

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Саратовский государственный технический университет
имени Гагарина Ю.А.»

Филиал федерального государственного бюджетного образовательного
учреждения высшего образования
«Саратовский государственный технический университет
имени Гагарина Ю.А.» в г. Петровске

УТВЕРЖДАЮ

Директор филиала СГТУ

имени Гагарина Ю.А. в г. Петровске

Е.А.Бесшапошникова

«06» июня 2024 г.



МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

по дисциплине

ОП.04 «Основы алгоритмизации и программирования»

направление подготовки

09.02.07 «Информационные системы и программирование»

Методические указания рассмотрены
на заседании предметной (цикловой)
комиссии общепрофессиональных дисциплин,
профессиональных модулей специальностей
технического профиля
«14» июня 2024года, протокол №12

Председатель ПЦК Табарова /Ю.А.Табарова/

Петровск 2024

Пояснительная записка

Методические указания по выполнению лабораторных работ подготовлены на основе рабочей программы учебной дисциплины ОП.04 «Основы алгоритмизации и программирования», разработанной на основе ФГОС СПО по специальности 09.02.07 «Информационные системы и программирование» и соответствующих общих (ОК) и профессиональных (ПК) компетенций:

ОК 01 - Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам,

ОК 02 -Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности.

ОК 04 - Эффективно взаимодействовать и работать в коллективе и команде.

ОК 05 - Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 09 - Пользоваться профессиональной документацией на государственном и иностранном языках.

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием.

ПК.1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4. Выполнять тестирование программных модулей.

ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

При выполнении лабораторных работ студент должен *знать*:

- понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции;
- эволюцию языков программирования, их классификацию, понятие системы программирования;
- основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти;
- подпрограммы, составление библиотек подпрограмм;

объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов, инкапсуляция и полиморфизма, наследования и переопределения

При выполнении лабораторных работ студент должен *уметь*:

- разрабатывать алгоритмы для конкретных задач;
- использовать программы для графического отображения алгоритмов;
- определять сложность работы алгоритмов;
- работать в среде программирования;
- реализовывать построенные алгоритмы в виде программ на конкретном языке программирования;
- оформлять код программы в соответствии со стандартом кодирования;
- выполнять проверку, отладку кода программы.

Содержание лабораторных работ определено рабочей программой и тематическим планированием, соответствует теоретическому материалу изучаемых разделов учебной дисциплины.

Объем лабораторных работ по дисциплине определяется учебным планом по данной специальности.

Продолжительность лабораторной работы – 2 академических часа. Перед проведением лабораторной работы преподавателем организуется инструктаж, а по ее окончании – обсуждение итогов.

Комплект методических указаний по выполнению лабораторных работ по дисциплине ОП.04 «Основы алгоритмизации и программирования» содержит 3 лабораторных работы.

**Перечень лабораторных работ
по дисциплине ОП.04 «Основы алгоритмизации и программирования»**

ЛАБОРАТОРНАЯ РАБОТА № 1.

Тема: Изучение интегрированной среды разработчика

ЛАБОРАТОРНАЯ РАБОТА № 2.

Тема: Разработка интерфейса приложения

ЛАБОРАТОРНАЯ РАБОТА № 3.

Тема: Тестирование, отладка приложения

ЛАБОРАТОРНАЯ РАБОТА № 1

Тема: Изучение интегрированной среды разработчика

Цель работы: изучить интегрированную среду разработки Delphi; получить практические навыки создания современного программного обеспечения: научиться работать с дизайнером форм, редактором кода, инспектором объектов, а также создавать примитивные Windows-приложения.

Оборудование: ПК, интернет, программное обеспечение – MS Word, среда программирования Borland Delphi 7, инструкции по выполнению работы

Справочный материал:

Запускается Delphi обычным образом, т. е. выбором из меню Borland Delphi 7 команды Delphi 7 (рис. 1).

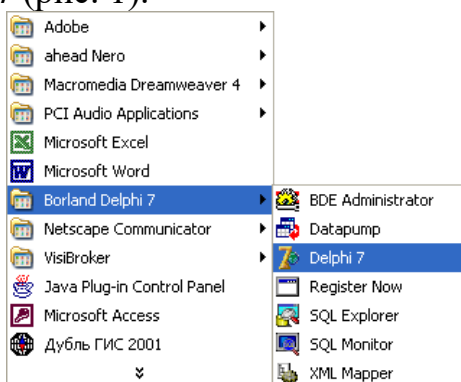


Рис. 1. Запуск Delphi

Вид экрана после запуска Delphi несколько необычен (рис. 2). Вместо одного окна на экране появляются пять:

- главное окно – *Delphi 7*;
- окно стартовой формы – *Form 1*;
- окно редактора свойств объектов – *Object Inspector*;
- окно просмотра списка объектов – *Object TreeView*;
- окно редактора кода – *Unit1.pas*.

Окно редактора кода почти полностью закрыто окном стартовой формы.

