Оглавление

1. Быстрая разработка приложений. Разработка MDI–приложения 2
2. Быстрая разработка приложений. Разработка SDI–приложения 8
3. Быстрая разработка приложений. Методы отладки 14
4. Быстрая разработка приложений. Создание справочной системы приложения с помощью утилиты Help Workshop 18
5. Быстрая разработка приложений. Создание дистрибутива приложения 22

Быстрая разработка приложений. Разработка MDI–приложения

КОД: 0641.02.02/ПТ.0641.07

Продолжительность:
80 мин.

Дисциплина:
Технология программирования. Юнита 2.

Предназначено:
Для студентов по направлению ***и***нформатика и ***ВТ*** в соответствии с учебным планом.

Цель: Изучить процесс проектирования и программирования приложения с MDI–интерфейсом.

Результат обучения:
- изучение особенностей MDI и SDI–стилей приложений;
- закрепление навыков работы в среде RAD ;
- изучение способа создания MDI–приложения.

Используемые программы:
ИСР – 6.

Используемые папки:
ПТ_0641_07

План занятия:

Часть I. Быстрая разработка приложений	**10 минут**
Часть II. Создание приложения в стиле MDI	**30 минут**
Часть III. Самостоятельная работа	**40 минут**

ЗАПУСК ПРОГРАММЫ:
Предполагается, что требуемые программы уже инсталлированы на диске.
(См. «Инструкцию по установке программы на ПК»)

Часть I. Быстрая разработка приложений

Рекомендуемое время
10 минут

Модель быстрой разработки приложений (Rapid Application Development) — пример применения инкрементной стратегии конструирования. Инкрементная модель характеризуется тем, что в начале процесса проектирования определяются все пользовательские и системные требования, оставшаяся часть конструирования выполняется в виде последовательности версий. RAD–модель обеспечивает экстремально короткий цикл разработки. Быстрота разработки достигается за счет использования компонентно–ориентированного конструирования. Если требования полностью определены, а проектная область ограничена, RAD–процесс позволяет группе создать полностью функциональную систему за очень короткое время. Скорость выполнения работы определяется тем, что RAD–процесс работает с повторно используемыми программными компонентами или создает повторно используемые компоненты. Для обеспечения конструирования используются утилиты автоматизации.

В RAD–средах базовой парадигмой является объектно–ориентированное программирование. В основе ООП лежит понятие объекта, как некоего самостоятельного элемента (составной части программного кода), предназначенного для выполнения строго определенных операций над некоторыми данными.

Однажды написанные и отлаженные объекты позволяют повторно использовать программный код. Имея под рукой коллекцию готовых объектов программист с легкостью может вставить любой из них в новую программу.

Особенно плодотворной эта идея оказалась при разработке программных интерфейсов. Специальные объекты — компоненты позволяют собирать интерфейсы программ из нужных элементов управления за считанные минуты.

Применение объектов в практике программирования повышает надежность программ, так как каждый объект работает только со своими данными, а попытка использовать чужие переменные противоречит принципам ООП.

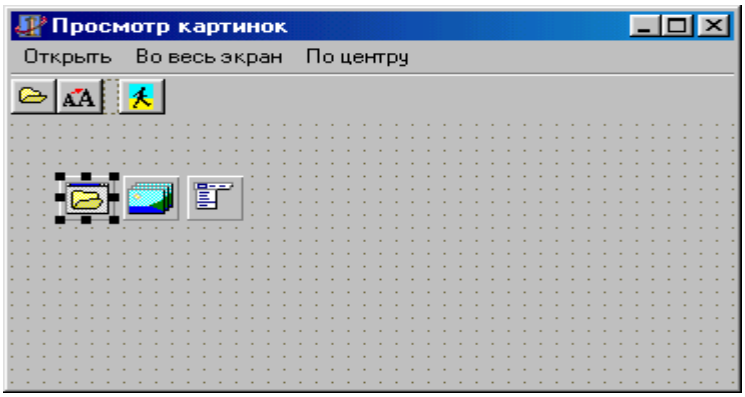
С появлением инструментов визуального программирования, таких, как Visual C, Visual Basic, Power Builder, , создание графического пользовательского интерфейса перестало быть прерогативой лишь немногих специалистов. Визуальное программирование позволило свести проектирование пользовательского интерфейса к простым и наглядным процедурам, которые дают возможность за минуты или часы сделать то, на что ранее уходили месяцы работы.

Часть II. Создание приложения в стиле MDI

Рекомендуемое время
30 минут

Подавляющее большинство современных приложений обеспечивают взаимодействие с пользователем на основе применения элементов управления, размещенных в одном или нескольких окнах. В зависимости от способа взаимодействия окон приложения с главным окном различают типы интерфейсов приложений.
Форма является основным строительным блоком в . Форма — это окно на этапе разработки.

1. Начните новое приложение (**File|New Application**).
2. Разновидности форм определяются значениями их свойств **FormStyle**, а также разнообразием форм–заготовок, хранящихся в **репозитории** . Выделите свойство **FormStyle** в Инспекторе объектов и посмотрите список значений этого свойства.
   Стили **fsMDIChild** и **fsMDIForm** используются при создании так называемых многодокументных приложений в стиле **MDI** (Multi Document Interface)**.** Этот стиль предполагает создание главного окна, внутри которого по мере надобности появляются дочерние окна. Дочерние окна не могут существовать без главного, — при закрытии приложения и уничтожении главного окна все открытые дочерние окна также сворачиваются. Примером такого стиля является текстовый редактор Word.
   Среда представляет собой многооконное приложение в стиле SDI, который не накладывает ограничений на положение и размеры вспомогательных форм. Для создания форм в этом случае используется стиль **fsNormal**.
   Стиль **fsStayOnTop** используется для окон, которые всегда должны располагаться над всеми другими окнами программы.
3. Разработаем приложение для просмотра графических изображений. Для этого воспользуемся **MDI**–стилем.



   3.1. В главном окне, содержащем инструменты по работе с графическими изображениями, можно будет открыть несколько дочерних окон. Установите свойство **FormStyle** родительской формы равным **fsMDIForm**.
   3.2. Заголовок формы должен носить информационный характер. Задайте свойству **Caption** формы значение «Просмотр картинок».
   3.3. Свойство **Name** определяет имя, под которым компонент будет известен программе. Рекомендуется давать компонентам имена «со смыслом». Задайте свойству ***Name*** формы значение ***M_Parent***
   3.4. Свойство **Hint** определяет содержание ярлычка (всплывающей подсказки). Для того, чтобы компонент мог показать ярлычок нужно свойству **ShowHint** присвоить значение **true**.
   3.5. Для создания инструментальных панелей предназначен компонент **TToolBar,** расположенный на панели **Win32** палитры компонентов. Поместите на форму компонент **TToolBar.**
   Для компонента специально разработан компонент **TToolButton.** Щелкните правой кнопкой на компоненте **TToolBar** и в контекстном меню выберите пункт **NewButton.** Это кнопки класса **TToolButton**. Добавьте еще одну кнопку на инструментальную панель. Для

функционального выделения групп элементов предназначены сепараторы. Вставьте эту разновидность кнопки, выбрав пункт меню **NewSeparator.** Добавьте еще одну кнопку на панель.

3.6. Поместите на форму компонент **TImageList** и, двойным щелчком по компоненту, вызовите редактор **ImageList Editor**. С помощью кнопки **Add** в контейнер изображений помещается картинка. Добавьте изображение **FldrOpen,** которое расположено в подкаталоге **Images\Buttons.** Растры из каталога **Images\Buttons** имеют по два изображения. Удалите второе изображение — нажмите кнопку **Delete.** Добавьте еще одну картинку из каталога **Buttons** (имя файла **FontSize**). Не забудьте удалить второе изображение. Имя следующей картинки **Picture.** Добавьте ее в контейнер изображений.

3.7. Выделите компонент **TToolBar.** В Инспекторе объектов найдите свойство **Images**. Это свойство позволяет назначить набор картинок. Выберите **ImageList1** из выпавшего списка свойства.

3.8. Свойство **Hint** определяет содержание ярлычка — небольшого справочного окна возле элемента, на котором остановился курсор. Назначьте свойству **Hint** компонентов **ToolButton1, ToolButton2, ToolButton3** значения «Загрузить», «Заполнить клиентскую область», «Центровать».

3.9. Установите на форму компонент **TOpenDialog**. Свойство **Filter** типа String компонента **TOpenDialog** используется для фильтрации файлов, показываемых в диалоговом окне. Свойству **Filter** задайте значение**: BitMaps (*.bmp)]*.bmp.** Если опции **ofPathMustExist** и **ofFileMustExist** в свойстве **Options** определены как **True**, то диалоговое окно разрешает указывать файлы только из существующих каталогов и только существующие файлы. Установите эти свойства.

3.10. Теперь создадим дочернюю форму. Выберите из меню **File** команду **New Form**. Установите свойство **FormStyle** равным **fsMDIChild**.

Свойство **Position** определяет положение и размеры окна в момент его появления на экране. Если это свойство равно **poDefaultPosOnly**, то положение формы будет как на этапе конструирования, а размеры окна устанавливает Windows. Установите это значение в свойстве **Position**.

Свойству **Name** задайте значение **F_Child**.

3.11. Поместите компонент **TImage** во вновь созданную форму и установите его свойство **Align** равным **alClient**.

3.12. Удалим дочернюю форму из списка автоматически создаваемых форм. Вызовите диалоговое окно опций проекта, выбрав команду **Project**|**Options.** Выберите **F_Child** в списке **Auto-Create forms** и щелкните по кнопке со стрелкой вправо. Форма **F_Child** перенесена в список **Available forms** (список форм, которые не создаются автоматически). Закройте окно установки опций проекта.

