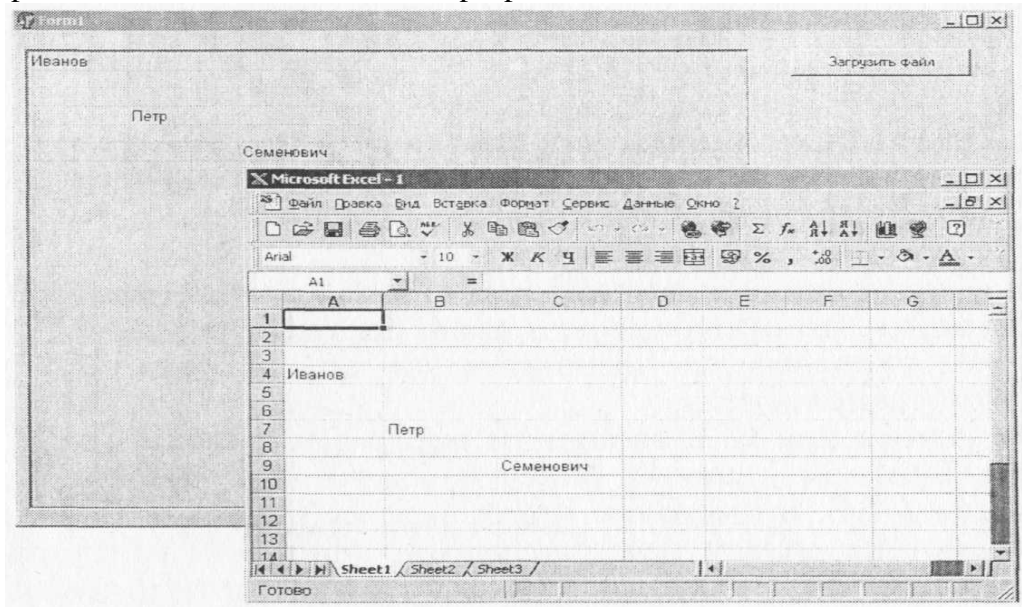


Рис.1. Иллюстрация шага 6.

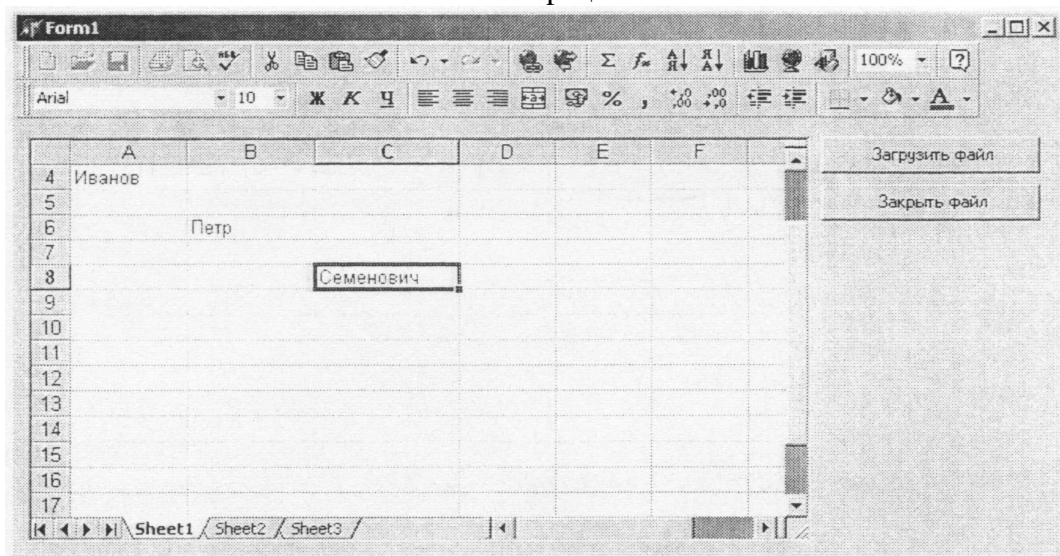


Рис.2. Иллюстрация шага 9.

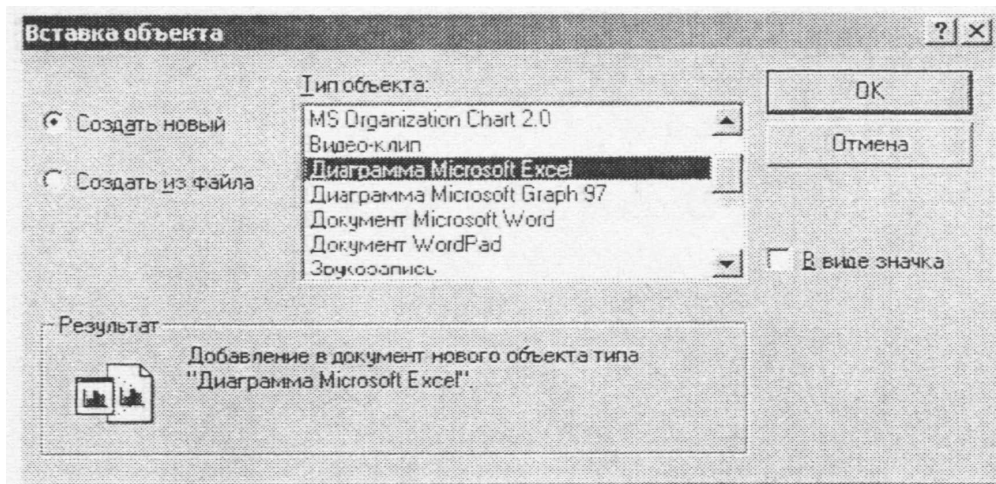


Рис.3. Иллюстрация шага 11.

Можно создать новый объект этих программ или использовать объекты из файла, также есть возможность связывать объект или внедрять в приложение (рис. 4).

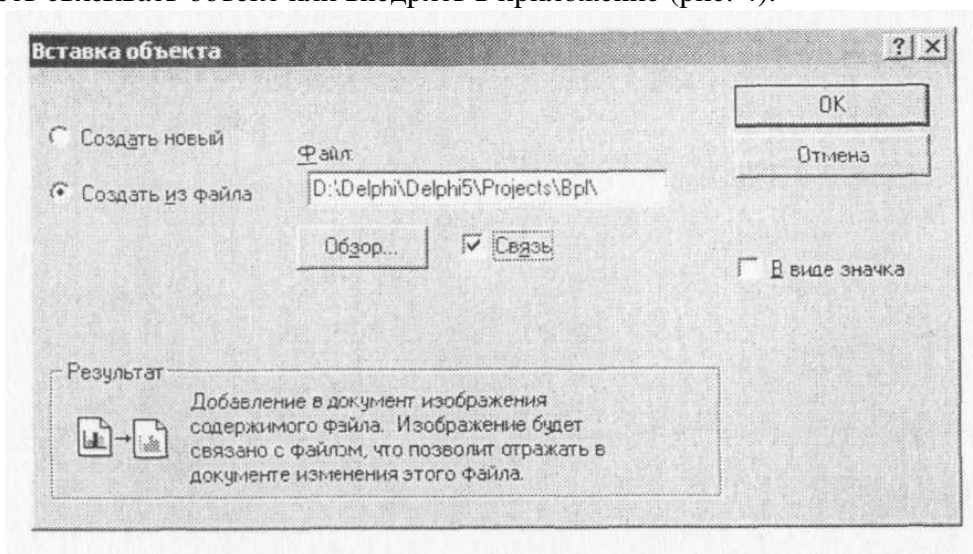


Рис. 4. Иллюстрация шага 11.