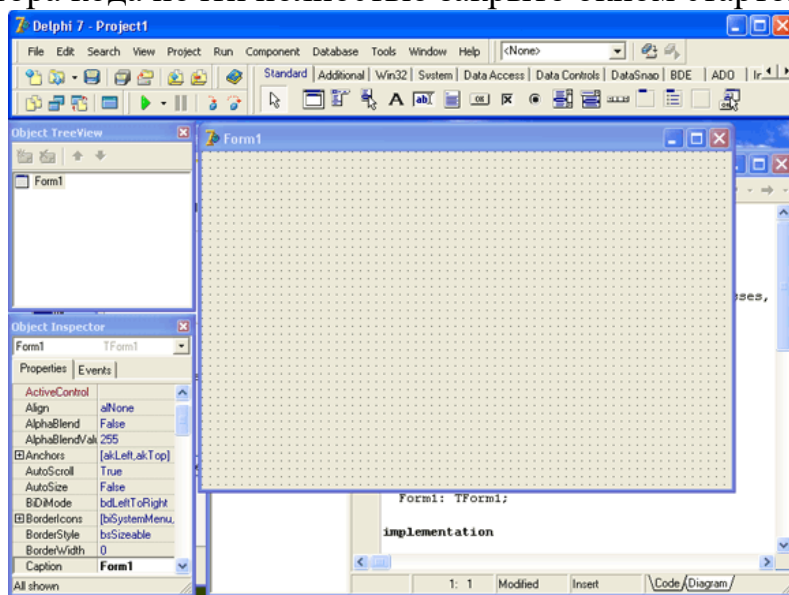


Рис. 2. Вид экрана после запуска Delphi

В главном окне (рис. 3) находится меню команд, панели инструментов и палитра компонентов.

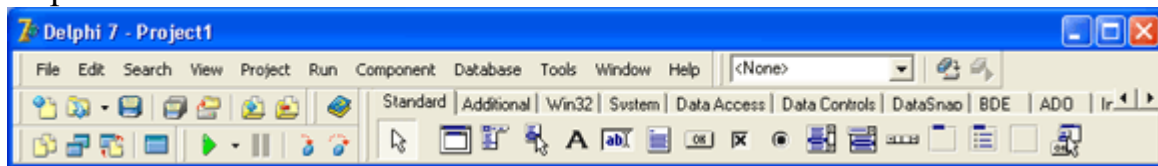


Рис. 3. Главное окно

Ниже полосы главного меню расположены две инструментальные панели. Левая панель содержит два ряда быстрых кнопок, дублирующих некоторые наиболее часто используемые команды меню. Правая панель содержит палитру компонентов библиотеки визуальных компонентов. Палитра компонентов содержит ряд страниц, закладки которых видны в ее верхней части. Правее полосы главного меню размещена еще одна небольшая инструментальная панель, содержащая выпадающий список и две быстрые кнопки. Это панель сохранения и выбора различных конфигураций окна ИСР.

Полоса главного меню

Разделы меню File (файл) позволяет создать новый проект, новую форму, фрейм, модуль данных, открыть ранее созданный проект или форму, сохранить проекты или формы в файлах с заданными именами.

Разделы меню Edit (правка, редактирование) позволяют выполнять обычные для приложений Windows операции обмена с буфером, а также дают возможность выравнивать группы размещенных на форме компонентов по размерам и месторасположению.

Разделы меню Search (поиск) позволяют осуществлять в коде разрабатываемого приложения поиск и контекстные замены, которые свойственны большинству известных тестовых редакторов.

Разделы меню View (просмотр) позволяют вызывать на экран различные окна необходимые для проектирования.

Разделы меню Project позволяют добавлять и убирать из проекта формы, задавать опции проекта, компилировать проект без его выполнения и делать много других полезных операций.

Разделы меню Run (выполнение) дает возможность выполнять проект в нормальном или отладочном режимах, продвигаясь по шагам, останавливаясь в указанных точках кода, просматривая значения переменных и т.д.

Меню Component (компонент) позволяет создавать и устанавливать новые компоненты, конфигурировать палитру компонентов, работать с пакетами packages.

Разделы меню DataBase (база данных) позволяет использовать инструментарий для работы с базами данных.

Меню Tools (инструментарий) включает ряд разделов, позволяющих выполнять настройки ИСР и вызывать различные вспомогательные программы.

Меню Windows (окно) имеется только начиная с 6 версии. Разделы этого меню позволяют ориентироваться среди массы окон, обычно одновременно открытых в процессе проектирования и переключаться в нужное окно.

Меню Help (справка) содержит разделы, помогающие работать со встроенной в Delphi справочной системой.

В основной половине окна слева располагаются Дерево Объектов (Object tree View), под ним – Инспектор Объектов (Object Inspector). Окно Дерево Объектов отображает иерархическую связь визуальных и не визуальных компонентов и объектов разрабатываемого приложения. Инспектор Объектов – это основной инструмент, с помощью которого можно задавать свойства компонентов и обработчики событий.

