Дисциплина: **Информатика и программирование**

специальность: **информатики**

курс, группа: **4 к 1,2 г**

форма обучения: **заочная**

**7 семестр (сессия)**

период проведения занятий: **20,26 октября 2020 г.**

вид занятий, кол-во часов: Консультации 4 часа, экзамен

**-20.10.2020 г. Консультация 2 часа,**

**-26.10.2020 г. Экзамен (Конс. 1,6+Экз. 0,4 часа)**

преподаватель: **Астахов В.К.**

электронная почта преподавателя: **vadast@mail.ru**

**Литература:**

## 1. Основная литература:

1. Давыдов И.С. Информатика [Электронный ресурс]: учебное пособие / И.С. Давыдов. — Электрон. текстовые данные. — СПб.: Проспект Науки, 2016. — 480 c. — 978-5-903090-19-8. — Режим доступа: http://www.iprbookshop.ru/35850.html (гриф)
2. Задохина Н.В. Математика и информатика. Решение логико-познавательных задач [Электронный ресурс]: учебное пособие для студентов вузов/ Задохина Н.В.— Электрон. текстовые данные.—М.: ЮНИТИ-ДАНА, 2015.— 127 c.— Режим доступа: http://www.iprbookshop.ru/34474.— ЭБС «IPRbooks», по паролю (гриф УМЦ, НИИ)
3. Тарасов, В. Н. Математическое программирование. Теория, алгоритмы, программы: учебное пособие / В. Н. Тарасов, Н. Ф. Бахарева. — Самара : Поволжский государственный университет телекоммуникаций и информатики, 2017. — 222 c. — ISBN 5-7410-0559-4. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/73832.html (дата обращения: 19.11.2019). — Режим доступа: для авторизир. пользователе

## 2. Дополнительная литература:

1. Алексеев А.П. Информатика 2015 [Электронный ресурс]: учебное пособие / А.П. Алексеев. — Электрон. текстовые данные. — М.: СОЛОН-ПРЕСС, 2015. — 400 c. — 978-5-91359-158-6. — Режим доступа: http://www.iprbookshop.ru/53821.html
2. Петров, В. Ю. Информатика. Алгоритмизация и программирование. Часть 1 : учебное пособие / В. Ю. Петров. — СПб. : Университет ИТМО, 2016. — 93 c. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/66473.html (дата обращения: 19.11.2019). — Режим доступа: для авторизир. пользователей
3. Роганов, Е. А. Основы информатики и программирования [Электронный ресурс] / Е. А. Роганов. — 2-е изд. — Электрон. текстовые данные. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 392 c. — 2227-8397. — Режим доступа: <http://www.iprbookshop.ru/73689.html>
4. Электронный конспект лекций по дисциплине «Информатика»