Шаг 12. Поэкспериментируйте с различными приложениями.

Программа 2 (COM сервера Delphi).

I

Рассмотрим также, как можно использовать COM-технологии для внесения изменений в документы других программ.

Следующая программа предназначена для записи данных из вашей программы в документ Excel и считывания информации из документа. 1

1. Найдите на закладке Servers компонент ExcelApplication и поместите его на форму нового приложения. Установите свойство этого компонента AutoQuit=true. 1

2. Создайте файл 'H:\2.xls\ I

3. Поместите на форму одну кнопку. Дайте ей название «Записать данные». Напишите для нее обработчик события нажатия на кнопку:

```
procedure TForm1.Button1Click(Sender: TObject);
```

1

```
var Filenamel:01eVariant;
```

```

begin
ExcelApplication1.Connect;
//Запись в существующий файл
Filename1: = 'H:\2.xls ' ;
ExcelApplication1.Workbooks.Open(Filename1, EmptyParam,
EmptyParam,EmptyParam, EmptyParam, EmptyParam,
EmptyParam, EmptyParam,EmptyParam, EmptyParam,
EmptyParam, EmptyParam, false, 0);
//Установка цвета заливки ячейки Excel
ExcelApplication1.Range[Edit2.Text,Edit2.Text].Interior .ColorIndex := 5;
//Занесение информации в определенную ячейку таблицы
ExcelApplication1.Range[Edit2.Text,Edit2.Text].Value:=
Edit1.Text;
ExcelApplication1.Disconnect;
end;

```

4. Добавьте на форму два компонента Label и Edit, как показано на рисунке.

5. Скомпилируйте проект. Внесите текст для вставки в Excel и номер ячейки (он состоит из буквы - номер столбца, и цифры - номер строки) (рис. 5).

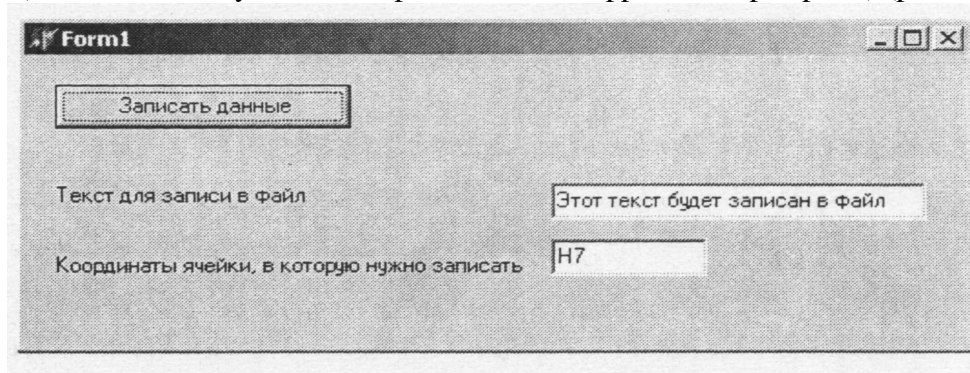


Рис.5. Иллюстрация шага 5.

После нажатия на кнопку программа спросит вас: «Сохранить изменения в файле?». Сохраните. Потом откройте файл и убедитесь, что данные записаны в нужную ячейку.

6. Добавьте в предыдущую процедуру для наглядности следующую строку:

... ..

```

ExcelApplication1.Visible[0]:=TRUE;
//Установка цвета заливки ячейки Excel

```

.....

7. Скомпилируйте проект еще раз. Теперь Excel будет запущен не в фоновом режиме, как это было ранее.

Примечание. При программировании работы с Excel и Word рекомендуем написать макрос в самом Excel или Word, а потом перенести код макроса в свою программу с учетом небольших особенностей синтаксиса выбранного языка.

8. Измените название элементов на форме.

9. Измените процедуру нажатия на кнопку следующим образом:

```

procedure TFcml.Button1Click(Sender: TObject);
var FileNamel: OleVariant;
begin
ExcelApplication1.Connect;
Filename1: = 'D:\2.xls';
ExcelApplication1.Workbooks.Open(Filename1, EmptyParam,

```

```

EmptyParam, EmptyParam, EmptyParam, EmptyParam,
EmptyParam,
EmptyParam, EmptyParam, EmptyParam, EmptyParam,
EmptyParam, false, 0);
Edit1.Text:=ExcelApplication1.Range[Edit2.Text, EmptyParam].Text;
ExcelApplication1.Disconnect; end;

```

10. Создайте программу для записи информации в документ Word. Для этого измените форму проекта, как показано на рис. 6 (добавьте компоненты WordApplication и WordFont) и перепишите обработчик нажатия на кнопку:

```

procedure TForm1.Button1Click(Sender: TObject);
begin
WordApplication1.Connect;
//создание нового документа
WordApplication1.Documents.Add(EmptyParam,EmptyParam);
WordApplication1.Visible:=true;
//Установка шрифта
WordFont1.ConnectTo (WcrdApplication1. Selection . Font) ;
WordFont1.Bold:=3;
WordFont1.Size:=17;
//Вставка текста
WordApplication1.Selection.InsertAfter(Edit1.Text);
WordApplication1.Disconnect;
end;

```

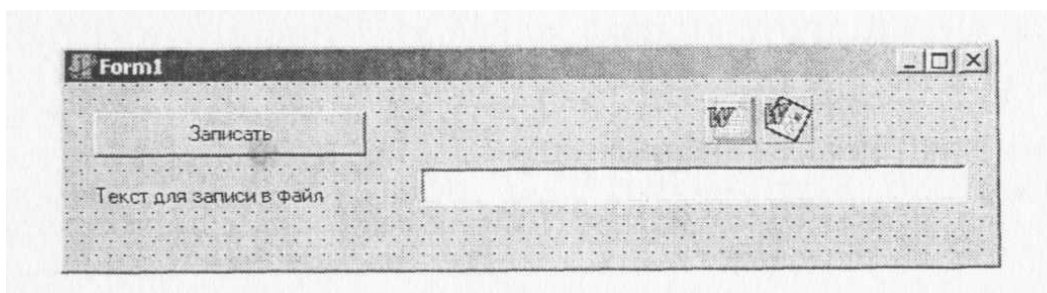


Рис.6. Форма с компонентами WordApplication и WordFont.

Индивидуальные задания.

Ниже приведено 15 вариантов задач. По указанию преподавателя выберите свое индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных.

Во всех заданиях скалярные переменные вводить с помощью компонента TEdit с соответствующим пояснением в виде компонента TLabel. Скалярный результат выводить в виде компонента ExcelApplication в файл табличного процессора Excel или WordApplication в файл Word.