Окно Object Inspector (рис. 4) – окно редактора свойств объектов предназначено для редактирования значений свойств объектов. В терминологии визуального проектирования объекты – это диалоговые окна и элементы управления (поля ввода и вывода, командные кнопки, переключатели и др.). Свойства объекта – это характеристики, определяющие вид, положение и поведение объекта.

Окно Инспектора Объектов имеет две страницы. В верхней части окна имеется выпадающий список всех компонентов, размещенных на форме. В нем можно выбрать тот компонент, свойства и события которые вас интересуют.

Страница свойств (Properties) Инспектора Объектов показывает свойства того объекта, который в данный момент выделен.

Страница событий (Events) показывает все события, на которые может реагировать выбранный объект.

Например, свойства *Width* и *Height* задают размер (ширину и высоту) формы, свойства *Top* и *Left* – положение формы на экране, свойство *Caption* – текст заголовка.

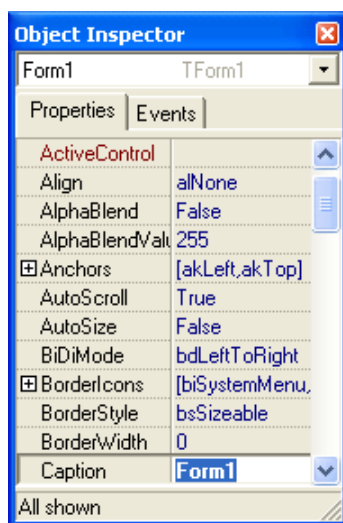


Рис. 4. На вкладке Properties перечислены свойства объекта и указаны их значения

Правее этих окон располагается окно пустой формы, готовой для переноса на нее компонентов.

Окно стартовой формы (Form1) представляет собой заготовку главного окна разрабатываемого приложения.

Стартовая форма создается путем изменения значений свойств формы *Form1* и добавления к форме необходимых компонентов (полей ввода и вывода текста, командных кнопок).

Свойства формы определяют ее внешний вид: размер, положение на экране, текст заголовка, вид рамки.

В окне редактора кода (рис. 5), которое можно увидеть, отодвинув в сторону окно формы, следует набирать текст программы. В начале работы над новым проектом это окно редактора кода содержит сформированный Delphi шаблон программы.

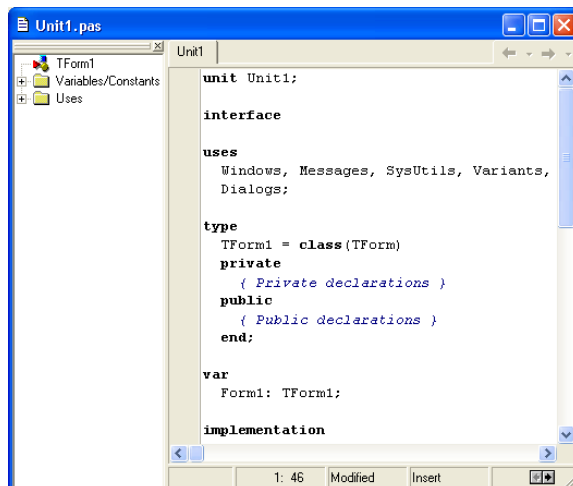


Рис. 5. Окно редактора кода

В нижней части окна Редактора Кода располагается строка состояния. В самой левой ее позиции находится индикатор строки и колонки, который помогает определить в каком месте кода вы находитесь. Второй элемент строки состояния - индикатор модификации, который указывает, были ли сделаны изменения в коде. Третий элемент строки состояния - индикатор режима вставки, который показывает, будут ли вводимые символы вставляться в текст или писаться поверх текста. Переключение режима вставки производится клавишей Insert.

Порядок выполнения работы:

Задание. Разработать приложение, используя которое, можно вычислить скорость, с которой спортсмен пробежал дистанцию.

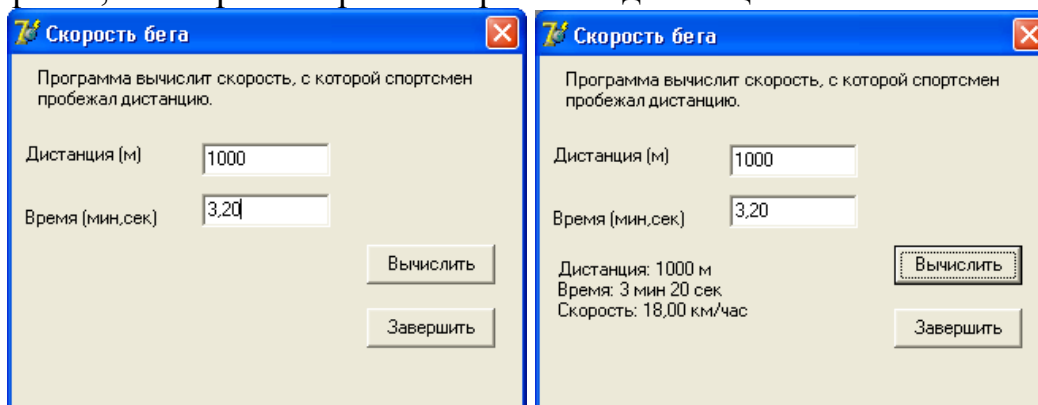


Рис. 6. Окно программы вычисления скорости бега

1. Для начала работы над новой программой запустите Delphi. Если вы уже работаете в среде разработки и у вас загружен другой проект, выберите в меню *File* (Файл) команду *New > Application* (Создать > Приложение).