**Интернет- ресурсы**

1. Видео лекции [Электронный ресурс]: раздел Информатика. — Электрон. дан. — Режим доступа: http://www.4ege.ru/video-informatika/2502-video-lekcii-po-informatike.html
2. Видео лекции по теме «Информатика» для учителей, методистов, преподавателей, учащихся [Электронный ресурс]: раздел Информатика. — Электрон. дан. — Режим доступа: http://www. metodist.lbz.ru
3. Видео лекции учебных курсов Физтеха для студентов МФТИ и не только [Электронный ресурс]: раздел Информатика. — Электрон. дан. — Режим доступа: http://www. lectoriy. mipt.ru/
4. Видео лекции, видео уроки онлайн по информатике [Электронный ресурс]: раздел Информатика. — Электрон. дан. — Режим доступа: http://www. botaniks.ru/ videouroki.php
5. Видео лекции, новые видео лекции по информатике [Электронный ресурс]: раздел Информатика. — Электрон. дан. — Режим доступа: http://www.hi-edu.ru/blog/?p=229
6. Видео лекции, операционные системы [Электронный ресурс]: раздел Информатика. — Электрон. дан. — Режим доступа: http://www.edu.jobsmarket.ru/library/ituser/ 15249/
7. Видеоуроки по Delphi 7 (видеокурсы) [Электронный ресурс]: офиц. сайт — Электрон. дан. — Режим доступа: http://www.videouchenik.ru›comp…po-delphi-7-2006…i-video.html
8. Научная электронная библиотека eLIBRARY.ru [Электронный ресурс]: раздел Информационные технологии. — Электрон. дан. — Режим доступа: http://www.elibrary.ru/ defaultx.asp
9. Научная электронная библиотека IPRbooks.ru [Электронный ресурс]: раздел Информационные технологии. — Электрон. дан. — Режим доступа: http://www. iprbooks.ru (по паролю)
10. Научная электронная онлайн-библиотека Порталус [Электронный ресурс]: раздел Информационные технологии. — Электрон. дан. — Режим доступа: http://www. portalus.ru
11. Пособие по Информатике [Электронный ресурс]: раздел Информатика. — Электрон. дан. — Режим доступа: http://www. www.alleng.ru/edu/comp4.htm

**Задания по темам и датам**

| **дата, время** **занятия** | **тема** **занятия** | **кол-во часов** | **вопросы для изучения и обсуждения** | **литература** | **контрольные вопросы, задания** |
| --- | --- | --- | --- | --- | --- |
| 20.10.202011.10-12.05 | Консультация | 2 | 1.Технологии работы с текстом, таблицами, базами данных, создание презентаций. Офисные пакеты MS Office2. Компьютерные сети, защита информации в ПК | 1.ЭКЛ «Информатика», с.83-1022.Осн. и доп. литература, интернет-ресурсы | 1.Изучить рекомендованную литературу. (присылать материалы преподавателю для проверки НЕ НУЖНО) |
| 30.10.202013.00-14.30 | Консультация, | 1,6 | 3.Основы алгоритмизации и программирования | 1.ЭКЛ «Информатика», с.103-1232.Осн. и доп. литература, интернет-ресурсы | 1.Изучить рекомендованную литературу. (присылать материалы преподавателю для проверки НЕ НУЖНО) |
| 30.10.202013.00-14.30 | Экзамен | 0,4  | Перечень вопросов к экзамену по дисциплине «Информатика и программирование» |  | Перечень вопросов к экзамену по дисциплине находится на сайте во вкладке: Студенту. Далее: Вопросы к зачетам и экзаменам-Прикладная информатика (бакалавриат)-Полный перечень вопросов и заданий (Cкачать. ZIP). Затем в скачанном списке ищем дисциплину «Информатика и программирование», открываем файл в word. |

**Рекомендации и требования к выполнению указанных заданий**

Изучить теоретический материал и подготовиться к ответам на вопросы, представленными в Электронном конспекте лекций (ЭКЛ) по дисциплине «Информатика». Дополнительно отчитываться по контрольным вопросам и присылать материал преподавателю не надо.

**Краткий материал по вопросу**

**«Основы алгоритмизации и программирования»**

### *Основы объектно-ориентированного программирования*

В процессе развития программирования многие стали приходить к выводу, что структурный подход в программировании обладает многими недостатками: как бы эффективно не применялся структурный подход, он не позволяет в достаточной мере упростить большие сложные программы.

Вторая, более важная, проблема структурного подхода заключается в том, что отделение данных от функций оказывается малопригодным для отображения картины реального мира. В реальном мире нам приходится иметь дело с физическими объектами, такими, например, как люди или машины. Эти объекты нельзя отнести ни к данным, ни к функциям, поскольку реальные вещи представляют собой совокупность *свойств* и *поведения*.

Примерами свойств для людей могут являться цвет глаз, рост, вес, место работы; для машин – мощность двигателя, количество дверей и т.д. Таким образом, свойства объектов равносильны данным в программах: они имеют определенное значение.