1. Дан массив из k символов. Вывести на в файл табличного процессора сначала все цифры, входящие в него, а затем все остальные символы, сохраняя при этом взаимное расположение символов в каждой из этих двух групп.
2. Дан массив, содержащий от 1 до k символов, за которым следует точка. Напечатать этот текст в файл Word в обратном порядке.
3. Дан непустой массив из цифр. Вывести в файл Excel цифру, наиболее часто встречающуюся в этом массиве.
4. Отсортировать элементы массива X по возрастанию. Результат напечатать в файл Excel.

5. Элементы массива X расположить в обратном порядке. Результат напечатать в файл Excel.
6. Элементы массива X циклически сдвинуть на k позиций влево. Результат напечатать в файл Excel.
7. Элементы массива X циклически сдвинуть на k позиций вправо. Результат напечатать в файл Excel.
8. Преобразовать массив X по следующему правилу: все отрицательные элементы массива перенести в начало, а все остальные - в конец, сохраняя исходное взаимное расположение как среди отрицательных, так и среди остальных элементов. Результат напечатать в файл Excel.
9. Элементы каждого из массивов X и Y упорядочены по неубыванию. Объединить элементы этих двух массивов в один массив Z так, чтобы они снова оказались упорядоченными по неубыванию. Результат напечатать в файл Excel.
10. Дан массив из 4 символов. Определить, симметричен ли он, т.е. читается ли он одинаково слева направо и справа налево. Если симметричен, то вывести его в файл Word.
11. Дано два массива. Найти наименьшее среди тех элементов первого массива, которые не входят во второй массив. Напечатать эти наименьшие числа в файл Word.
12. Определить количество инверсий в этом массиве X (т.е. таких пар элементов, в которых большее число находится слева от меньшего: $x\{i\} > x\{j\}$ при $i < j$). Напечатать эти пары в файл.
13. Дан массив из строчных латинских букв. Вывести в файл Word этот массив в алфавитном порядке все буквы, которые входят в этот текст по одному разу.
14. Вывести в файл Excel заданный массив из k символов, удалив из него повторные вхождения каждого символа.
15. Определить, сколько различных символов входит в заданный текст (текст вводится в файл Word и открывается посредством соответствующей компоненты), содержащий не более k символов и оканчивающийся точкой (в сам текст точка не входит).

Лабораторная работа 5. ЗНАКОМСТВО СО СРЕДОЙ ПРОЕКТИРОВАНИЯ ПРОГРАММНЫХ ПРОДУКТОВ VISUAL C++.

Цель работы: познакомиться с принципами объектно-ориентированного программирования в среде Visual C++.

Задание:

- 1) Создать проект программы
- 2) Визуально спроектировать диалоговую панель
- 3) Связать элементы управления с событиями
- 4) Сохранить и выполнить программу

Порядок выполнения работы.

Теперь рассмотрим все пункты по порядку.

1) Вначале запустим Visual C++. Зайдем в меню FILE -> NEW..., выберем там Project. Ваша панель должна выглядеть теперь так:

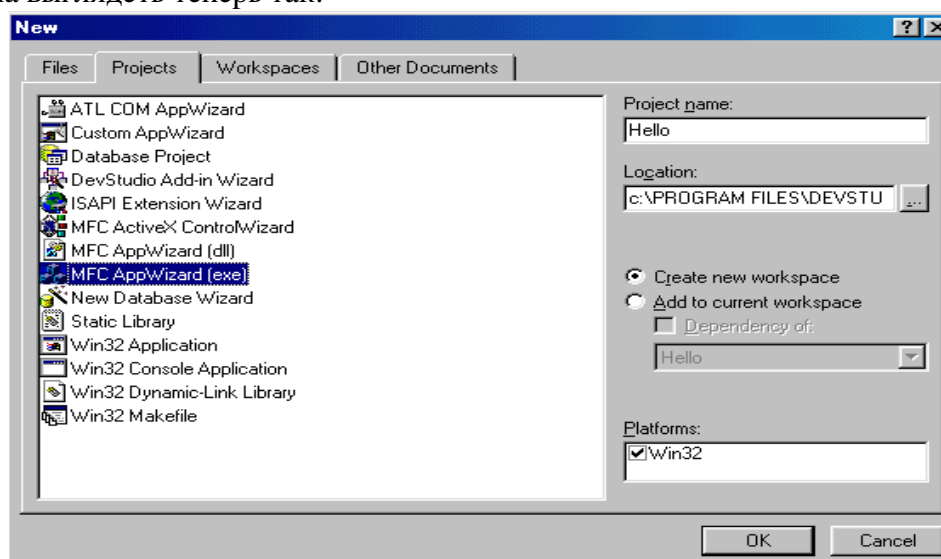


Рис.1.

Затем выбираем MFC AppWizard(exe). В окне редактирования Project Name задаем имя программы и нажимаем ОК.

Шаг 1. Теперь выбираем Dialog based, как показано на рис 2. Вы задали компилятору, что программа ваша будет основана на диалоговых окнах.

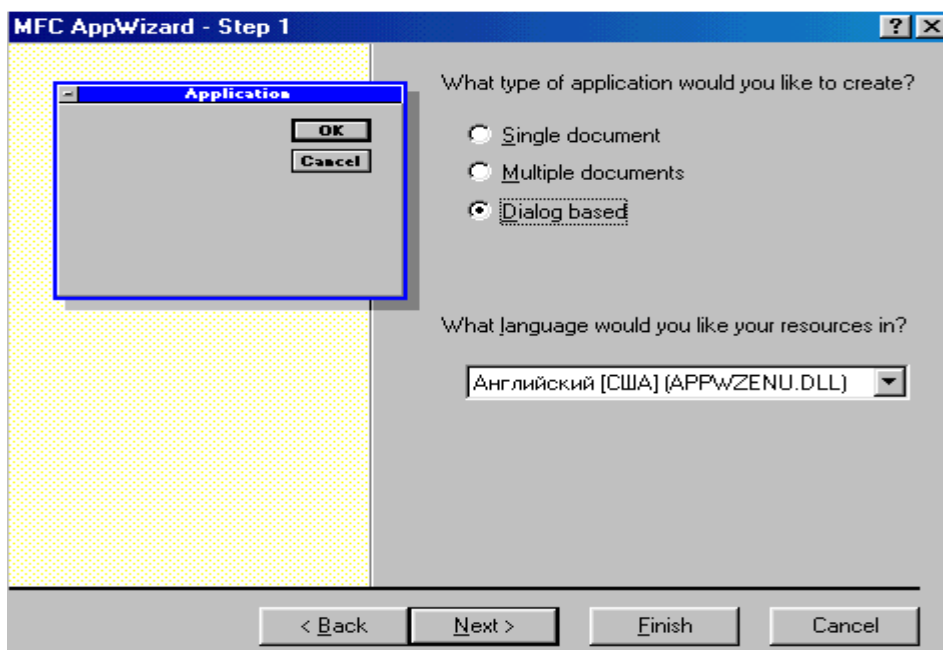


Рис. 2.

Шаг 2. В этой панели введите название программы как показано на рис 3 . Рассмотрим здесь группу "What features would you like to include".