Сохраните проект в своем каталоге Lab7, выбрав команду **File|Save Project As**

3.13. Создайте обработчик события для первой кнопки панели инструментов.

Выбор графического файла, отображаемого в дочерней форме, осуществляется с помощью компонента **TOpenDialog**. Впишите в заготовку обработчика следующий код:

```
If OpenDialog1.Execute Then
        //В случае, если графический файл выбран
    Begin
      F_Child:= TF_Child.Create(Self);
      // Для создания нового экземпляра формы используется конструктор //Create.
      F_Child.Caption := OpenDialog1.FileName;
      //В заголовок новой формы запишем имя файла, который загружается в
      //компонент Image1 формы.
      F_Child.Image1.Picture.LoadFromFile(OpenDialog1.FileName);
      //Для того чтобы загрузить картинку в компонент Image1, воспользуемся
      //методом LoadFromFile.
    End;
```

При разработке **MDI**–приложения метод **Show** не нужен, так как автоматически показывает все вновь созданные дочерние **MDI**–формы.

3.14. Вторая кнопка панели инструментов изменяет свойство **Stretch** компонента **Image1**. Если значение этого свойства **True**, то рисунок заполняет всю клиентскую область компонента. Двойным щелчком по кнопке создайте обработчик события **OnClick**, в котором запишите операторы**:**

```
//Свойство ActiveMDIChild ссылается на активную (имеющую фокус) форму
```

```
F_Child := M_Parent.ActiveMDIChild as TF_Child;
// Рисунок при каждом нажатии кнопки изменяет свое положение
if F_Child.Image1.Picture.Stretch Then
                                    F_Child.Image1.Picture.Stretch :=False
                                Else
                                    F_Child.Image1.Picture.Stretch :=True;
```

3.15. При нажатии на третью кнопку панели компонентов изображение помещается по центру компонента **Image1**. Свойство **Center** позволяет центрировать изображение в границах компонента. Создайте обработчик события **OnClick** для третьей кнопки, позволяющий центровать изображение.

3.16. Создайте главное меню формы с пунктами: «Открыть», «Во весь экран», «По центру». Назначьте обработчикам **OnClick** этих пунктов меню соответствующие обработчики кнопок на панели **ToolBar1**.

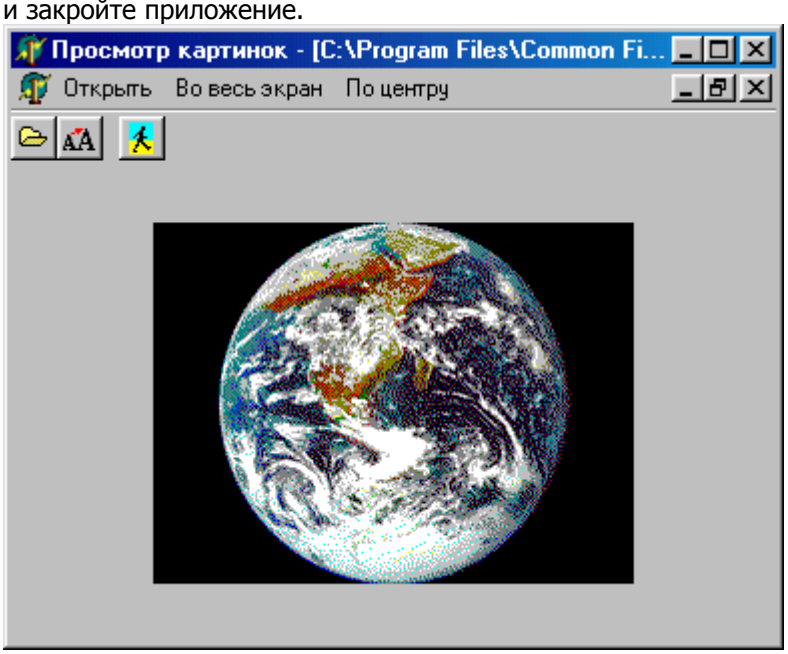
3.17. Запустите приложение на выполнение. Установите курсор мыши последовательно над кнопками панели инструментов. Кнопки панели инструментов снабжены подсказками, которые Вы поместили в свойство **Hint**.

Для загрузки графического файла нажмите на первую кнопку. Загрузите файл **Skyline** в созданное Вами приложение. Измените размеры дочерней формы: сверните дочернее окно, восстановите его, разверните на весь экран.

Нажмите на вторую кнопку панели инструментов. Картинка заняла всю клиентскую область компонента **Image1**. Нажмите еще раз на эту же кнопку. Размеры картинки восстановлены.

Нажмите на третью кнопку панели инструментов. Свойству **Center** присвоено значение **True** и картинка расположилась по центру компонента **Image1**.

Загрузите новый графический файл в приложение. Для этого нажмите на кнопку загрузки и в диалоговом окне выберите файл **Handshak.bmp.** Создано новое дочернее окно. В окно загружен выбранный Вами файл. Нажав на вторую кнопку, увеличьте размер изображения и закройте приложение.



Часть III. Самостоятельная работа

Рекомендуемое время
40 минут

Создайте приложение в стиле MDI в соответствии с вариантом.

Примечание 1: Для того чтобы дочернее окно имело размерные кнопки, в родительском окно должно присутствовать главное меню.

Примечание 2: Справка по работе со стандартными диалогами приведена в приложении 1.

№ варианта	Условие задачи	Примечание
1	Разработайте приложение, представляющее собой простейший редактор. В программе предусмотрите возможности: загрузить текстовый файл, записать текст на диск, изменять начертание и цвет шрифта.	Загрузить файл — LoadFromFile(<имя файла>). Записать файл — SaveToFile(<имя файла>).
2	Разработайте приложение, которое представляет собой комбинацию текстового и графического редактора. В программе должна существовать возможность загружать и сохранять текстовые и графические файлы.	Загрузить файл — LoadFromFile(<имя файла>). Записать файл — SaveToFile(<имя файла>).
3	Разработайте приложение для ввода результатов сессии. Организуйте табличный ввод, используя компонент TSringGrid. Предусмотрите запись результатов в текстовый файл на диск.	Содержание клетки таблицы в свойстве Cells[ACol, ARow].
4	Разработайте приложение, которое представляет собой комбинацию графического редактора и таблицы TStingGrid. В программе должна существовать возможность загружать и сохранять графические файлы, а также редактировать и записывать в текстовый файл содержимое таблицы.	См. примечания предыдущих вариантов.
5	Разработайте приложение «Табулирование функций». Программа позволяет получить значения аргумента и функции в заданном интервале с заданным шагом. Вид функции можно задать выбором в компоненте TListBox или в компоненте TRadioGroup.	Строки компонента TListBox (TRadioGroup) содержатся в свойстве Items. Номер выбранной строки в свойстве ItemIndex.

Приложение 1.

TOpenDialog — диалоговая панель выбора открываемого файла по шаблону.
TSaveDialog — диалоговая панель создания файла.
TFontDialog — диалоговая панель выбора шрифта и его характеристик.
TColorDialog — диалоговая панель выбора цвета.
TPrintDialog — диалоговая панель вывода на устройство печати.
TPrinterSetupDialog — диалоговая панель настройки устройства печати.
TFindDialog — диалоговая панель поиска.
TReplaceDialog — диалоговая панель замены.
TOpenPictureDialog — диалоговая панель выбора графического изображения с просмотром.
TSavePictureDialog — диалоговая панель сохранения графического изображения с просмотром.

Основные правила использования диалоговых панелей.

После того как компонент помещен на форму, необходимо связать отображение этой панели с каким–либо событием. Чаще всего таким событием является выбор команды меню или нажатие определенной кнопки. В обработчике события необходимо использовать метод *Execute*. При обращении к Execute на экране появляется соответствующее диалоговое окно. Окно диалога является модальным. Модальное окно, как известно, приостанавливает выполнение программы до тех пор, пока пользователь не закроет окно. Execute — логическая функция. Проанализировав результат Execute, программа может присвоить возвращаемые значения свойствам тех компонентов, на которые они влияют.

Например,

```
Procedure TForm1.Button1Click(Sender : TObject);
Begin
  If ColorDialog1.Execute then
  Form1.Color := ColorDialog1.Color;
End;
```

В первой строке этого метода вызывается стандартная диалоговая панель выбора цвета, а во второй — значение, возвращенное этой диалоговой панелью, присваивается свойству Color формы.

Компоненты TOpenDialog и TSaveDialog имеют идентичные свойства.

Свойство FileName содержит маршрут поиска и выбранный файл при успешном завершении диалога. Программа может использовать это свойство для доступа к файлу с целью читать из него

данные или записывать в него. Свойство Filter используется для фильтрации файлов, показываемых в диалоговом окне. Это свойство можно устанавливать с помощью специального редактора или программно.

Установить начальный каталог позволяет свойство InitialDir.

Быстрая разработка приложений. Разработка SDI–приложения

КОД: 0641.02.02/ПТ.0641.08

Продолжительность:
80 мин.

Дисциплина:
Технология программирования. Юнита 2.

Предназначено:
Для студентов по направлению ***и***нформатика и ***ВТ*** в соответствии с учебным планом.

Цель: Изучить процесс проектирования и программирования приложения с SDI–интерфейсом.

Результат обучения:
- закрепление навыков работы в среде RAD ;
- умение связывать воздействия пользователя с реакциями на них с помощью компонента TActionList;
- умение организовать интерфейс Drag&Dock.

Используемые программы:
ИСР – 6.