Поведение – это некоторая реакция объекта в ответ на внешнее воздействие. Например, если вы нажмете на тормоз автомобиля, это повлечет за собой его остановку.

Основополагающей идеей объектно-ориентированного подхода является объединение *данных* и *действий*, *производимых над этими данными*, в единое целое, которое называется объектом.

Функции объекта (его действия), называемые *методами* или *функциями-членами*, обычно предназначены для доступа к данным объекта. Если необходимо считать какие-то данные объекта, нужно вызвать соответствующий метод, который выполнит считывание и возвратит нужное значение. Прямой доступ к данным невозможен. Данные сокрыты от внешнего воздействия, что защищает их от случайного изменения. Говорят, что данные *инкапсулированы*. Термины *сокрытие* и *инкапсуляция* данных являются ключевыми в объектно-ориентированном программировании.

Аналогично, если необходимо изменить данные объекта, то это действие также будет возложено на методы класса. Никакие другие функции не могут изменять данные класса. Такой подход облегчает написание, отладку и использование программы.

 *Классами* в Delphi называются специальные типы, которые содержат поля, методы и свойства. Как и любой другой тип, класс служит лишь образцом для создания конкретных экземпляров реализации, которые называются объектами. Пример описания класса:

**type**

 TMyClass = **class**(TObject) // тип класс с именем TMyClass

Field: Integer; // содержит в себе поле данных Field

**procedure** PrintField; // и метод PrintField

…

**end;**

**var**

MyClass: TMyClass; //объявление переменной типа класс

Классы призваны упростить разработку сложных программ и улучшить их качество. В основе классов лежат три фундаментальных принципа: *инкапсуляция, наследование* и *полиморфизм*.

Класс представляет собой единство трех сущностей – полей, методов и свойств. Их объединение в единое целое называется инкапсуляцией.

Любой класс может быть порожден от другого класса. Для этого при его описании указывается имя класса-родителя, например:

TChildClass = **class**(TParentClass)

В нашем примере класс TchildClass является потомком класса TParentClass.

Порожденный класс автоматически наследует поля, методы и свойства своего родителя и может добавлять новые. Этот механизм называется *наследованием*. Таким образом, принцип наследования обеспечивает поэтапное создание сложных классов и разработку собственных библиотек классов.

*Полиморфизм* – это свойство классов решать схожие по смыслу задачи разными способами. Поведение класса определяется набором входящих в его состав методов. Изменяя алгоритм того или иного метода в потомках класса можно придавать специфические действия классу-потомку, которые отсутствуют у класса-родителя.

### *Составляющие класса*

**Поля**

Полями называются данные класса. Поля могут быть любого типа, в том числе – классами, например:

**type**

TMyClass = **class**

IntField: Integer;

 StrField: **String**;

 ObjField: TObject;

**end**;

В данном примере описан новый тип – класс, содержащий три поля различных типов: IntField, StrField, ObjField.

Назовем объектом данного класса переменную типа класс, например:

**var**

MyClass: TMyClass; // описание объекта типа TMyClass

Каждый объект получает уникальный (принадлежащий только этому объекту) набор полей, но общий набор методов и свойств. В отличие от фундаментального принципа инкапсуляции, Delphi позволяет обращаться к полям объекта напрямую, т.е. не используя методов класса, например в нашем случае возможны такие обращения:

MyClass.IntField:=0;

MyClass.StrField:=’Пример строки’;

**Методы**

Методы класса – это процедуры и функции инкапсулированные в классе. Их можно объявлять также, как и обычные подпрограммы, например:

**type**

TMyClass = **class**

IntField: Integer;

 StrField: **String**;

 ObjField: TObject;

 **Procedure** PrintIntField;

 **Function** IncIntField (param: Integer): Integer;

**end**;

**var** MyObject:TMyClass;

Обращение к методам класса, также как и к его полям возможен при помощи составных имен, формирующихся следующим образом: указывается имя объекта, ставится точка, а затем указывается имя поля или метода, например:

MyObject.IntField:=10;

MyObject.PrintIntField;

Как уже говорилось выше, методы могут перекрываться в потомках, т.е. в классе-родителе может существовать метод с таким же именем, как и в потомке, например:

**type**

TParentClass=**class**

 **Procedure** Work;

 **end;**

TChildClass=**class** (TParentClass)

 **Procedure** Work;

 **end;**

Объекты обоих, описанных нами классов могут выполнять процедуру Work, но каждый из них будет делать это по-своему. Такое перекрытие методов называется статическим, т.к. выполняется непосредственно компилятором. В отличие от такого перекрытия существуют и другие способы создания одноименных методов, но их изучение выходит за рамки нашего курса.

**Свойства**

Свойства – это специальный механизм классов, регулирующий доступ к полям. Свойства объявляются при помощи зарезервированных слов **property, read** и **write**.Обычно свойство связано с некоторым полем и указывает те методы класса, которые должны использоваться при записи в это поле или при чтении из него, например:

**type**

TMyClass = **class**

IntField: Integer;

 **Procedure** GetIntField;

 **Procedure** PrintIntField;

 **Property** IntValue: Integer **read** GetIntField **write** PrintIntField;

**end**;

При выполнении программы свойство ведет себя как обычное поле. Например:

**var**

MyObject:TMyClass;

 Value: Integer;

**begin**

MyObject:=TMyClass.Create; {обязательное обращение к конструктору}

 MyObject.IntValue:=0;

 …

 Value:=MyObject.IntValue;

 …

 FreeAndNil(MyObject); //удаление ненужного объекта

**end.**

Описание любого класса может содержать секции (разделы), определяемые зарезервированными словами **published**(опубликованные), **private** (закрытые), **protected** (защищенные), **public** (открытые) и **automated** (автоматизированные). Внутри каждой секции вначале определяются поля, а затем методы и свойства.

Секции определяют области видимости элементов описания класса. Секция **public** не накладывает никаких ограничений на область видимости элементов – их можно вызвать из любого места программы. Элемент, объявленный в секции **private**, доступен только для методов определенных внутри этого класса, находящихся в одном модуле с классом. Секция **protected** напоминает предыдущую, но поля и методы объявленные в этой секции доступны также и в других модулях.

### *Структура программ Delphi*

Любая программа в Delphi состоит из файла проекта (с расширением dpr) и одного или нескольких модулей (с расширением pas). Каждый из таких файлов описывает программную единицу Delphi.

Файл проекта представляет собой программу, предназначенную для обработки компилятором. Эта программа создается автоматически и содержит лишь несколько строк, например:

**program** Project1;

**uses**

 Forms,

 Unit1 **in** 'Unit1.pas' *{Form1}*;

{$R \*.res}

**begin**

 Application.Initialize;

 Application.CreateForm(TForm1, Form1);

 Application.Run;

**end**.

Жирным шрифтом выделяются зарезервированные слова, а курсивом – комментарии. Текст программы начинается зарезервированным словом **program** и заканчивается словом **end** с точкой за ним.

Зарезервированные слова являются условными обозначениями для компилятора, предполагающее выполнение некоторых действий.

Комментарии, наоборот, ничего не значат для компилятора, и он их игнорирует. В Delphi комментарием считается любая последовательность символов, заключенная в фигурные скобки. В приведенном примере таких фрагментов кода два, но скобка {$R \*.res} комментарием не является. Этот фрагмент кода называется *директивой* компилятора. Отличие директив от комментариев состоит в том, что вслед за открывающей фигурной скобкой стоит знак $.

Аналогично языку программирования Pascal (а именно он является «прародителем» Delphi) здесь используются операторные скобки **begin … end**.