2. В окне *Object Inspector* установить следующие свойства для стартовой формы:

Свойство	Значение
Caption	Скорость бега
Height	250
Width	330
BorderStyle	bsSingle
BorderIcons .biMinimize	False
BorderIcons .biMaximize	False
Font. Size	10

3.Добавить 2 компонента *Edit1* и *Edit2* - поля ввода-редактирования текста. Компонент Edit1 предназначен для ввода длины дистанции, Edit2 – для ввода времени. Для компонента Edit1 установить указанные свойства, а для компонента Edit2 аналогично самостоятельно:

Свойство	Компонент	
	Edit1	Edit2
Text		
Top	56	
Left	128	
Height	21	
Width	121	

4.В форму надо добавить четыре компонента *Label*. Первое поле предназначено для вывода информационного сообщения, второе и третье – для вывода информации о назначении полей ввода, четвертое поле – для вывода результата расчета (скорости).

Установить следующие свойства:

Компонент	Свойство	Значение
Label1	AutoSize	False
	Wordwrap	True
	Caption	Программа вычислит скорость, с которой спортсмен пробежал дистанцию
	Top	8
	Left	8
	Height	33
	Width	209
Label2	Top	56
	Left	8

Компонент	Свойство	Значение
	Caption	Дистанция (метров)
Label3	Top	88
	Left	8
	Caption	Время (минуты, секунды)
Label4	AutoSize	False
	Wordwrap	True
	Top	120
Label 4	Left	8
	Height	41
	Width	273

5. Добавить в форму две командные кнопки: *Вычислить* и *Завершить*. После добавления к форме двух командных кнопок нужно установить значения их свойств:

Свойство	Компонент	
	Button1	Button2
Caption	Вычислить	
Top	176	
Left	16	
Height	25	
Width	75	

6. Окончательный вид формы разрабатываемого приложения приведен на рис. 7.

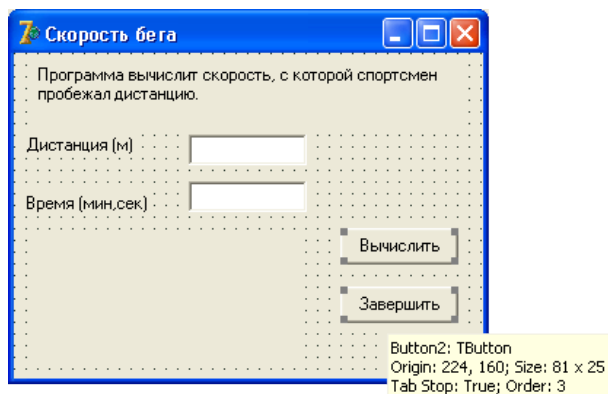


Рис. 7. Форма программы Скорость бега

7. В редакторе кода для кнопок необходимо прописать процедуры обработки событий. Процедура обработки события OnClick на кнопке Button1 (Вычислить).

// нажатие кнопки Вычислить

procedure TForm1.Button1Click(Sender: TObject);

```

var
  dist: integer; // дистанция, метров
  t: real; // время как дробное число
  min: integer; // время, минуты
  sek: integer; // время, секунды
  v: real; // скорость
begin
  // получить исходные данные из полей ввода
  dist := StrToInt(Edit1.Text); t := StrToFloat(Edit2.Text);
  // предварительные преобразования
  min := Trunc(t); // кол-во минут – это целая часть числа t
  sek := Trunc(t*100) mod 100;
  // кол-во секунд – это дробная часть числа t
  // вычисление
  v := (dist/1000) / ((min*60 + sek)/3600);
  // вывод результата
  label4.Caption := 'Дистанция: ' + Edit1.Text + ' м' + #13 + 'Время: ' +
    IntToStr(min) + ' мин ' + IntToStr(sek) + ' сек ' + #13 + 'Скорость: ' +
    FloatToStrF(v,ffFixed,4,2) + ' км/час';
end;

```

Процедура обработки события Onclick на кнопке Button2 (Завершить).

```

// нажатие кнопки Завершить
procedure TForm1.Button2Click(Sender: TObject);
begin
  Form1.Close; // закрыть главное окно программы
end;

```

8. Чтобы сохранить проект, нужно из меню *File* выбрать команду *Save Project As*. Если проект еще ни разу не был сохранен, то Delphi сначала предложит сохранить модуль (содержимое окна редактора кода), поэтому на экране появится окно *Save Unit1 As*. В этом окне (рис. 8) надо выбрать папку, предназначенную для файлов проекта, и ввести имя модуля. После нажатия кнопки *Сохранить*, появляется следующее окно (рис. 32), в котором необходимо ввести имя файла проекта.