Используемые папки:
ПТ_0641_08

План занятия:

Часть I. Пример разработки приложения в стиле SDI	**30 минут**
Часть II. Самостоятельная работа	**50 минут**

ЗАПУСК ПРОГРАММЫ:
Предполагается, что требуемые программы уже инсталлированы на диске.
(См. «Инструкцию по установке программы на ПК»)

Часть I. Пример разработки приложения в стиле SDI

Рекомендуемое время
30 минут

На основе компонента **TMemo** создадим текстовый редактор, позволяющий выполнять часто встречающиеся команды: вырезать, вставить, найти, заменить и другие.

1. Создайте новое приложение. Установите на форму компоненты TMainMenu, TPopupMenu. Организуйте пункты главного меню:

Файл	**Поиск**	**Шрифт**	**Выход**
Создать	**Найти**		
Открыть	**Заменить**		
Сохранить			
Сохранить как			

   Вспомогательное меню должно содержать такие же пункты.
2. Установите на форму компонент **TImageList**, в который поместите графические файлы из каталога **Buttons**. Картинки должны соответствовать действиям, перечисленным в меню.
3. В нашем приложении мы будем выполнять одни и те же действия разными способами — выполнением пунктов главного и вспомогательного меню, нажатием кнопки на панели инструментов. Для связывания воздействия пользователя с реакциями на них разработан компонент **TActionList**. Установите компонент **TActionList** (панель **Standard**) на форму. Вызовите редактор списка действий двойным щелчком мышью на компоненте.
   Первая кнопка панели инструментов предназначена для добавления действия. Добавьте действие.
   Имя первого действия **Action1**. В инспекторе объектов отображены свойства класса **TAction**. Измените свойство **Caption** на «Создать». Свойству **ImageIndex** задайте значение 0.
   Нажмите на первую кнопку панели инструментов и добавьте еще одно действие. Имя второго действия **Action2.** Измените свойство **Caption** на «Открыть». Свойству **ImageIndex** задайте значение 1.
   Добавьте пять действий. Действие добавляется с помощью кнопки на панели инструментов. В свойство **Caption** необходимо ввести: «Сохранить», «Сохранить как», «Найти», «Заменить»,

«Шрифт». Свойству **ImageIndex** задайте соответственно значения: 2, 3, 4, 5, 6, Закройте редактор списка действий.

Свойству **Images** компонента **TActionList** задайте значение **ImageList1** из выпадающего списка значений этого свойства.

4. Установите на форму второй компонент **TActionList**. Вызовите редактор списка действий двойным щелчком по компоненту. Для программиста предусмотрен набор типовых, наиболее часто встречающихся действий. Откройте список правее кнопки «Добавить» и выберите пункт **New Standard Action**

   К типовым действиям относятся операции для работы с буфером обмена. В списке типовых действий выделите четыре действия от **TEditCopy** до **TEditPaste** (при выделении держите клавишу Shift нажатой). Нажмите на Ok.

   В правой части окна редактора выделите **TEditDelete** и второй кнопкой панели инструментов удалите действие. Выделите действие **TEditCut** и с помощью кнопки «Стрелка вверх» переместите его на одну строку вверх.

   Измените значения свойства **Caption** для этих действий на «Вырезать», «Копировать», «Вставить».

   Теперь необходимо присвоить свойству **Action** каждого пункта меню соответствующее действие. Выберите пункт меню «Создать» главного меню формы. В списке действий, который открывается в свойстве **Action**, выберите **Action1**. Продолжите установку свойства **Action** для остальных пунктов меню. Последовательно выделяйте пункты меню и назначайте им действия из списка действий в свойстве **Action**.

   Двойным щелчком мыши на компоненте вызовите редактор **PopupMenu** и установите действия для пунктов контекстного меню.

5. Добавим в приложение панель инструментов. Компонент **TControlBar** служит контейнером для размещения инструментальных панелей. Этот компонент использует технологию **Drag&Dock** (причаливание) для управления положением панелей. Установите на форму компонент **TControlBar,** который расположен на панели **Additional**.

   Установите в его свойство **Align** значение **alTop**, в **AutoSize** — **True**, в **DragKind** — **dkDock**. Оставьте в свойстве **DragMode** значение **dmManual**.

   Поместите на компонент **TCoolBar** компонент **TToolBar (**панель **Win32** палитры компонентов**).** Для вставки кнопки щелкните по **ToolBar1** правой кнопкой мыши и выберите опцию **NewButton**. Добавьте еще три кнопки.

   Выделите компонент **ToolBar1.** В свойство **DragKind** компонента **ToolBar1** поместите значение **dkDock**, в **DragMode** — **dmAutomatic** и в **AutoSize** — **True.**

   Поместите на компонент **TCoolBar** еще один компонент **TToolBar.** Поместите на панель шесть кнопок (правая клавиша, **NewButton**).

   Выделите компонент **ToolBar2.** В свойство **DragKind** компонента **ToolBar2** поместите значение **dkDock**, в **DragMode** — **dmAutomatic** и в **AutoSize** — **True**.

   Для того чтобы на кнопке отображалась надпись вместо картинки, необходимо изменить свойство **ShowCaption**. В инспекторе объектов найдите свойство **ShowCaption** и установите значение этого свойства равным **True**.

   Выделите панель **ToolBar1**. В инспекторе объектов найдите свойство **Images**. Установите значение этого свойства равным **ImageList1**.

   Определите для кнопок свойство **Action**.

   Сделать это нужно следующим образом:

   1) выделяете кнопку;
   2) обращаетесь к свойству Action;
   3) в списке выбираете действие.

   Для первой панели — Action1, Action2, Action3, Action4.

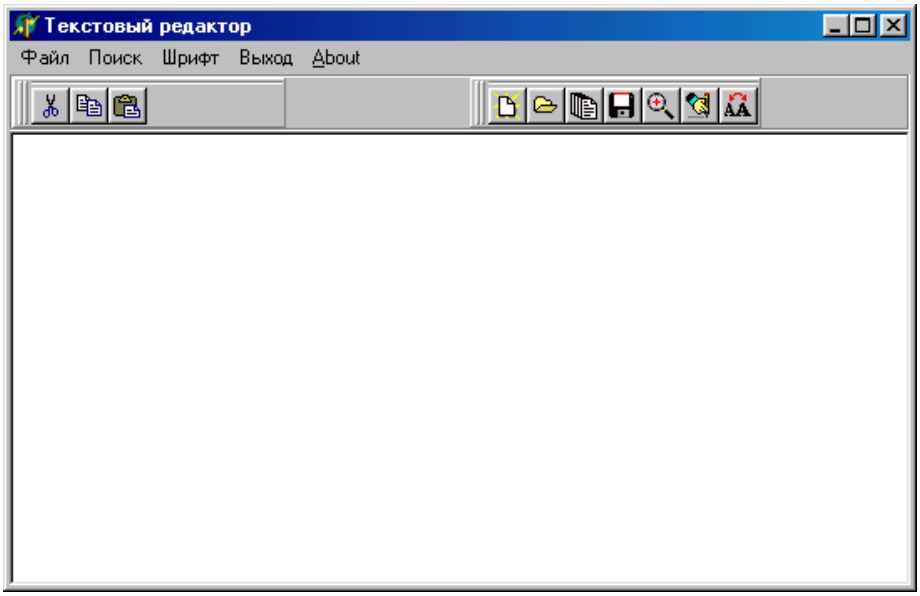
   Для второй панели — EditCut1, EditCopy1, EditPaste1, Action5, Action6, Action7.

6. Запустите приложение на выполнение.

   На панелях имеются кнопки управления. Ухватитесь за кнопку управления первой панели и перетащите панель в центр окна. Ухватитесь за кнопку управления второй панели и потяните панель влево так, чтобы кнопки расположились в один ряд. Отбуксируйте первую панель назад на контейнер панелей.

   Завершите работу программы.

7. Поместите на форму компонент **TMemo**. Свойству **Align** назначьте значение **alClient**. Очистите свойство **Lines**. Установите на форму компоненты-диалоги: **TOpenDialog**, **TSaveDialog**, **TFontDialog**, **TFindDialog**, **TReplaceDialog**.



8. Разработаем функциональность формы. Главное событие каждого действия компонента **TActionList** — событие **onExecute**. Оно определяет реакцию на это действие.

8.1. Вызовите редактор списка действий двойным щелчком по компоненту. Выполните двойной щелчок по имени первого действия. Вы создали заготовку обработчика для события «Создать». Введите операторы:

```
//Если в редакторе есть текст, то записать его на диск
if SaveDialog1.FileName <> '' Then
                    Memo1.Lines.SaveToFile(SaveDialog1.FileName);
Memo1.Clear; // Очистить редактор
```

8.2. Выполните двойной щелчок по имени второго действия. Вы создали заготовку обработчика для события «Открыть».