В объекте Application собраны данные и подпрограммы, необходимые для нормальной работы программы в целом. Данный объект создается автоматически для каждого нового проекта.

Файл проекта полностью формируется средой программирования Delphi и в большинстве случаев не предназначен для редактирования. При разработке сложных приложений этот файл можно использовать для выполнения некоторых специфических действий перед началом работы программы.

Рассмотрим структуру программного модуля Delphi.

Модули – это программные единицы, предназначенные для размещения отдельных частей программ. С помощью содержащегося в них программного кода реализуется вся поведенческая сторона программы.

Любой модуль имеет следующую структуру:

1. заголовок;
2. секция интерфейсных объявлений;
3. секция реализаций.

Заголовок открывается зарезервированным словом **unit**, за которым следует имя модуля и точка с запятой. Секция интерфейсных объявлений открывается зарезервированным словом **interface**, а секция реализаций – словом **implementation**. Модуль обязательно заканчивается словом **end** с точкой. Приведем общую схему модуля:

**unit** Unit1;

**interface**

// Секция интерфейсных объявлений

**implementation**

// Секция реализаций

**end.**

В интерфейсной секции описываются программные элементы (типы, классы, процедуры и функции), которые будут «видны» другим модулям, а в секции реализаций описывается механизм работы этих элементов.

В отличие от файла проекта программист может (и должен) вносить изменения в файлы модулей, придавая программе нужные функции. В отличие от файла проекта, файл модуля лишь частично формируется Delphi.

Элементы программы – это ее минимальные неделимые части, имеющее определенное значение для компилятора. К элементам относят:

1. зарезервированные слова;
2. идентификаторы;
3. типы;
4. константы;
5. переменные;
6. метки;
7. подпрограммы;
8. комментарии.

Большая часть этих понятий вам знакома из языка программирования Pascal.

Рассмотрим подробнее понятие типа.

Типы – это специальные конструкции языка, которые рассматриваются компилятором как образцы для создания других элементов программы, таких как переменные, константы и функции. Любой тип определяет две важные вещи: объем памяти, выделяемый для размещения элемента, и набор допустимых действий, которые программист может совершать над элементами данного типа.

Как и в Pascal’е в среде Delphi различают простые и составные типы. К простым относят порядковые (целые, логические, символьный, перечисляемый тип, тип-диапазон), вещественные, типы для хранения даты-времени. Основными из составных типов можно назвать строки, множества, записи, массивы, классы. Правила работы с переменными и константами определенного типа сохранились теми же, какими они были в Pascal’e.

Аналогично из Pascal’я были перенесены в Delphi многие операторы, например: составной оператор, условный оператор **if … then … else**, оператор выбора **case**, операторы организации циклических конструкций **for, while** и **repeat**; поэтому не будем останавливаться здесь на их описании.

### *Визуальное программирование*

В отличие от языков программирования (а точнее их оболочек) таких как TurboPascal, в Delphi некоторую часть программы создает непосредственно среда программирования, точнее программист имеет возможность создавать некоторые объекты «не задумываясь» над их технической реализацией. Приведем пример: для того, чтобы создать пустое окно в операционной системе Windows необходимо написать более 1000 строк программного кода. А в Delphi такое окно создается автоматически и называется **формой**.

Форма является основой любой визуальной программы, на ней располагаются все остальные элементы. Общий вид оболочки Delphi при запуске выглядит следующим образом: в верху экрана располагается главное меню и меню компонент; слева находятся два окна ObjectTreeView (Дерево объектов) и Object Inspector (Инспектор объектов). В первом из перечисленных окон отображаются названия всех объектов, использованных в программе, а во втором окне отображаются свойства выбранного (текущего) объекта, изменяя которые можно изменить некоторые характеристики объекта.



Программирование в Delphi производится при помощи помещения объектов, находящихся на палитре компонентов, на форму. Опишем наиболее часто используемые компоненты и вкладки, на которых они расположены.