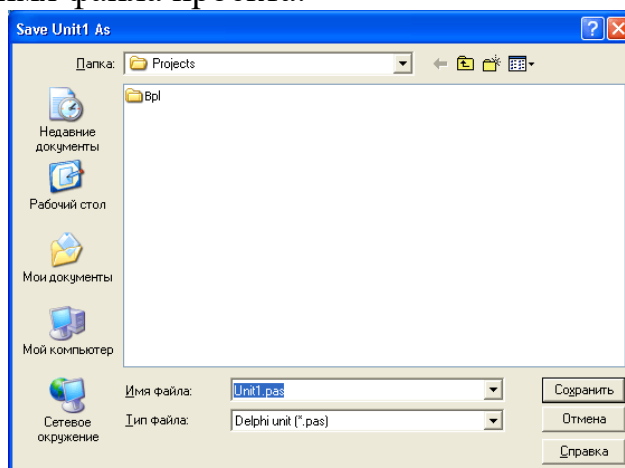


Рис. 8. Сохранение модуля формы

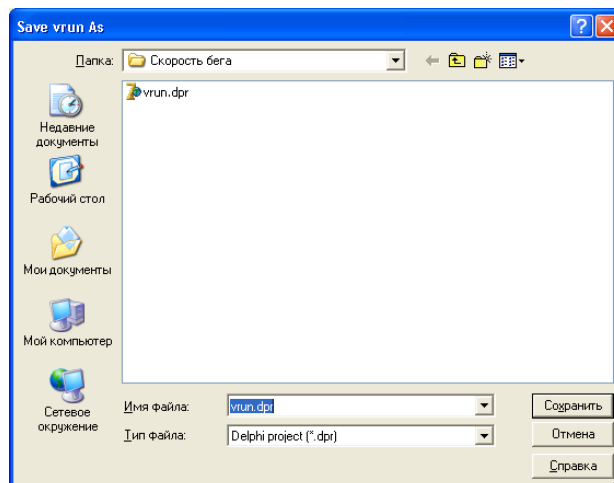


Рис. 9. Сохранение проекта

Обратите внимание на то, имена файлов модуля (pas-файл) и проекта (dpr-файл) должны быть разными. Имя генерируемого компилятором исполняемого файла совпадает с именем проекта. Поэтому файлу проекта следует присвоить такое имя, которое, по вашему мнению, должен иметь исполняемый файл программы, а файлу модуля – какое-либо другое имя, например, полученное путем добавления к имени файла проекта порядкового номера модуля.

Тема: Разработка интерфейса приложения

Цель работы: получить навыки работы в визуальной среде программирования Delphi, усвоить применение компонентов Button, Edit, Label.

Оборудование: ПК, интернет, программное обеспечение – MS Word, среда программирования Borland Delphi 7, инструкции по выполнению работы

Справочный материал:

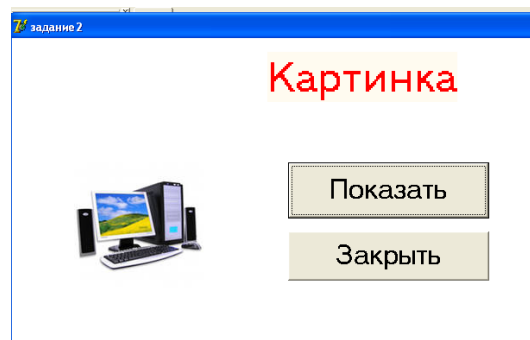
Создание интерфейса пользователя начинается с наполнения формы различными компонентами. Разместив на форме компоненты, нужно установить их свойства. Имена и заголовки компонентов должны быть информативными, т.е. такими, чтобы прочитав их, можно было понять их значение. Например, по умолчанию первая кнопка формы имеет вид Button1. Такое же значение имеет и ее заголовок. Вы должны обязательно изменить это имя, потому, что другому программисту оно ничего не говорит, а пользователь, глядя на такой заголовок, тем более ничего не узнает о назначении кнопки.

Порядок разработки интерфейса:

1. Интерфейс должен быть таким, чтобы пользователь мог легко понять назначение его компонентов
2. Внешний вид интерфейса должен быть приятным
3. Избегайте информационной перегрузки интерфейса. На нем не должно быть одновременно представлено слишком много информации или элементов управления. Если пользователь должен вводить много информации попытайтесь применить меню, несколько форм или файл данных
4. Направление потока информации и вводимых данных должно быть естественным – слева на право или сверху вниз
5. Интерфейс должен «вести» пользователя по этапам ввода данных.