Введите операторы

```
With OpenDialog1 Do
If Execute Then //Если в диалоге выбран файл, то загрузить его в редактор
        begin
          Memo1.Lines.LoadFromFile(FileName);
          Caption:= 'Мой редактор - '+ FileName;
          SaveDialog1.Filename:= FileName;
          FileName:= '';
        end;
```

8.3. Создайте обработчики действий для записи файла на диск, выбора шрифта, окончания работы приложения (методы **Close** и **Terminate**), а так же для действий **EditCut1**, **EditCopy1**, **EditPaste1**. Чтобы создать обработчики для последних трех действий, воспользуйтесь методами компонента TMemo: **CutToClipboard, CopyToClipboard, PasteFromClipboard.**

8.4. Для события «Найти» (Action5) создайте следующий обработчик:

```
procedure TForm1.Action5Execute(Sender: TObject);
  begin
    FindDialog1.Execute;
    If pos(FindDialog1.FindText,Memo1.Text)<>0 Then
                    // FindText — текст,введенный в диалоговое окно поиска
    begin
      Memo1.HideSelection:=False;
      Memo1.SelStart:=pos(FindDialog1.FindText,Memo1.Text)-1;
                    // SelStart — первая позиция выделенного текста
      Memo1.SelLength:=Length(FindDialog1.FindText);
                    // SelLength — длина выделенного текста
    end;
  end;
```

8.5. Создайте обработчик для события «Заменить» (Action6).

```
procedure TForm1.Action6Execute(Sender: TObject);
  label 10;
    begin
```

```
        replaceDialog1.Execute;
        10:
            If pos(ReplaceDialog1.FindText, memo1.Text)<>0 Then
            begin
                Memo1.SelStart:=pos(ReplaceDialog1.FindText,Memo1.Text)-1;
                Memo1.SelLength:=Length(ReplaceDialog1.FindText);
                memo1.SelText:=ReplaceDialog1.ReplaceText;
                        // Выделенный текст заменяем на текст замены
                        //из диалогового окна «Найти/Заменить»
                goto 10;
            end;
    end;
```

9. Добавьте к приложению форму AboutBox из репозитория (**File/New/Forms**). Добавьте еще один пункт меню «О программе». Создайте обработчик события для этого пункта меню, в котором запишите код, позволяющий визуализировать форму **AboutBox** (AboutBox.Show). Форма содержит справочную информацию о программе.
10. Запустите приложение на выполнение. Выполните следующие действия:
    - Введите текст.
    - Запишите текст на диск.
    - Создайте новый документ.
    - Откройте сохраненный текст.
    - Осуществите поиск и замену.
    - Скопируйте, вырежьте, вставьте фрагмент текста.

    Если программа работает без ошибок — сохраните ее в своем каталоге Lab8.
11. Репозиторий содержит заготовки форм и приложений. Вызовите окно репозитория, откройте страницу Projects. На этой странице расположены заготовки приложений в стиле SDI и в стиле MDI. Вызовите последовательно эти проекты и ознакомьтесь с их содержанием.

Часть II. Самостоятельная работа

Рекомендуемое время
50 минут

В среде разработано приложение в стиле SDI. Создана программа для работы с базой данных — телефонным справочником, сформированным в виде файла записей. В каждой записи определены поля: фамилия и имя (строки длиной по 20 символов каждая), адрес и номер телефона (строки длиной 100 и 15 символов соответственно).

В программе реализованы следующие операции: считывание данных из файла, запись данных в файл, навигация по базе, поиск записи в списке.

Интерфейс приложения на рисунках 1 и 2. На форме расположен многостраничный блокнот.

На первой странице блокнота (рис. 1) в табличном виде выводится содержимое базы данных. Табличные данные могут корректироваться. Содержимое таблицы может быть записано в файл.

На второй странице блокнота (рис. 2) реализована навигация по записям базы данных. В раскрывающемся блокноте можно выбрать фамилию из базы данных щелчком мыши. Записи отображаются в окнах ввода.

Познакомьтесь с работой приложения, для этого откройте проект Project1 из каталога Проект_Телефонный_справочник. Часть пунктов меню неактивны. Вам предстоит осуществить реализацию этих действий.

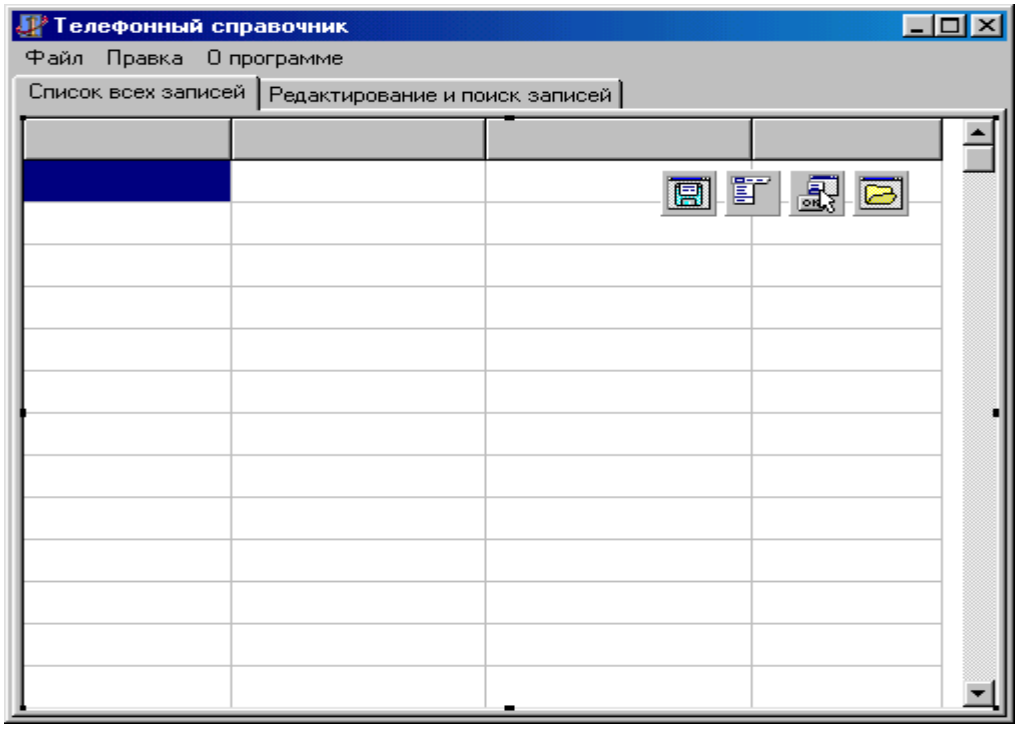


Рис. 1. Страница 1 многостраничного блокнота.

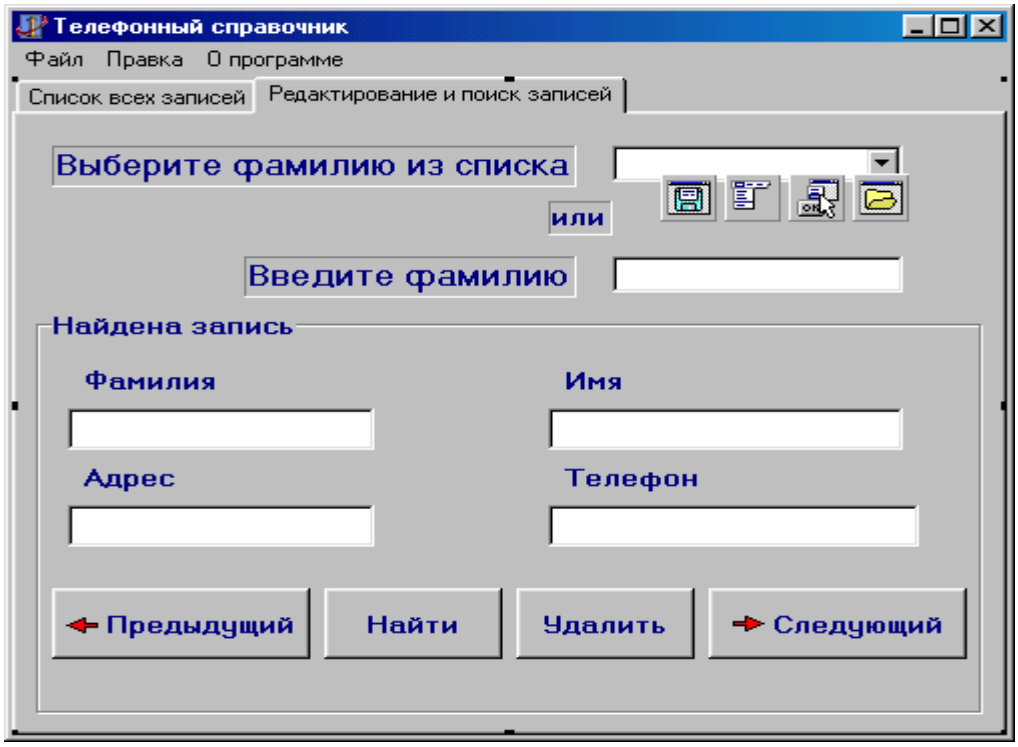


Рис. 2. Страница 2 многостраничного блокнота.

Задание

1. Доработайте приложение. Внесите исправления в форму «О программе»: укажите свою фамилию и группу, измените картинку.
2. В момент выхода из программы обеспечьте появления окна для сохранения файла телефонного справочника в случае, если в нем были сделаны изменения.
3. Добавьте контекстное меню, в которое включите все пункты меню и активных кнопок Вашего приложения.
4. Допишите программу в соответствии с заданием Вашего варианта. Отладку программы можно продолжить на следующем занятии.
5. Снабдите приложение панелями инструментов с механизмом Drag&Dock.

№ варианта	Условие задачи	Примечание
1	Запрограммируйте кнопку «Найти» так, чтобы на панели «Найдена запись» отображалась информация о лице, фамилия которого указана в поле редактирования Edit5 «Введите фамилию». Если такой фамилии нет, то выведите сообщения об этом в отдельном окне.	Добавьте действие в TActionList и используйте обработчик Execute.
2	Запрограммируйте команды подменю «А -> Я» и «Я -> А» пункта главного меню «Сортировать» так, чтобы фамилии в таблице и комбинированном списке были отсортированы в алфавитном или обратном порядке.	Используйте хранилище действий TActionList.
3	Удалите команду «Сохранить изменения в списке записей» из главного меню и обеспечьте сохранение изменений в момент редактирования таблицы и полей на панели «Найдена запись».	Используйте хранилище действий TActionList.
4	Запрограммируйте кнопку «Удалить» так, чтобы после нажатия на нее удалялись данные на панели «Найдена запись» и из комбинированного списка. В таблице такая запись должна быть помечена звездочкой.	Используйте хранилище действий TActionList.
5	1) Запрограммируйте поиск фамилии по номеру телефона. Найденная информация должна отображаться на панели «Найдена запись». 2) В случае попытки открыть несуществующий файл телефонного справочника обеспечьте появление окна с соответственным сообщением.	Используйте хранилище действий TActionList.