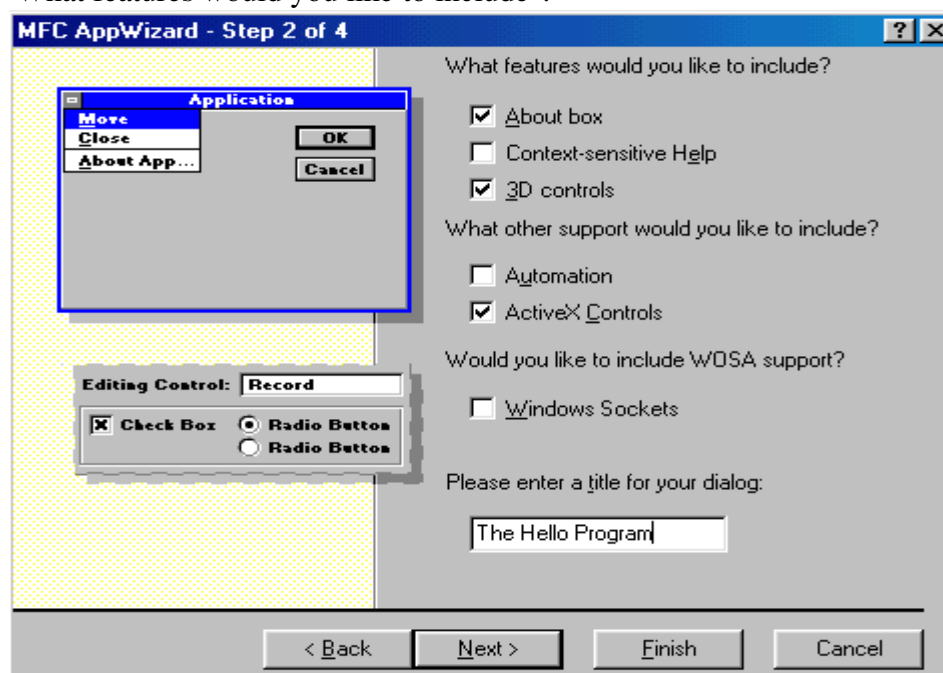


Рис. 3.

Первое включение говорит о использовании справки (небольшой диалоговой панели, которая всплывает при нажатии на иконку).

Теперь щелкаем NEXT.

Шаг 3. Вы видите, что диалоговая панель задает два вопроса:

- 1) Нужно ли вставлять комментарии в исходный текст, который будет являться каркасом вашей программы?
 - 2) Программа, которую создает для вас MFC AppWizard, будет иметь библиотеку с динамической компоновкой (DLL), а не статической? DLL дает преимущество в том, что EXE файл будет меньшего размера, а рядом к нему будет прилагаться DLL файл.
- Выбираем всё как показано на рис 4.

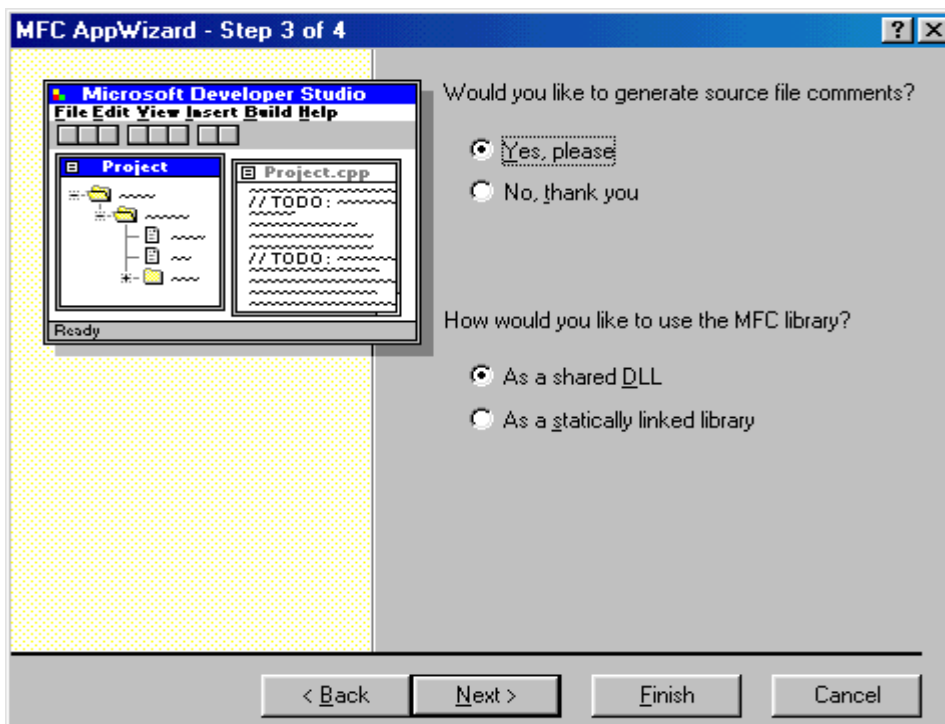


Рис. 4.

Затем нажимаем NEXT и щелкаем FINISH, посмотрите, чтобы было выбрано все, как показано на рис. 5.

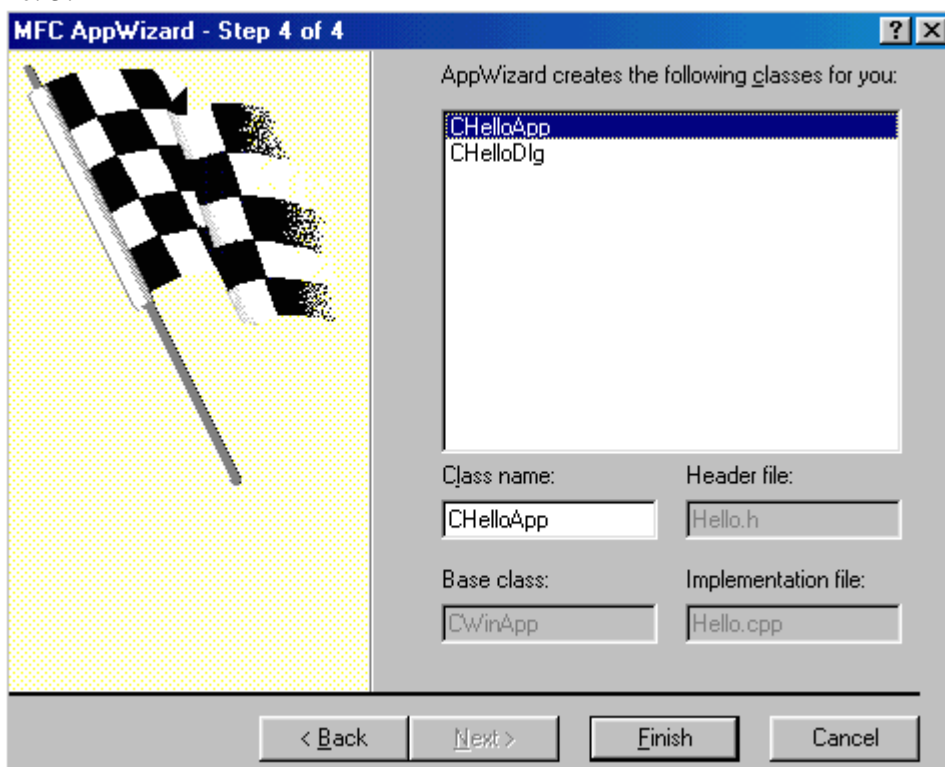


Рис.5.