Порядок выполнения работы:

Задание 1. Разработать приложение с интерфейсом, показанным на рисунке:



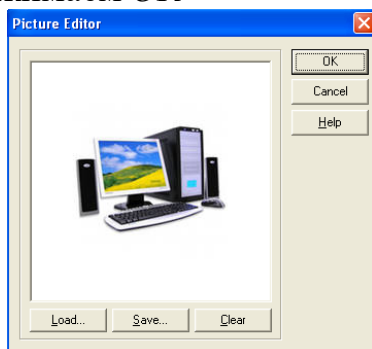
1. Запустите Delphi и создайте новый проект. Сохраните его в вашей папке под названием *Интерфейс*.
2. В окне инспектора объектов *Object Inspector* измените свойства объекта «Форма»: Name – Группа; Caption – задание2; Color – clScrollBar
3. Поместите на форму компонент *Label* (надпись). Поэкспериментируйте с размещением надписи на форме, ее размером. Измените свойства объекта надпись: Name – lblPic; Caption – Пустая строка (имя не вводим); Font: Шрифт – Arial, Размер 1 – 36, Начертание – жирный, Цвет – красный

4. Поместите на форму компонент *Button* (кнопка). Измените свойства кнопки: Name – btnMyButton1; Caption – Щелкни меня; Font: Шрифт – Arial, Размер – 32, Начертание – жирный, Цвет – по Вашему усмотрению, Left – 320, Top – 144, Height – 65, Visible – False, Width – 233

5. Поместите на форму компонент *Button2* (кнопка). Измените свойства кнопки: Name – btnMyButton2; Caption – Закрыть; Font: Шрифт – Arial, Размер – 32, Начертание – жирный, Цвет – по Вашему усмотрению, Left – 320, Top – 244, Height – 57, Visible – True, Width – 233

6. На форму поместите компонент *Image* (Рисунок), который находится на вкладке *Additional* (Стандартные) палитры компонентов. Измените свойства рисунка: Name – Imgpic; Caption – Пустая строка (имя не вводим); Font: Шрифт – Arial, Размер – 32, Начертание – жирный, Цвет – по Вашему усмотрению, Left – 72, Top – 144, Height – 145, Visible – False, Width – 201.

7. В окне инспектора объектов *Object Inspector* открываем вкладку *Picture*. В открывшемся окне. Используя вкладку *Load* выбираем нужный рисунок и нажимаем *OK*



8. Приступите к созданию исходного кода. Создайте обработчик события *OnClick* для кнопок:

Чтобы нажатие кнопки приводило к каким-либо действиям, необходимо изменить содержимое обработчика события. Задайте появление текста «Картинка» и рисунка при нажатии кнопки «Показать». Для этого в окне редактора кода введите:

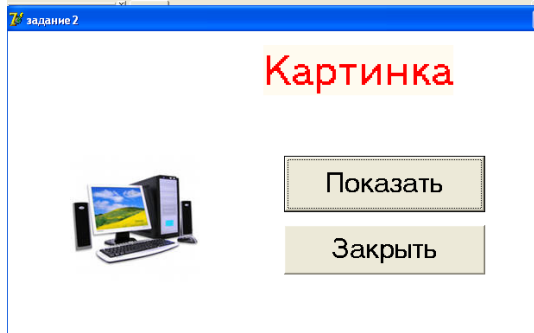
```
implementation
{$R *.dfm}
procedure TfrmГруппа. btnMyButton1 Click(Sender: TObject);
begin
    ImgPic.Visible:=True;
    Lblpic.Caption:='Картинка'
end;
end.
```

Чтобы задать закрытие формы при нажатии кнопки закрыть нужно ввести в редакторе кода:

```
implementation
{$R *.dfm}
procedure TfrmГруппа. btnMyButton2.Click(Sender: TObject);
begin
```