Быстрая разработка приложений. Методы отладки в среде

КОД: 0641.02.02/ПТ.0641.09

Продолжительность:
80 мин.

Дисциплина:
Технология программирования. Юнита 2.

Предназначено:
Для студентов по направлению ***и***нформатика и ***ВТ*** в соответствии с учебным планом.

Цель: Изучить возможности встроенного отладчика среды быстрой разработки приложений .

Результат обучения:
- овладение знаниями по вопросам классификации ошибок, методах трассировки программы, точках прерываний;
- овладение методами просмотра и изменения значений переменных в процессе отладки программы;
- опыт работы в RAD–среде.

Используемые программы:
ИСР – 6.

Используемые папки:
ПТ_0641_09

План занятия:
Часть I. Изучение возможностей встроенного отладчика **20 минут**

Часть II. Самостоятельная работа **60 минут**

ЗАПУСК ПРОГРАММЫ:
Предполагается, что требуемые программы уже инсталлированы на диске.
(См. «Инструкцию по установке программы на ПК»)

Часть I. Изучение возможностей встроенного отладчика

Рекомендуемое время
20 минут

1. Ошибки, возникающие в процессе проектирования и программирования приложений можно подразделить на:
   - синтаксические,
   - времени выполнения,
   - логические.

   Редко удается ввести все операторы без единой ошибки. Даже самые опытные программисты допускают помарки при вводе операторов. Синтаксические ошибки, их также называют ошибками времени компиляции (compile-time error), наиболее легко устранимы. Синтаксические ошибки обнаруживаются компилятором автоматически. Сообщения о найденных ошибках отображаются в нижней части редактора. Сообщение описывает ошибку достаточно подробно, чтобы можно было понять ее причину.
   На этапе компиляции может выдавать подсказки или предупреждения (Warning) в нижней части окна кода . Так среда предупреждает об оплошностях, допущенных в программе. Например, указывает на объявленные в программе, но не использованные переменные, на отсутствие инициализации переменных, нуждающихся в этом.
   При возникновении ошибки времени выполнения (run-time error) в программе, запущенной из , среда разработки прерывает работу программы и на экране появляется диалоговое окно с сообщением о типе ошибки. Определить причину возникновения исключительной ситуации не сложно, но, исправив ошибку, работу программы приходиться начинать с начала. Например, отсутствие функций приведения типов, неверное задание имен файлов.
   Логические ошибки — это ошибки программиста при разработке алгоритма. Найти их сложнее, чем ошибки, возникающие во время выполнения программы.
2. предоставляет программисту мощное средство поиска ошибок в программе — отладчик. Отладчик среды предоставляет следующие средства:

- трассировка;
- точки останова программы;
- просмотр значений переменных с помощью списка наблюдений;
- прерывание по условию.

Трассировка — это выполнение программы по шагам.

Команда	Действие
Run / Step over (F8)	Выполнение программы по шагам без захода в процедуры.
Run / **Trace into** (F7)	Выполнение программы по шагам без захода в процедуры.
Run / Run (F9)	Отмена трассировки программы — продолжение выполнение программы.
F5 (щелчок по бордюру в области оператора).	Установка точек останова программы.
Run / Program Reset (Ctrl F2)	Остановка выполнения программы.

Обычно разработчику приблизительно известно, в какой подпрограмме или ее части возникает ошибка. По шагам можно выполнить ту часть программы, где скрывается ошибка. С помощью клавиши **F5** помечается строка, в которой произойдет остановка выполнения программы. Такая строка будет помечена красным цветом. На бордюре окна программы появится красный кружок — признак точки прерывания.

Точку останова легче всего добавить щелчком по бордюру в области нужного оператора. Выделение строки снимается точно таким же способом.

После остановки программы можно перейти к выполнению программы по шагам (F8 или F7).

Можно просмотреть значения переменных в момент остановки программы. Для этого нужно навести курсор на имя переменной, значение которой Вас интересует.

3. Когда во время работы программы приходится контролировать большое количество переменных, просматривать их значения можно в специальном окне **Watch List.** Окно вызывается из меню **View,** выбором команды **Debug Windows** (окна отладки) и далее опцией следующего уровня **Watches** (наблюдения).

   Имена переменных, которые необходимо наблюдать при отладке, вводятся в диалоговом окне **Watch Properties,** которое вызывается с помощью команды **Add Watch**. контекстного меню окна **Watch List.**

   В поле **Expression** вводится имя переменной. В поле **Repeat Count** указывается, сколько элементов массива будет отображаться. В нижней части окна **Watch Properties** имеется ряд переключателей, с помощью которых можно указать тип результирующего значения. Если переключатель установлен в **Default**, то это означает, что показываться будет значение, имеющее тот же тип, что и переменная. В поле **Digits** задается число значащих цифр после запятой, которые будут выводиться для дробных чисел. С помощью переключателя **Enable** можно временно отключить контроль значений некоторых переменных списка.
4. Неверное значение можно исправить вручную, если вызвать окно быстрого просмотра значений командой **Run/Evaluate/Modufy.** Команда **Run/ Evaluate/Modufy** вызывает на экран диалоговое окно, где в поле **Expression** вводится вычисляемое выражение, а в поле **Result** (результат) выводится значение этого выражения. В поле **New value** (новое значение) можно ввести новое значение переменной и продолжить расчеты. Для вычисления выполняется действие **Evaluate**, для модификации — **Modufy.**
5. Иногда прерывания в программе нужно осуществлять не каждый раз, а по условию. Для этого точки останова определяются с помощью команды ***Run / Add Breakpoint / Source Breakpoint***. В этом случае выводится диалоговое окно **Add Source Breakpoint**.

   Поле **filename** диалогового окна содержит имя файла программы, куда добавляется точка останова. Поле **Line number** содержит номер строки текста программы, в которую добавляется точка останова. В поле **Condition** записывается условие прерывания. Поле **Pass count** содержит число пропусков выполнения условий перед прерыванием.
6. Список точек останова можно посмотреть, если выполнить команду **View / Debug Windows / Breakpoints.** Контекстное меню этого окна позволяет добавлять новые точки останова или изменять свойства уже включенных в список точек прерывания.
7. Откройте проект Project1 каталога «Тел_Спр_с_ошибкой». В приложение включен алгоритм сортировки базы данных методом «пузырька». В алгоритме допущена ошибка.

   7.1. Запустите приложение. Исправьте синтаксическую ошибку.

   7.2. Определите точку останова в обработчике TForm1.Action4Execute перед оператором for i := 1 To N - 1 Do. Добавьте в диалоговое окно **Watch List** имена переменных X, k и i. Так как

переменные программы доступны только во время выполнения программы, то после имени переменной выводится сообщение: процесс недоступен.

Растяните окно наблюдений на всю ширину экрана. Сделайте так, чтобы окно наблюдений всегда находилось поверх всех остальных окон. Вызовите контекстное меню окна наблюдений и выберите пункт **Stay on Top** (поверх других окон).

7.3. Запустите приложение на выполнение. Загрузите базу данных из файла proba каталога «Тел_спр_ с_ошикой». Выполните сортировку. После остановки программы нажимайте **F8** и наблюдайте за изменением значений переменных в окне наблюдений.

При выполнении оператора x[j] := x[j-1]; x[j-1] := x[j]; два первых значения массива X становятся равными (значение Самойлова). Еще раз нажмите F8 — два первых элемента массива К стали равными 1. Вывод — обмен между элементами массива осуществлен неверно.

7.4. Неверное значение можно исправить вручную, если вызвать окно быстрого просмотра значений. Выполните команду **Run** | **Evaluate/Modufy**

Вы вызвали окно быстрого просмотра значений. В поле **Expression** вводится вычисляемое выражение. Введите X[1]. Перейдите в поле **New value** и введите значение, которое должно было получиться — Поперечная. Нажмите кнопку **Modufy**.

В окне списка наблюдаемых переменных (**Watch List**) значение элемента X[1] изменилось.

Введите в поле **Expression** выражение **к[1]** и нажмите на кнопку **Evaluate**. Перейдите в поле **New value**. Сотрите старое значение величины **к[1],** введите число 2 и нажмите кнопку **Modufy**.

В окне **Watch List** отразилось новое значение элемента**.**

Прервите выполнение программы, нажмите **Ctrl F2,** и закройте окно списка просматриваемых переменных.

Щелчком мыши уберите точку прерывания.

7.5. Установите новую точку прерывания в той строке программы, где происходит обмен элементами массива.

В нашей задаче было полезным прервать выполнение программы именно в тот момент, когда условие **x[j-1] > x[j]** выполняется. Остановившись в этот момент, можно выполнить несколько следующих операторов по шагам и посмотреть, как происходит обмен элементами массива.

Выполните команду **Run | Add Breakpoint | Source Breakpoint**.

Запишите в поле **Condition**

x[j-1] > x[j]

Поле **Pass count** содержит число пропусков выполнения условий перед прерыванием. Нам необходимо, чтобы останов в работе программы произошел при первом выполнении условия. Значение **Pass count** оставляем без изменения.

Запустите программу на выполнение. Просмотрите значения переменных в момент остановки программы путем наведения курсора на имя переменной в листинге программы.

Остановите выполнение программы.

При выполнении оператора присваивания значение одного из элементов массива X теряется. То же самое происходит и с массивом К. Исправьте ошибку в тексте процедуры.