**Вкладка Standard**

На данной вкладке палитры компонентов сосредоточены стандартные для Windows интерфейсные элементы, без которых не обходится практически ни одна программа.

* Frame – фрейм. Наравне с формой служит контейнером для размещения других компонентов. В отличие от формы, может размещаться в палитре компонентов, создавая заготовки компонентов.
* MainMenu — главное меню программы. Компонент способен создавать и обслуживать сложные иерархические меню.
* PopupMenu - контекстное, или локальное, меню. Обычно это меню появляется после щелчка правой кнопки мыши,
* Label — метка. Этот компонент используется для размещения в окне не очень длинных однострочных надписей.
* Edit - однострочное редактируемое текстовое поле. Предназначено для ввода, отображения или редактирования одной текстовой строки.
* Memo - многострочное редактируемое текстовое поле. Используется для ввода и/или отображения многострочного текста.
* Button - кнопка. Обработчик события OnClick этого компонента обычно используется для реализации некоторой команды.
* CheckBox - флажок. Щелчок мышью на этом компоненте в работающей программе изменяет его логическое свойство Checked.
* RadioButton - переключатель. Обычно объединяется как минимум еще с одним таким же компонентом в группу. Щелчок на переключателе приводит к автоматическому освобождению ранее выбранного переключателя в той же группе.
* ListBox — список. Содержит список предлагаемых вариантов (пунктов списка) и дает возможность проконтролировать текущий выбор.
* ComboBox — комбинированный список. Представляет собой комбинацию списка и однострочного текстового поля.
* ScrollBar- полоса прокрутки. Представляет собой вертикальную или горизонтальную полосу, напоминающую полосы прокрутки по бокам окна программы.
* GroupBox — панель группировки. Этот компонент используется для группировки нескольких связанных по смыслу компонентов.
* RadioGroup — группа переключателей. Содержит специальные свойства для обслуживания нескольких связанных переключателей.
* Panel — панель. Этот компонент, как и GroupBox, служит для объединения нескольких компонентов. Содержит внутреннюю и внешнюю кромки, что позволяет создать эффекты «вдавленности» и «выпуклости».
* ActionList — список действий. Служит для централизованной реакции программы на действия пользователя, связанные с выбором одного из группы однотипных управляющих элементов, таких как команды меню, графические кнопки и т. п.

**Вкладка Additional**

На вкладку Additonal помещены 26 дополнительных компонентов, с помощью которых можно разнообразить вид диалоговых окон.