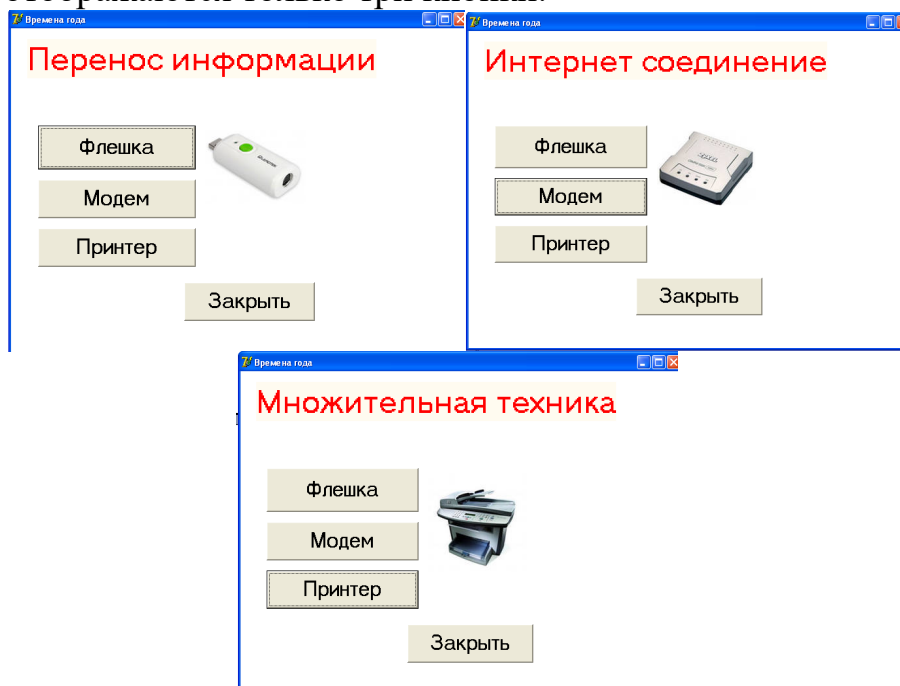
```
Close;  
end;  
end.
```

9.Выполните программу, щелкнув *Run* (выполнить) на панели *Отладка* или нажав *F9* . При этом на экране появится созданная Вами форма.



10.Проверьте правильность работы кнопки. Закройте приложение.

Задание 2.Написать программу, которая по нажатию кнопки показывает картинку и ее описание. Картинки прилагаются в папке. После запуска программы отображаются только три кнопки.



Цель работы: изучить методы тестирования и отладки на практическом примере.

Оборудование: ПК, интернет, программное обеспечение – MS Word, среда программирования Borland Delphi 7, инструкции по выполнению работы

Справочный материал:

1) Методы тестирования и отладки

Тестирование — процесс выполнения программ с целью обнаружения факта наличия ошибок.

Отладка — процесс локализации и устранения ошибок.

Процессы тестирования и отладки схематически могут быть представлены следующим образом:



Тестирование начинается с разработки множества тестов и их исполнения на основе одной из выбранных методик. Подготовка дополнительных тестов потребуется при недостаточной полноте тестирования, невозможности локализовать проблему с помощью имеющихся тестов и необходимости выполнить контроль сделанного исправления.

Существуют две основные стратегии тестирования:

Тестирование программы как черного ящика, при котором программа рассматривается как объект, внутренняя структура которого неизвестна.

Тестирование программы как прозрачного (белого) ящика подразумевает знание исходного кода программы и полный доступ к нему.

При тестировании по типу «черного ящика» тесты демонстрируют:

- как выполняются функции программы;
- как принимаются исходные данные;
- как вырабатываются результаты;
- как сохраняется целостность внешней информации.

При тестировании «черного ящика» рассматриваются системные характеристики программ, игнорируется их внутренняя логическая структура.

Для тестирования программ методом «черного ящика» готовятся определенные группы тестов.

– Для тестирования классов эквивалентностей. Классы эквивалентности позволяют вместо большого количества тестов использовать лишь их небольшое подмножество. Каждый тест представляет набор тестов, на которых программа ведет себя одинаково. Существует два типа классов эквивалентностей:

- Класс корректных тестовых случаев, отражающих типичную «нормальную» ситуацию.
 - Класс тестов, содержащих ненормальную ситуацию, т.е. описывающих ситуацию, которой быть не должно.
- Для тестирования граничных значений.
- Для анализа причинно–следственных связей.
- Для тестирования тех утверждений, которые приводятся в документации.

При тестировании по типу «белого ящика» исследуются внутренние элементы и связи между ними. Объектом тестирования является не внешнее, а внутреннее поведение программы. Проверяется корректность построения всех элементов программы и правильность их взаимодействия друг с другом. Тестирование по принципу «белого ящика» характеризуется степенью, в какой тесты выполняют или покрывают логику (исходный текст) программы.

Наиболее важный принцип, относящийся к тестированию программ, состоит в том, чтобы думать об этой стадии еще на этапе написания программы. Следует постоянно задаваться вопросом: как будет тестироваться данный сегмент? Если ответ на вопрос о способе тестирования программы неясен, она должна быть либо переписана заново, либо разбита на модули.

Для составления тестов используются следующие источники: справочники; вычисления вручную; использование результатов, полученных при помощи другой программы.

Поскольку в процессе разработки приходится тестировать еще не завершенную программу, все подходы делятся на две группы.

Тестирование сверху вниз. Применяется, если программа разрабатывается сверху вниз. В данном случае используются «заглушки» — фрагменты кода, имитирующие еще не написанные части программы.

Тестирование снизу вверх. При этом, как правило, дополнительно должна быть создана программа — «драйвер», организующая взаимодействие уже написанных модулей.

Если процесс тестирования показал, что программа работает неправильно, то начинается процесс отладки. В процессе отладки локализуется ошибка.