7.6. Просмотрите список точек останова, выполнив команду **View | Debug Windows | Breakpoints.** В диалоговом окне **Breakpoint List** с помощью контекстного меню можно добавлять новые точки останова или изменять свойства уже включенных в список точек прерывания. Закройте **Breakpoint List**.

Часть II. Самостоятельная работа

Рекомендуемое время
60 минут

Используйте возможности отладчика при выполнении задания, полученного на предыдущей лабораторной работе.

1. **Сформируйте контрольные примеры для проверки правильности работы приложения.**
2. **Осуществите трассировку программы, используя клавишу F8. Прервите выполнение программы.**
3. **Осуществите трассировку программы, используя клавишу F7. Прервите выполнение программы.**
4. **Определите точку останова. Запустите программу (F9). После остановки программы продолжите выполнение с помощью трассировки.**

5. **Задайте имена переменных в окне просмотра, выполните программу и проведите трассировку.**
6. **Создайте точку прерывания по условию. Выполните программу.**
7. **Продолжайте отладку до полной уверенности в ее правильной работе.**

Варианты заданий см. в предыдущей работе.

Быстрая разработка приложений. Создание справочной системы приложения с помощью утилиты Help Workshop

КОД: 0641.03.02/ПТ.0641.10

Продолжительность:
80 мин.

Дисциплина:
Технология программирования. Юнита 3.

Предназначено:
Для студентов по направлению ***и***нформатика и ***ВТ*** в соответствии с учебным планом.

Цель: Изучить способ создания справочной системы с помощью утилиты Help Workshop.

Результат обучения:
- умение создавать текст справочного файла;
- умение создавать файл справочной системы;
- использование файла справочной системы в приложении;
- опыт работы в RAD–среде.

Используемые программы:
ИСР – 6.

План занятия:
Часть I. Изучение способа создания файла справочной системы на примере 20 минут

Часть II. Самостоятельная работа **60 минут**

ЗАПУСК ПРОГРАММЫ:
Предполагается, что требуемые программы уже инсталлированы на диске.
(См. «Инструкцию по установке программы на ПК»)

Часть I. Изучение способа создания файла справочной системы на примере

Рекомендуемое время
20 минут

Качественное приложение обеспечивает пользователя справочной системой. Двумя видами справки Вы уже воспользовались — это всплывающие подсказки и диалог «О программе». Кроме того, приложение должно сопровождаться справочным файлом. Справочный файл имеет расширение HLP и автоматически вызывается при нажатии клавиши F1 или при выборе соответствующего пункта меню, например, с названием «Help» или «?».
Создание справочной системы состоит из четырех этапов:

1. Определение номеров контекста справочной системы для управляющих элементов формы.
2. Создание текстового файла с помощью текстового редактора.
3. Создание справочного файла с помощью специального компилятора.
4. Подключение справочных файлов к приложению.

Все четыре этапа выполняются в среде .

1. Каждый управляющий элемент (компонент) формы может иметь номер контекста для справочной системы. Контекст представляет собой число, задаваемое значением свойства **HelpContext**, по значению которого определяется раздел справочной системы, соответствующий данному элементу.

2. Текстовый файл справки создается как документ в формате **RTF**. Для его подготовки можно использовать любой редактор, позволяющий работать с документами этого формата, в том числе текстовый редактор **MicroSoft Word**. Документ может содержать графические изображения и таблицы, стилевое и шрифтовое оформление.
   Текст справочной системы представляется в виде гипертекста — состоит из многих разделов, связанных между собой перекрестными ссылками. Каждый раздел обычно имеет заголовок, отображаемый в верхней части окна просмотра, идентификатор, набор ключевых слов, по которым можно найти раздел, а также ссылки на другие разделы.
   Раздел справочного файла может иметь следующие атрибуты:
   - контекст;
   - заголовок;

- список ключевых слов;
- номер в последовательности просмотра;
- макрокоманда;
- тег компиляции.

Из всех атрибутов обязательным является контекст, представляющий собой уникальный в пределах файла идентификатор раздела. Для обозначения контекста используются русские и латинские буквы, цифры, знак подчеркивания.

Атрибуты назначаются разделу с помощью сносок. Для каждого атрибута справки используется свой знак в сноске. Например, для контекста используется символ #.

2.1. Ограничимся контекстной справкой и создадим справочную систему для нашего приложения.

На первом этапе необходимо создать файл документа справочной системы. Этот файл должен будет иметь расширение **Rtf**.

Создайте каталог **Lab10** в своем каталоге. Вызовите проект, созданный Вами на предыдущей лабораторной работе. Запишите все файлы проекта в каталог **Lab10**.

В меню «Пуск» выполните Программы | **MicroSoft Word.**

В документ Word введите следующий текст:

Содержание

TPageControl

TStringGrid

TComboBox

«**Содержание**» — заголовок раздела справки. Выделите его жирным шрифтом. Установите курсор перед заголовком.

Заголовок раздела справки необходимо пометить сноской. Из **меню** «**Вставка**» выполните команду «**Сноска**». В диалоговом окне «**Сноска**» в группе «Вставить сноску» установите переключатель в положение «**Обычная**», а в группе «**Нумерация**» — в положение «**Другая**». В поле номера сноски введите символ # и нажмите Ok.

В нижней части окна появилось окно сноски. Запишите идентификатор **IDH_00**1. Между символом сноски и текстом должен быть один пробел.

Текст каждого раздела справки должен находиться на отдельной странице документа. В конце раздела справки введите символ «разрыв страницы» (**Ctrl – Enter**).

Запишите текст следующего раздела:

TPageControl

*Компонент **TPageControl** может содержать несколько перекрывающих друг друга панелей класса **TTabSheet**. Каждая панель выбирается связанной с ней закладкой и может содержать свой набор помещаемых на нее компонентов.*

*Чтобы на этапе конструирования добавить новую панель или выбрать ранее вставленную, щелкните по компоненту правой кнопкой мыши и выберите **New page** (новая панель), **Next Page** (следующая панель) или **Previous page** (предыдущая панель). Смена панелей идет циклически, т.е. после показа последней показывается первая и наоборот.*

Выделите заголовок жирным шрифтом (выделите и измените шрифт с помощью пиктограммы Ж.)

Установите курсор перед заголовком. Выполните команду «**Сноска**» из меню «**Вставка**». Установите переключатель в положение «**Обычная**», а в группе «**Нумерация**» — в положение «**Другая**».

В поле номера сноски введите символ # и нажмите Ok. Запишите идентификатор раздела **IDH_002**

В конце текста раздела справки введите символ «разрыв страницы» (**Ctrl – Enter**).

Запишите текст следующего раздела:

TStringGrid

Компонент предназначен для создания таблиц, в ячейках которых располагаются произвольные текстовые строки. Характерные свойства:

Property Cells[ACol, ARow: Integer] — определяет содержимое ячейки с табличными координатами (ACol, ARow).

Property Cols[index: Integer] — содержит все строки колонки с индексом Index.

Property Objects[ACol, ARow: Integer] — обеспечивает доступ к объекту, связанному с ячейкой (ACol, ARow).

Property Rows[index: Integer] — содержит все столбцы ряда с индексом Index.

Установите курсор перед заголовком. Выполните команду «**Сноска**» из меню «**Вставка**». Установите переключатель в положение «**Обычная**», а в группе «**Нумерация**» — в положение «**Другая**».

В поле номера сноски введите символ # и нажмите Ok. Запишите идентификатор раздела **IDH_003**.

В конце раздела справки введите символ «разрыв страницы» **(Ctrl – Enter).**

Запишите текст следующего раздела:

TComboBox

Комбинированный список, представляющий собой комбинацию TListBox и редактора TEdit.

Установите курсор перед заголовком. Выполните команду «**Сноска**» из меню «**Вставка**». Установите переключатель в положение «**Обычная**», а в группе «Нумерация» — в положение «**Другая**». В поле номера сноски введите символ # и нажмите Ok. Запишите идентификатор раздела **IDH_004.** В конце раздела справки введите символ «разрыв страницы» (**Ctrl – Enter**).

2.2. Теперь организуем ссылки на раздел. При использовании ссылки вид курсора при наведении на ключевое слово изменяется. Он превращается во всем нам известную руку.

В первом разделе справки сделаем слово **TPageControl** ссылкой на второй раздел.

Подчеркнем это слово двойной линией и изменим цвет шрифта. Выделите слово **TPageControl**. Выполните команду «**Формат**» | «**Шрифт**». На вкладке «**Шрифт**» выберите в раскрывающемся списке «**Подчеркивание**» вариант «**Двойное**». Цвет шрифта сделайте синим.

Сразу же за словом **TPageControl** нужно поместить идентификатор раздела справки, к которому должен быть выполнен переход. За словом **TPageControl** сразу (без пробела) запишите **IDH_002**. Оформите вставленный идентификатор как скрытый текст.

Для того чтобы два следующих слова сделать ссылками на разделы 3 и 4, проделайте аналогичные действия со словами **TStringGrid** и **TComboBox**. Должно получиться:

#Содержание

TPageControl

TStringGridTComboBox

Сохраните текстовый файл в каталоге **Lab10** под именем **Help1.rtf**. Закройте окно **Word.**

3. Разверните окно . Приступим к созданию справочного файла.

Для использования в приложении текст справки должен быть преобразован из формата **RTF** в формат **HLP**. Процесс преобразования называется компиляцией справочного файла и выполняется с помощью специальных программ. Рассмотрим создание справочного файла с помощью программы **Microsoft Help Workshop** (компилятор справочных файлов). Исполняемый файл **Hcrtf.exe** находится в каталоге **Help/Tools** каталога . Эта утилита может быть включена в инструменты . В этом случае она может быть вызвана командой **Tools/Help Workshop**.