Вы закончили создание проекта программы!

Все каркасные файлы для вашей программы написал Visual C++, с помощью мастера настроек MFC AppWizard. MFC - это вещь очень полезная, с помощью нее можно быстро создавать программы, так как все дежурные файлы он пишет сам, что сильно облегчает работу программиста, ведь не писать стандартный набор текста каждый раз!

А теперь мы спроектируем диалоговую панель.

Сейчас мы визуально спроектируем диалоговую панель. Для этого вначале выберите закладку "Resource View" и раскройте пункт Hello Resource, как показано на рис. 6.

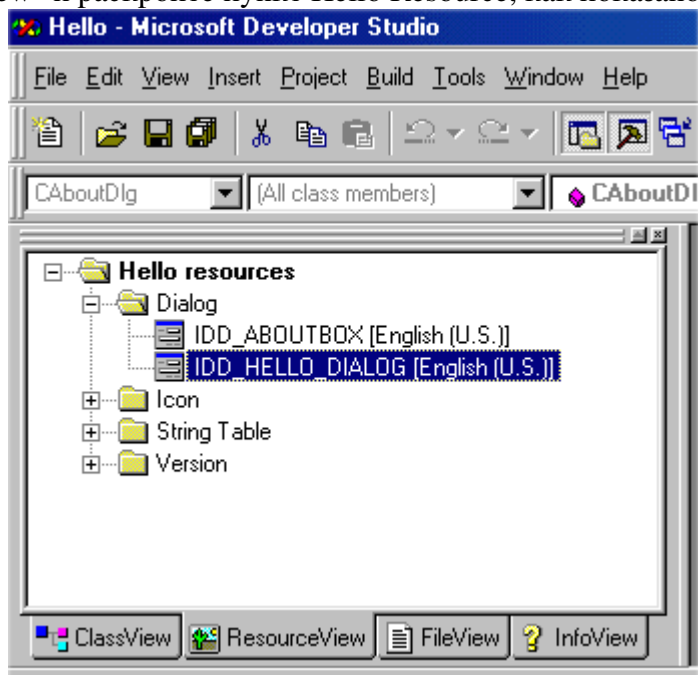


Рис.6.

Здесь вы видите две записи:

- 1) IDD_ABOUTBOX
- 2) IDD_HELLO_DIALOG

1) - это название диалоговой панели ABOUT, а 2)- название главной диалоговой панели.

Выполните щелчок по второй строчке(2) и справа появится, диалоговая панель. В этом режиме можно ее редактировать (см. рис. 7).

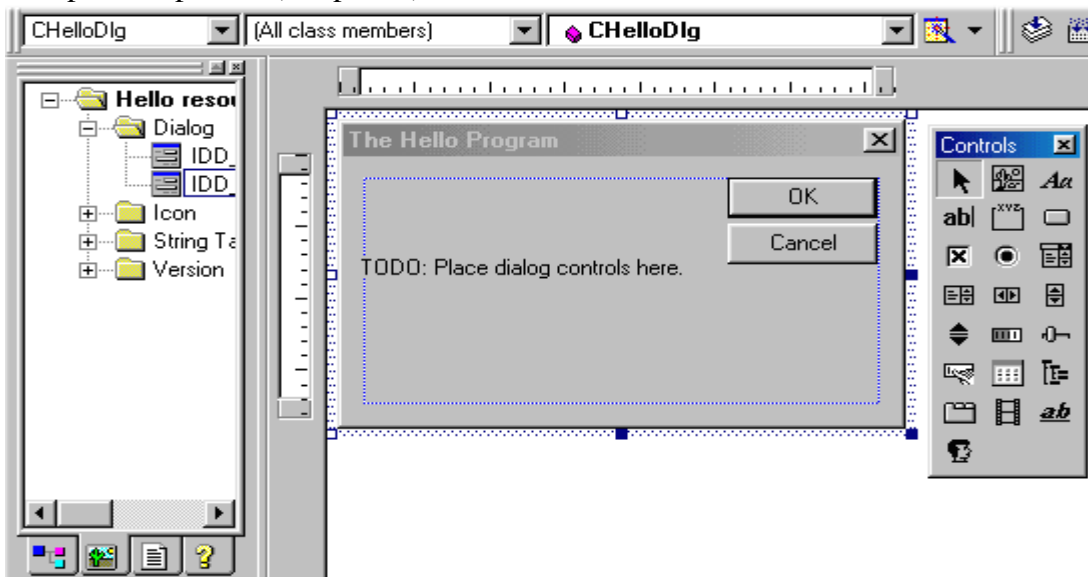


Рис.7.

Удалите из диалоговой панели текст: "TODO:.." и две кнопки, для этого щелкните на эти элементы и нажмите DEL. Теперь у вас чистая панель и вы можете спроектировать ее на свой вкус.

Справа от диалоговой панели находятся элементы управления(если вы их не видите, то зайдите в TOOLS->CUSTOMIZE, в закладке TOOLBARS, CONTROLS - поставьте флажок). Вы можете редактировать панель в полном экране, для этого зайдите в VIEW->FULLSCREEN.

Сейчас выберите в элементах управления Button,



и щелкните мышкой на диалоговую панель, которая должна выглядеть, как показано на рис. 8.

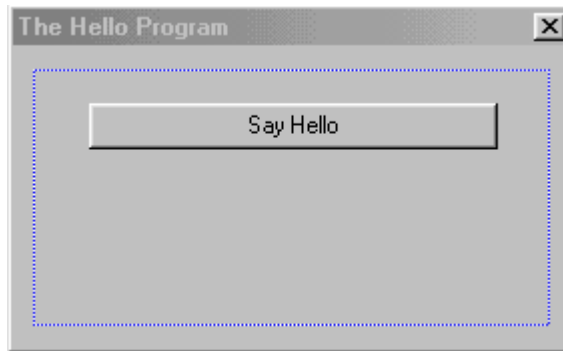
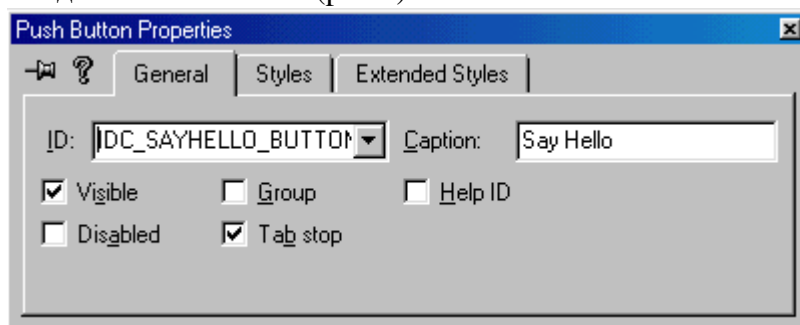


Рис.7.