* BitBtn — кнопка с изображением.
* Speed Button — кнопка панели инструментов. Обычно используется для быстрого доступа к тем или иным командам главного меню.
* MaskEdit — поле с маской ввода. Этот компонент способен фильтровать вводимый текст, например, для правильного ввода даты.
* StringGrid — текстовая таблица. Этот компонент обладает мощными возможностями для представления текстовой информации в табличном виде.
* DrawGrid — произвольная таблица. В отличие от StringGrid, ячейки этого компонента могут содержать произвольную информацию, в том числе и рисунки.
* Image — изображение. Этот компонент предназначен для отображения рисунков, в том числе значков и метафайлов.
* Shape — стандартная фигура. С помощью этого компонента вы можете вставить в окно правильную геометрическую фигуру — прямоугольник, эллипс, окружность и т. п.
* Bevel — кромка. Служит для выделения отдельных частей окна трехмерными рамками или полосами.
* ScrollBox — панель с полосами прокрутки. В отличие от компонента Panel, автоматически вставляет полосы прокрутки, если размещенные в нем компоненты отсекаются его границами.
* CheckListBox — список флажков. Отличается от стандартного компонента ListBox наличием рядом с каждым пунктом списка флажка CheckBox, что дает возможность выбора сразу нескольких пунктов,
* Splitter — вешка разбивки. Этот компонент размещается на форме между двумя другими видимыми компонентами и дает возможность пользователю во время прогона программы перемещать границу, отделяющую компоненты друг от друга.
* StaticText- текстовая метка, Отличается от стандартного компонента Label наличием собственного оконного ресурса, что позволяет обводить текст рамкой или выделять его в виде «вдавленной» части формы.
* ControlBar - контейнер для панелей инструментов. Служит контейнером для «причаливаемых» (Drag&Dock) компонентов.
* Application Events - обработчик сообщений Windows, Если этот компонент помещен на форму, он будет получать все предназначенные для программы сообщения Windows (без этого компонента сообщения принимает глобальный объект-программа Application).
* ValueListEditor- специализированный редактор списков, содержащих пары имя = значение. Пары такого типа широко используются в Windows, например, в файлах инициализации, в системном реестре и т. п.
* Labeled Edit - комбинация однострочного поля и метки.
* ColorBox - список выбора цвета. Специальный вариант компонента ComboBox для выбора одного из системных цветов.
* Chart - диаграмма. Этот компонент облегчает создание специальных панелей для графического представления данных.
* ActionManager — менеджер действий. Совместно с тремя следующими компонентам обеспечивает создание приложений, интерфейс которых (главное меню и инструментальные кнопки) может настраиваться пользователем.
* ActionMainMenuBar — строка меню для действий. Команды меню создаются с помощью компонента ActionManager.
* ActionToolBar - панель инструментов для действий. Служит контейнером для кнопок, создаваемых с помощью компонента ActionManager.
* CustomizeDlg- диалоговое окно настройки. С помощью этого компонента пользователь может по своему вкусу настроить интерфейс работающей программы.
* XPColorMap - совместно с тремя следующими компонентами впервые введен в версии 7 для настройки цветов и наполнения панелей ActionToolBar.
* StandardColorMap- в функциональном плане подобен компоненту XPColorMap. WilightColorHap - в функциональном плане подобен компоненту XPColorMap.

**Вкладка System**

На вкладке System представлены компоненты, которые имеют различное функциональное назначение, в том числе компоненты, поддерживающие стандартные для Windows технологии межпрограммного обмена данными OLE (Object Linking and Embedding — связывание и внедрение объектов) и DDE (Dynamic Data Exchange — динамический обмен данными).

* Timer — таймер. Этот компонент служит для отсчета интервалов реального времени.
* PaintBox — окно для рисования. Создает прямоугольную область, предназначенную для прорисовки графических изображений.
* Media Player — медиаплейер. С помощью этого компонента можно управлять различными мультимедийными устройствами.
* OteContainer — OLE-контейнер. Служит приемником связываемых или внедряемых объектов.

**Вкладка Dialogs**

Компоненты вкладки Dialogs реализуют стандартные для Windows диалоговые окна.

* Open Dialog — окно открытия файла. Реализует стандартное диалоговое окно открытия файла.
* SaveDialog — окно сохранения файла. Реализует стандартное диалоговое окно сохранения файла.
* OpenPictureDialog - окно открытия изображения. Реализует специальное окно выбора графических файлов с возможностью предварительного просмотра изображений.
* SavePictureDialog — окно сохранения изображений. Реализует специальное окно сохранения графических файлов с возможностью предварительного просмотра рисунков.
* FontDialog - окно выбора шрифта. Реализует стандартное диалоговое окно выбора шрифта.
* ColorDialog — окно выбора цвета. Реализует стандартное диалоговое окно выбора цвета.
* PrintDialog — окно настройки параметров печати. Реализует стандартное диалоговое окно выбора параметров для печати документа.
* PrinterSetupDialog - окно настройки параметров принтера. Реализует стандартное диалоговое окно для настройки печатающего устройства.
* FindDialog — окно поиска. Реализует стандартное диалоговое окно поиска текстового фрагмента.
* ReplaceDialog — окно поиска и замены. Реализует стандартное диалоговое окно поиска и замены текстового фрагмента.