3.1. Основным файлом компилятора является файл проекта, который объединяет такие элементы, как текстовые файлы справки, опции, номера контекстов, и позволяет создать из них справочный файл. В начале работы над новым справочным файлом автоматически создается файл проекта типа HPJ. Элементы проекта указываются в проекте в виде отдельных секций.

Первоначально в проекте имеется только секция Options, в которую включены параметры окна справки и вывода отчета о компиляции проекта.

Вызовите программу Microsoft Help Workshop командой Tools/Help Workshop. В окне утилиты выполните команду **File/New**. В открывшемся диалоговом окне выберите пункт **Help Project** и щелкните по Ok. В окне **Project File Name** выберите папку, где находится файл справки и само приложение (папка «**Lab10**»). Задайте имя файла проекта **Help1** и сохраните.

В диалоговом окне проекта справочной системы вызовите окно опций щелчком по кнопке **Options**. На вкладке **Sorting** в раскрывающемся списке укажите язык справочной системы (русский). На вкладке **Files** в поле **Help File** задано имя файла справки **Help1.Hlp** по имени файла проекта справки, который мы только что начали. Для того чтобы в поле RTFFiles ввести имя файла с содержанием справки, необходимо нажать кнопку **Change**, затем в окне **Topic Files** (Topic — тема) кнопку A**d**d. В диалоговом окне выбрать файл **Help1.rtf**. Закройте окно **Options**.

3.2. Обычно при вызове справки в процессе работы приложения автоматически вызывается раздел справки, соответствующий управляющему элементу, который находится в фокусе ввода. Для организации такого вызова справки нужно установить соответствие между номерами контекстов управляющих элементов приложения и разделами справочной системы. Пронумеруем контексты в соответствии с номерами сноски, а затем в приложении зададим эти номера (свойство **HelpContext**) нужным компонентам. Карта соответствия номеров контекстов и разделов справочной системы указывается в секции **Map** (карта, соответствие).

Вызовите окно **Map** щелчком на кнопке **Map**. Нажмите на **Add**. В поле **Topic ID** окна **Add Map Entry** введите идентификатор раздела справки **IDH_001**, а в поле **Mapped numeric**

value — соответствующее идентификатору число, например 1. Нажмите на Ok. Трижды выполните команду **Add** для определения номеров разделов ID**H_002** (номер 2), ID**H_003** (номер 3) и **IDH_004** (номер 4). Запишите файл проекта справочной системы в свой каталог Lab10.

3.3. Выполните компиляцию проекта справочной системы командой **Compile** из меню File. В диалоговом окне **Compile a Help File** установите флажок: **Include .rtf filename and topic ID in Help** file (включить имя файла **RTF** и номера разделов в файл справки). Нажмите на кнопку «**Compile**».

После компиляции выводится информация о числе разделов, гиперссылок, ключевых слов и рисунков. Закройте окно утилиты. В диалоговом окне подтвердите намерение сохранить файл проекта справки.

4. Для того чтобы подключить справку к приложению, нужно вызвать окно установки опций проекта **Project**/**Options** и на странице **Application** выбрать этот файл в поле **Help file** с помощью кнопки **Browse**. Подключите файл **Help1**.**hlp** к проекту.

   Номер раздела справки каждого компонента, которое вызывается клавишей F1, вводят в свойство **HelpContext**. Перейдите в окно инспектора объектов. Раскройте список компонентов и, выделяя последовательно компоненты в списке, установите номер раздела справки в свойстве **HelpContext**.

   Для компонента **TPageControl** введите 2, для **TStringGrid** — 3, **для TComboBox** — 4. Для всех остальных компонентов значение свойства задайте равным 1.

   В списке компонентов выделите **компонент Form1**. На странице **Properties** найдите свойство **HelpFile** и введите имя файла справки — **Help1.hlp**.

5. В хранилище действий **TActionList** добавьте еще одно действие, в **свойство Caption** которого введите слово «**Помощь**». Создайте обработчик Execute этого действия. В этом обработчике событий обратимся к функции **WinHelp**, которая запускает программу **Windows Help**. Введите оператор

```
WinHelp(Form1.Handle,'Help1.Hlp',HELP_CONTEXT,1);
```

6. Создайте новый пункт главного меню и назначьте ему соответственное свойство Action для вызова справки.
7. Запустите программу на выполнение и проверьте работоспособность справочной системы.

Если нужно дополнить справку, то: 1) файл текста справки дополняется; 2) справочная система компилируется заново.

Часть II. Самостоятельная работа

Рекомендуемое время
60 минут

Продолжите разработку справочной системы приложения:

1. **Снабдите компоненты приложения всплывающими подсказками (свойства Hint для компонентов и ShowHint для формы).**
2. **Дополните справочную систему контекстными справками по всем компонентам приложения и пунктам меню.**

Быстрая разработка приложений. Создание дистрибутива приложения

КОД: 0641.03.02/ПТ.0641.11

Продолжительность:
80 мин.

Дисциплина:
Технология программирования. Юнита 3.

Предназначено:
Для студентов по направлению ***и***нформатика и ***ВТ*** в соответствии с учебным планом.

Цель: Познакомиться с приложением InstallShield

Результат обучения:
- знание основных этапов процесса создания инсталляционной копии;
- создание инсталляционной версии учебной программы.

Используемые программы:
ИСР – 6. InstallShield Express v. 4.0

План занятия:
Часть I. Установка и развертывание приложений **30 минут**

Часть II. Самостоятельная работа **50 минут**

ЗАПУСК ПРОГРАММЫ:
Предполагается, что требуемые программы уже инсталлированы на диске.
(См. «Инструкцию по установке программы на ПК»)

Часть I. Установка и развертывание приложений

Рекомендуемое время
30 минут

Процесс создания готового продукта не завершается на этапе написания работоспособной программы. Приложение, как правило, должно быть установлено у пользователя или группы пользователей, в локальной сети или Интернете. Достаточно обширная категория пользователей работает с Windows на уровне значков и папок. Для таких категорий пользователей имеется неофициальный стандарт на установку новых приложений, например с компакт–дисков. Для этого используются инсталляционные программы (обычно Setup.exe), которые используют привычные в Windows Мастера. Они задают в процессе установки несколько уточняющих вопросов (например, о каталоге, в который должна быть помещена программа) и затем выполняют все действия по инсталляции приложения.

1. Настройка коммерческой версии приложения

Перед созданием инсталляционной копии программы необходимо сформировать версию, ориентированную на массовое использование. Для этого надо дать команду **Project/Options** и выполнить ряд настроек.

1.1. Вкладка **Application**.

В поле **Title** вводится заголовок приложения, привязываемый к его значку. После установки приложения он отображается в папках Windows, в Главном меню, при переключении между запущенными программами с помощью комбинации ALT + TAB и других местах. В поле **Help file** указывается имя файла справки, в раздел **Icon** с помощью кнопки **Load Icon** загружается значок приложения.

1.2. Вкладка **Compile**.

С помощью этой вкладки обеспечивается максимальное быстродействие программного кода. Отладочная информация не генерируется. Все флажки на панелях Runtime errors (ошибки времени выполнения) и Debugging (отладка) должны быть сброшены. При этом отключается контроль ошибок времени выполнения: выхода индекса за границы массива, переполнения, ошибок ввода/вывода и др.

1.3. Вкладка **Packages** (пакеты).

Размер Exe–файла зависит от того, включен или нет флажок Build with runtime packages (создать программу, использующую пакеты времени выполнения). Если этот переключатель включен, то размер файла значительно сокращается в связи с тем, что к программе не подключаются различные пакеты системы : библиотеки стандартных

функций, классов, визуальных компонентов, модулей, ответственных за работу BDE и серверов COM, и другие. Создать программу в таком режиме можно, только если разработчик уверен, что у пользователя эти пакеты уже установлены.

Понятно, что надежнее перед созданием Exe–файла этот флажок снимать.

1.4. Вкладка **Version Info** (информация о версии)

Если установлен флажок **Include version information in project** (включить в проект сведения о версии), то в программу помещается информация о текущей версии. Она доступна, если пользователь выберет значок программы, установленной в системе Windows, и в контекстном меню укажет пункт **Свойства**. В случае установки этого флажка доступны следующие поля:

- **M**

  ajor version number (главный номер версии). Обычно меняется, когда программа переделывается полностью или в нее вносятся принципиальные улучшения.
- **M**

  inor version number (вспомогательный номер версии). Обычно определяет внесение в программу менее существенных изменений, связанных с повышением эффективности или расширением функциональных возможностей.
- **R**

  elease version number (номер версии выпуска). Обычно фиксирует незначительные изменения в программе: исправление обнаруженных ошибок, усовершенствование пользовательского интерфейса и пр.
- В

  раскрывающемся списке **Language** (язык) выбирается язык, на котором оформлен пользовательский интерфейс, в списке **Key/Value** — различная дополнительная информация о проекте (названия продукта, компании, авторские и имущественные права и прочие сведения).

1.5. Вкладка **Directories/Conditionals** (каталоги и настройки компиляции).

Здесь задаются каталоги для компиляции, сборки и получения готовых версий программы, а также дополнительные директивы компилятора.

В завершение надо закрыть окно **Project Options** (параметры проекта) и выполнить полную перестройку проекта командой **Project/Build** (проект/построить).

2. Приложение **InstallShield**

Сравнительно крупные программы хранят в Реестре (или в файлах .INI) большое число собственных настроек. Работать с Реестром вручную, выполнять множество рутинных действий при установке и удалении приложения весьма трудоемко. Для создания инсталляционной копии программы, которая автоматически внесет в Реестр все настройки, а при удалении ликвидирует их, в поставку включается программа **InstallShield**. При распространении приложений, созданных с помощью системы лучше всего применять именно это приложение. Это связано с тем, что оно входит в число немногих программ, официально сертифицированных корпорацией Borland для создания инсталляционных копий, способных автоматически устанавливать на компьютеры пользователей механизм BDE и другие важнейшие части приложений, правильно вносит регистрационную информацию в Реестр.