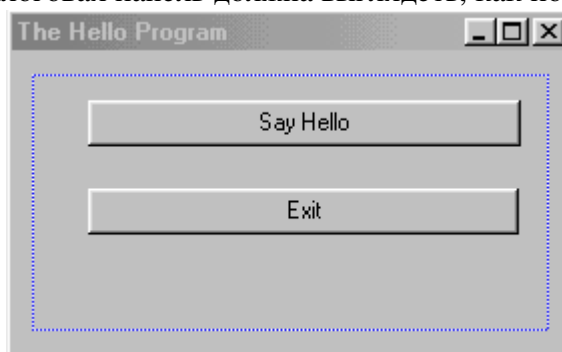
Теперь выделите кнопку мышкой, для этого нажмите на нее и отбуксируйте маркеры, чтобы кнопка стала такой же, как на рис 6. После этого опять выделите кнопку левым щелчком мыши, затем нажмите правую кнопку и в раскрывшемся меню выберите Properties(настройки).

Выведется такая диалоговая панель (рис 7).



Вместо текста IDC_BUTTON1 введите IDC_SAYHELLO_BUTTON, зададим идентификатор, по которому VC будет распознавать элемент. В строке Captions(надпись), введите название кнопки Say Hello.

Вставьте теперь еще одну кнопку такого же размера, с ID - IDC_EXIT_BUTTON и Captions(надпись) Exit. Диалоговая панель должна выглядеть, как показано на рис. 1.8



Все свойства диалогового окна будут задавать таблицей, где будут указываться свойства элементов диалога. Свяжем элементы управления с событиями. Два элемента - это две кнопки: Say Hello и Exit. Теперь воспользуемся еще одним мастером ClassWizard, он сильно облегчит нам работу. Зайдите в VIEW->ClassWizard, появится такая диалоговая панель

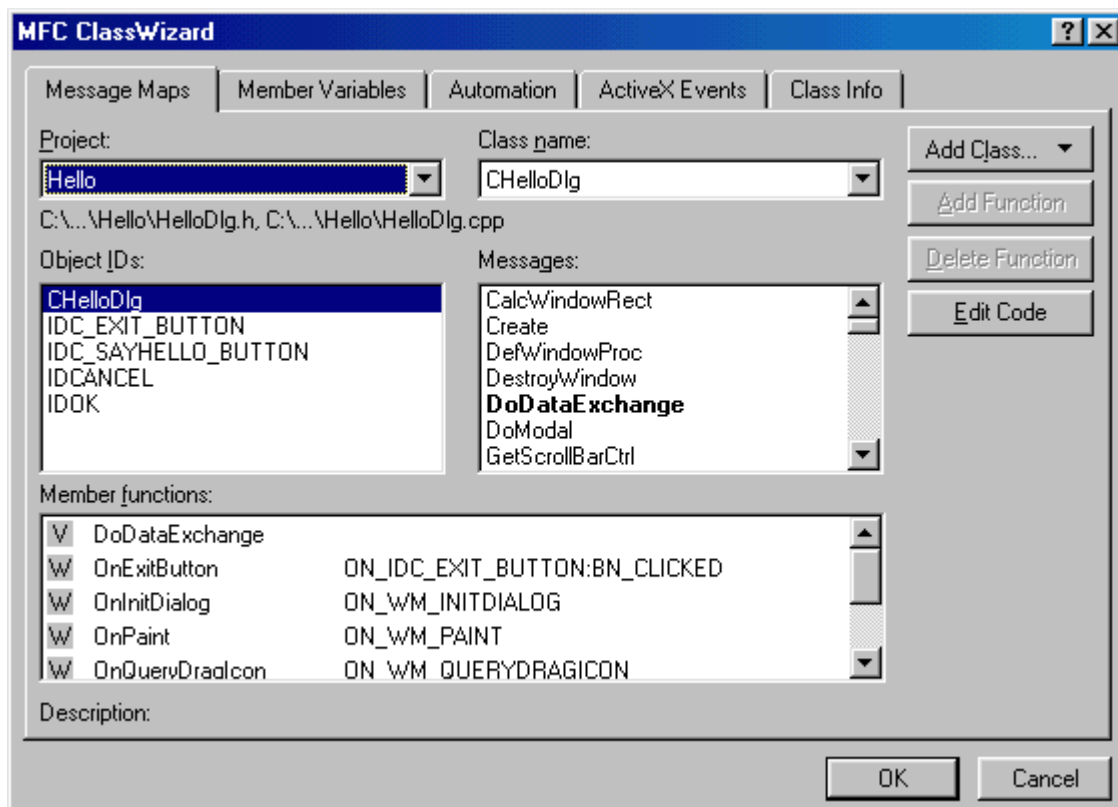


Рис.1.8.

Рассмотрим эту панель.

1) Project- здесь вы выбираете свой проект.

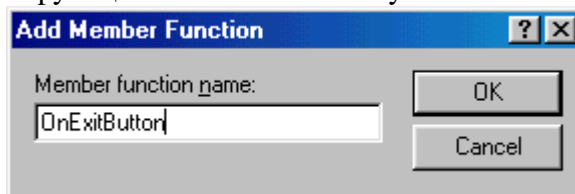
2) ClassName - название класса, элементы которого вы будете связывать с событиями, проверьте, чтобы там было установлено CHelloDlg - это класс, который связан с главным диалоговым окном.

3) Object IDs - это идентификаторы(названия) элементов управления, в нашем случае это две кнопки. Первый идентификатор - это название самого класса, оно туда включено, так как с ним связаны многие события(это такие значения, которые возвращаются Windows, при совершении какого-нибудь действия).

4) Messages - а это как раз те самые события, которые генерирует программа. К примеру, если вы нажали на кнопку, то генерируется событие BN_CLICKED, а если два раза - то BN_DOUBLECLICKED.

5) Members function - это список функций, которые вы включили, и соответствующие им события.

Теперь свяжем кнопку EXIT с событием BN_CLICKED - оно возникает при нажатие на кнопку. Для этого щелкнем в Object IDs на IDC_EXIT_BUTTON, справа появится события, с которыми можно связать эту кнопку. Выбираем там BN_CLICKED. И щелкаем на кнопке Add Function , которая находится справа. Этим нажатием вы говорите Visual C++, что вам необходимо связать это событие с функцией. Выведется такая диалоговая панель, где вам предложат выбрать название функции. Оставьте все по умолчанию и нажмите кнопку ОК.



В Members function появилась запись о том, что связали кнопку (с идентификатором IDC_EXIT_BUTTON) с функцией OnExitButton. Теперь щелкаем по кнопке Edit Code, которая

находится ниже кнопки Add Function. При нажатии на эту кнопку мы переходим в режим редактирования исходного текста. В окошке появится текст:

```
void CHelloDlg::OnExitButton()
{
// TODO: Add your control notification handler code here
}
```

Комментарий // TODO: Add your control notification handler code here говорит, что после него можно ставить свой код.

Начало вашего кода будет обозначаться

```
///Здесь начинается ваш код///
```

А конец вашего кода -

```
///Здесь заканчивается ваш код///
```

Теперь введем код, для этого вам надо переписать его

```
void CHelloDlg::OnExitButton()
{
// TODO: Add your control notification handler code here
///Здесь начинается ваш код///
OnOK();
///Здесь заканчивается ваш код///
}
```

Напишем OnOK();, эта функция будет выполняться при каждом нажатии на кнопку EXIT, и она предназначена для завершения программы. Проверьте, чтобы в написании OnOK, OK было написано с большой буквы, иначе, при компиляции программы, Visual C++ сообщит вам об ошибке, так символы верхнего и нижнего регистра в VC имеют распознаются по-разному.

Итак вы связали кнопку EXIT с событием BN_CLICKED, а его с функцией OnExitButton, которая будет выполняться при каждом нажатии на кнопку EXIT.

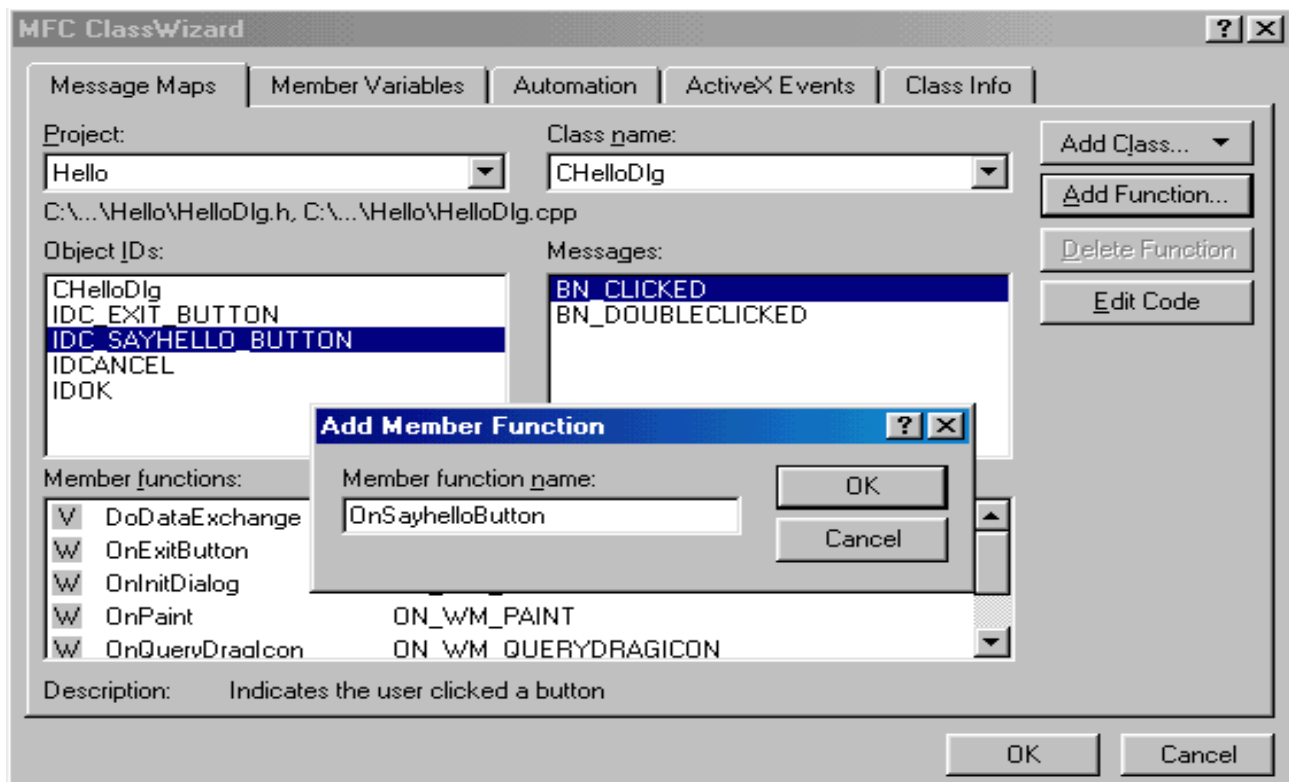
Теперь свяжем кнопку Say Hello с событием BN_CLICKED. Для этого зайдём в меню VIEW->ClassWizard.

Выберем в Object IDs IDC_SAYHELLO_BUTTON.

Щёлкнем на BN_CLICKED в Messages (этим связываем кнопку Say Hello с событием BN_CLICKED).

Нажимаем на Add function, этим связываем событие BN_CLICKED с функцией.

В раскрывшемся окне оставляем все по умолчанию и нажимаем кнопку ОК.



Затем нажимаем на кнопку Edit Code, для редактирование исходного текста.

И в раскрывшемся окне пишем следующий код:

```
void CHelloDlg::OnSayhelloButton()
{
// TODO: Add your control notification handler code here
///Здесь начинается ваш код///
MessageBox("Say Hello");
///Здесь заканчивается ваш код///
}
```

Функция `MessageBox("Say Hello");` вызывает окно сообщений, в котором написан текст Say Hello, который и является параметром этой функции.

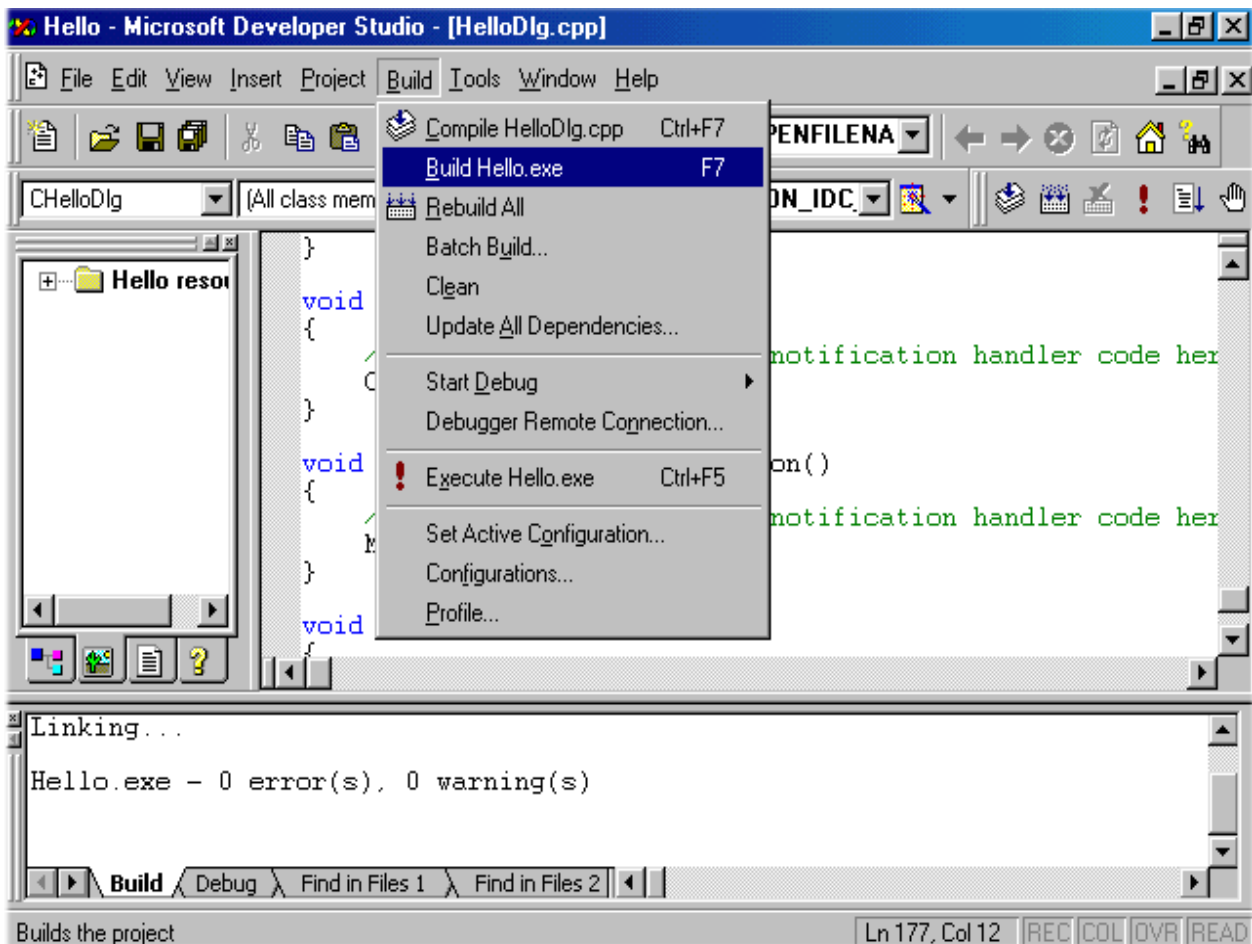
Итак, связали кнопки Say Hello и EXIT с событием BN_CLICKED. При нажатии на кнопку Exit программа завершается, а при нажатии на кнопку Say Hello выводится сообщение "Say Hello".