Если программа **InstallShield** установлена, то ее значок в меню Пуск/Программы расположен отдельно от значка . После ее запуска выбирается либо пункт Create a new project (создание нового проекта), либо пункт Open a project (открытие существующего проекта). После этого появляется окно настройки параметров. Все параметры разделены на шесть групп:

- определение общих настроек инсталляционного проекта;
- определение файлов, входящих в инсталляционный пакет;
- определение настроек для целевой платформы, где будет работать приложение;
- описание процесса инсталляции и сопроводительная информация;
- требования к программе инсталляции;
- создание и тестирование инсталляционной программы.

Разделы настройки процесса инсталляции:

General Information	Общая информация о продукте: его название, название компании, различная контактная информация, значок программы версия продукта и т.д.
Features	В данном пункте задаются специфические характеристики процесса установки. С их помощью можно определить, какие части приложения будут устанавливаться в зависимости от выбора пользователя. Имеется возможность организации такого выбора в иерархическом виде. **Features** — особенности, свойства.
Setup Types	Определяются типы установки (типичная поставка, минимальная поставка или в зависимости от настроек пользователя).
Upgrade Path	С помощью данного пункта можно задать способ обновления ранее установленных версий данного продукта.
Specify Application Data	В данном разделе выбираются файлы, входящие в приложение (**Files**). В подразделе **Files and Features** задаются дополнительные характеристики этих файлов. В пункте **Objects/Merge Modules** можно добавлять к проекту стандартные объекты и библиотеки, входящие в поставку и Windows. В пункте **Dependencies** выполняется полный анализ взаимосвязей между всеми входящими в проект файлами на наличие повторов, противоречий, отсутствия необходимых объектов и т.д.
Configure the Target System	Задаются каталоги (**Shortcuts/Folders)** целевой системы, в которой будет устанавливаться приложение. Пункт **Registry** позволяет с помощью визуального редактора определить, какие изменения будут вноситься в системный Реестр, пункт **ODBC Resources** с помощью подобного редактора позволяет легко настроить драйверы **ODBC**. В пункте **INI File Changes** задаются изменения, вносимые в файлы инициализации, как стандартные, так и используемые самой программой. С помощью пункта **File Extentions** можно сформировать взаимосвязи между расширениями устанавливаемых в операционной системе файлов и приложением, которое будет обрабатывать эти файлы по умолчанию. Пункт **Environment Variable** позволяет дополнить список переменных среды окружения — это требуется некоторым программам для корректного функционирования.
Customize the Setup	На шаге **Dialogs** определяется, какие диалоговые окна отображаются в ходе инсталляции и какую информацию они выдают пользователю. В пункте **Billboards** можно задать всевозможные экранные заставки, сопровождающие процесс инсталляции. Пункт **Text and Message** позволяет вынести текстовые сообщения в отдельный модуль, чтобы не привязываться к конкретному функциональному языку и в дальнейшем иметь возможность быстро настраивать процесс инсталляции на конкретную локализованную версию.
Define Setup Requirements and Actions	В разделе **Requirements** можно определить конкретные требования, которые устанавливаемая программа предъявляет к операционной системе. Это может быть тип этой системы, ее версия, процессор, объем оперативной памяти, разрешение экрана и т.п. В некоторых случаях возможностей данной инсталляционной программы недостаточно для установки сложных приложений. Тогда можно воспользоваться разделом **Custom Actions.** Он определяет дополнительные пользовательские действия и файлы (**Support Files**).

Prepare for Release	В пункте **Build Your Release** задается конкретный физический носитель, на котором планируется поместить инсталляционную программу. В зависимости от типа такого носителя можно дополнительно указать требуемый объем, способ сжатия, стандартные средства для синхронизации программ установки, входящих в ядро Windows и т.д. Тестирование созданной инсталляционной копии выполняется на шаге **Test Your Release** в полном соответствии с выбранным типом целевого физического носителя. На последнем этапе **Distribute Your Release** созданный ранее образ инсталляционного приложения переносится на выбранный физический носитель, например, CD–Rom или DVD.

Часть II. Самостоятельная работа

Рекомендуемое время
50 минут

1. Вызовите проект, созданный Вами на предыдущей лабораторной работе. Запишите проект в новый каталог Lab11 Вашего подкаталога. Скопируйте также файлы помощи.
2. Вызовите диалоговое окно опций проекта (**Project/Options**) и осуществите настройку коммерческой версии Вашего приложения
    a. На вкладке **Application** уточните заголовок приложения, значок приложения и файл справки. Если Вас все удовлетворяет, то можно все настройки оставить без изменения.
    b. На вкладке **Compile** уберите все флажки на панелях **Runtime errors** и **Debugging** для обеспечения максимального быстродействия программного кода.
    c. На вкладке **Packages** уберите флажок **Build with runtime packages**, чтобы включить в приложение все библиотеки .
    d. На вкладке **Version Info** установите информацию о версии приложения, название продукта и другую информацию по Вашему выбору.
3. Выполнить перестройку проекта командой **Project/Build**.
4. Запустите приложение **InstallShield**.
5. Вызовите файл помощи и ознакомьтесь с этапами создания дистрибутива приложения.
6. Для расположения проекта по умолчанию создается папка **My Setup** в каталоге **Мои документы.** Все созданные файлы, такие как файл проекта (.ism), установочный пакет (.msi файлы), располагаются в этом каталоге. Можно расположить файлы проекта в любом другом каталоге. Создайте в Вашем каталоге Lab11 два подкаталога: **Инсталляционный проект** и **Установка**. В подкаталог **Установка** осуществим пробную установку программы.
7. Создайте новый инсталляционный проект — **Create a new project**. Определите язык, используемый в инсталляционном проекте. Определите местонахождение Вашего инсталляционного проекта в окне **Project Name and Location** — нажмите кнопку **Browse,** найдите Ваш каталог «**Инсталляционный проект**» и запишите проект под именем Telefon.ism. Нажмите на кнопку **Create**.
8. Задайте общие настройки инсталляционного пакета. В подгруппе настроек **General Information** заполните список, расположенный справа. Введите:
    a. имя автора,
    b. комментарий,
    c. название программы (Product Name),
    d. пиктограмму приложения (по желанию),
    e. версию,
    f. обязательно определите строку **INSTALLDIR**, в которой определяется каталог для установки Вашего приложения (укажите специально созданный для этого каталог «**Установка**»),
    g. имя компании разработчика,
    h. контактные адреса и телефоны,
    i. увеличьте размер шрифта.

    В строке **Product Name** Вы должны записать имя Вашего приложения, под которым оно будет известно системе и пользователям.
9. Раздел **Features** оставьте без изменения, так как этот раздел определяет несколько вариантов установки. Вы будете выполнять типовую установку без вариантов.

10. В подгруппе **Setup Types** оставьте только типовую установку.
11. Определите файлы, которые будут входить в инсталляционный пакет. В разделе настроек **Specify Application Data** выберите подпункт **Files**. В правой части окна откроются два дерева навигации файлов. В верхнем дереве (Source computers folders) выделяются файлы, которые необходимо поместить в инсталляционный пакет. В нижнем дереве (Destination computers folders) необходимо создать каталог для этих файлов. Вызовите контекстное меню, выберите пункт **Show Predefined Folder** и далее [**INSTALLDIR**]. Переместите выделенные файлы из верхнего окна в созданный каталог методом Drag&Drop.
12. Третий этап построения (Configure the Target System) пропустите, так как ошибки этого этапа могут сказаться на работоспособности учебного класса.
13. На четвертом этапе **Customize the Setup** определите, какие окна будут отображаться при инсталляции. Стандартный набор диалоговых окон уже определен. Добавьте диалоговое окно **Readme**. Для изменения содержания этого диалогового окна создайте файл в формате .rtf с помощью редактора WordPad, а затем подключите этот файл в строке **Readme File** в окне **Readme Property**. Таким же способом измените содержимое диалогового окна **License Agreement**.
14. Создайте заставку для диалоговых окон. Выделите пункт **Billboards**. В правой части экрана из контекстного меню выполните пункт **New Billboard ins**. Переименуйте созданную заставку, а затем определите ее свойства: **Effect, Background Color, Title**.
15. Требования к операционной системе определять не нужно, поэтому пятый раздел настроек (**Define Setup Requirements and Actions**) можно пропустить.
16. Выделите пункт **Build Your Release** из раздела **Prepare for Release** и определите состав инсталляционного пакета. Выберите минимальный состав **Custom**. Создайте установочный пакет, выполнив команду **Build Custom** из пункта меню **Build** (или клавиша **F7**). Можно нажать на пиктограмму **Build** на панели инструментов. В нижней части окна **Installshild** посмотрите результаты компиляции файла проекта. Компиляция должна пройти без ошибок. Далее проверьте работу пакета — выберите **Test Your Release** или пиктограмму **Test**.
17. Запишите файл **Setup**.**exe** в каталог Lab11. Выполните пункт **Distribute Your Release**. В правой части найдите каталог Lab11, используя кнопку **Browse**, и нажмите **Distribute to Location**.
18. Закройте приложение **InstallShield Express**.
19. Вызовите утилиту «Установка и удаление программ». Установите приложение, используя созданный Вами файл **Setup.exe**. Этот Файл должен быть расположен в каталоге Lab11\Установка.
20. Вызовите утилиту «Установка и удаление программ». В списке установленных программ найдите Ваше приложение и удалите его.