Теперь осталось построить и выполнить программу. Нажимаем Далее...

Компиляция и запуск программы

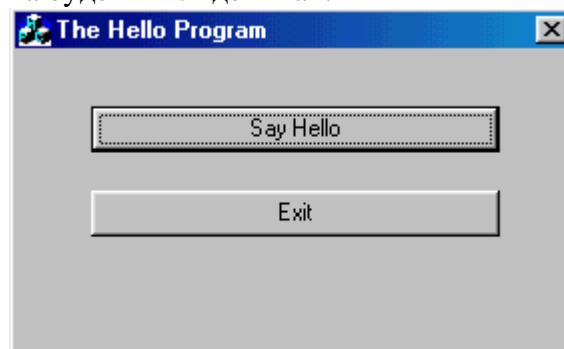
Для того, чтобы выполнить программу надо ее вначале сохранить. Зайдите в меню File и там нажмите на Save All. Затем надо ее построить (скомпилировать).

Зайдем в меню Build->Build Hello.exe, начнется построение программы. Если компилятор не выдал сообщение об ошибке, которые можно посмотреть в появившемся внизу окошке, то можно смело запускать программу.



Теперь осталось только выполнить программу, для этого зайдите в меню Build->Execute Hello.exe

Ваша программа должна будет выглядеть так:



А окно сообщений при нажатии на кнопку Say Hello:



Проект программы создан, визуально спроектирована диалоговая панель, элементы управления связаны с событиями с помощью мастера ClassWizard, программа выполняется.

Индивидуальные задания.

Ниже приведено 15 вариантов задач. По указанию преподавателя выберите свое индивидуальное задание. Уточните условие задания, количество, наименование, типы исходных данных. В соответствии с

этим установите количество окон, тексты заголовков на форме, размеры шрифтов, а также типы переменных и функции преобразования при вводе и выводе результатов.

С помощью инспектора объектов измените цвет формы, шрифт выводимых символов.

Задание: составить программу вычисления арифметического выражения по вариантам:

№ варианта	Программируемая формула	A	B	C	D	Результат
1	$\sqrt{\frac{A}{\pi(B - \sqrt{B^2 - C^2})}} + \sin D$	10^5	5	2	2.5	1.95862 E+2
2	$\sqrt{\frac{A}{(2D * \ln \frac{C}{B})}} - \frac{3.5}{C} \ln D $	10^4	10	0.1	-3	-1.48774 E+1
3	$\sqrt{\frac{A}{\left(\pi D \left(\ln \frac{D}{B} - 1.7\Delta\right)\right)}} - \pi\Delta$	10^4	10	0.2	3	1.79615 E+1
4	$\sqrt{\frac{A \cdot (\pi D + \pi B + 1 \cdot C)}{\pi D}}$	10^{-2}	-1.5	4.1	-3	1.61778 E-2
5	$\sqrt{\frac{\ln A \cdot (B + C)}{4D \cdot A(B - C)}} - \exp\left(\frac{\sin B}{\cos C}\right)$	10^1	-1.7	3.9	-3	-3.83304 E+0
6	$\left(\frac{A}{B(C + D) \ln\left(\frac{A(C + D)}{C - D}\right)}\right)^{\pi\Delta}$	10^3	3.5	4.1	-3	1.06442 E+1
7	$\sqrt{A \cdot B \left(\frac{1 + C \cdot \exp\left(\frac{D}{A \cdot B}\right)}{\pi D}\right)}$	10^1	-0.5	1.1	-1	9.65643 E-1
8	$1.7\Delta \left(\frac{\Delta \cdot \pi A}{\exp\left(\frac{B(C + 1.7\Delta)}{\Delta}\right)} - D\right)$	10^2	-20.5	5.1	-1.5	1.36556 E+3
9	$1.7\Delta \left(\frac{\pi D + A}{\exp\left(\frac{B\sqrt{C}}{\pi}\right)} - 1.7\Delta\right)$	10^{-1}	2.5	5.1	-1.5	-5.55037 E-1

10	$\frac{\sin A}{e - \exp B} - \sqrt{C \ln C} - \pi D$	10^{-1}	1.2	5.1	2.05	1.42678 E+3
11	$\frac{\sin(\ln A) + \cos(\lg B)}{\exp(\sqrt{C})}$	10^3	12	7.2 1	-	2.79759 E-1
12	$\sqrt{\frac{10A + 2\pi \cdot D}{1.1 + \sqrt{\frac{13B}{0.5 + \sqrt{\cos C}}}}}$	10	1.3	0.1	-0.05	4.66048 E+0
13	$7.7 \cdot 10^{-5} \cdot \sqrt{5.5^A - \frac{2 \sin(A+B)}{1 - \cos(\ln C)} + \frac{\pi}{D}}$	10^{-2}	1.3 9	3.1	0.55	1.39860 E-4
14	$\sqrt{A + \sqrt{B + \sin\left(\frac{C}{f}\right)}} - \sqrt{B \cdot \cos\left(\frac{D}{f}\right) + \sqrt{A}}$	10^{-3}	21. 39	23. 1	-0.1 2	-4.73017 E+0
15	$\frac{A^B + B^A \ln C - C \cdot \lg \sqrt{A}}{2B + D}$	10^{-1}	2.1	0.1	-3.1 2	-2.24257 E+0