Отладка программы — процесс творческий и плохо формализуемый. Тем не менее, основной идее отладки можно придать вид следующего алгоритма:

1. Следует начать с изучения уже доступных исходных и результирующих данных.
2. Сформулировать некоторую гипотезу, которая объясняет получение таких результирующих данных.

3. Подготовить новые исходные данные и провести эксперимент, который позволит доказать или опровергнуть гипотезу.

Для отладки программ в инструментальные среды программирования встраиваются специальные отладчики.

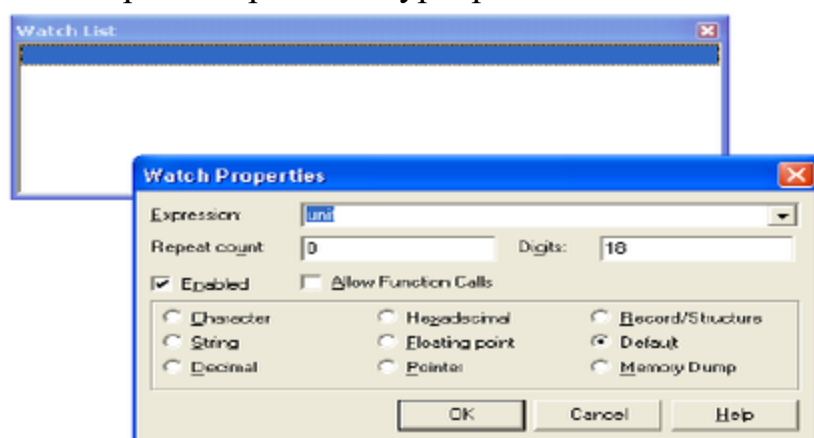
2) Средства отладки Delphi

Отладка — это локализация ошибки и ее исправление. Для этого используются точки прерывания, пошаговое выполнение программы и просмотр ряда переменных на различных шагах выполнения программы.

Окно наблюдения

Наблюдать за состоянием переменной или выражения можно с помощью специального окна, вызываемого опцией *View | Debug windows | Watches*.

Окно наблюдения используется в отладочном режиме для наблюдения за изменением значений выражений, помещенных в это окно. Для добавления нового выражения щелкните по окну правой кнопкой мыши и выберите опцию *New Watch*. В строке *Expression* введите выражение. Окно *Repeat count* определяет количество показываемых элементов массивов данных; окно *Digits* указывает количество значащих цифр для отображения вещественных данных; переключатель *Enabled* разрешает или запрещает вычисление выражения. Остальные элементы определяют вид представления значения. В последних версиях Delphi вы можете просмотреть в отладочном режиме текущее значение любой переменной, если укажете на нее курсором: значение появится в ярлычке рядом с курсором.

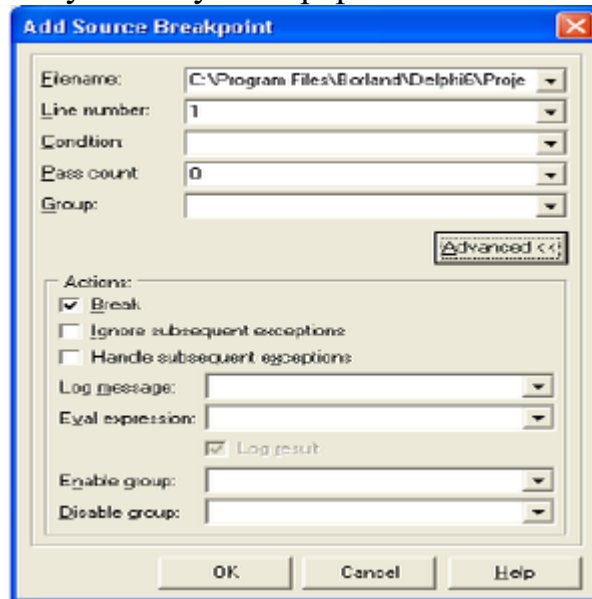


Точка контрольного останова определяет оператор в программе, перед выполнением которого программа прервет свою работу и управление будет передано среде Delphi. Точка останова задается с помощью опции *view | Debug windows | Breakpoints*.

Окно точек останова содержит список всех установленных в проекте точек, перед выполнением которых происходит прекращение работы программы и управление получает среда Delphi.

Для добавления новой точки следует щелкнуть по окну правой кнопкой мыши и выбрать опцию *Add*. В этом случае появляется окно, с помощью которого можно указать положение добавляемой точки: *FileName* - определяет имя файла; *Line number* - номер строки от начала файла (в момент появления окна оно содержит файл и строку с текстовым курсором). В строке *Condition*

можно указать условие останова в виде логического выражения (например, *MyValue = Max-Value-12*), а в строке *Pass count* - количество проходов программы через контрольную точку без прерывания вычислений.



В нижней части окна имеется панель *Actions*, с помощью которой и определяются действия для точки останова, указанной в верхней части окна.

Break - простой останов перед выполнением помеченного оператора.

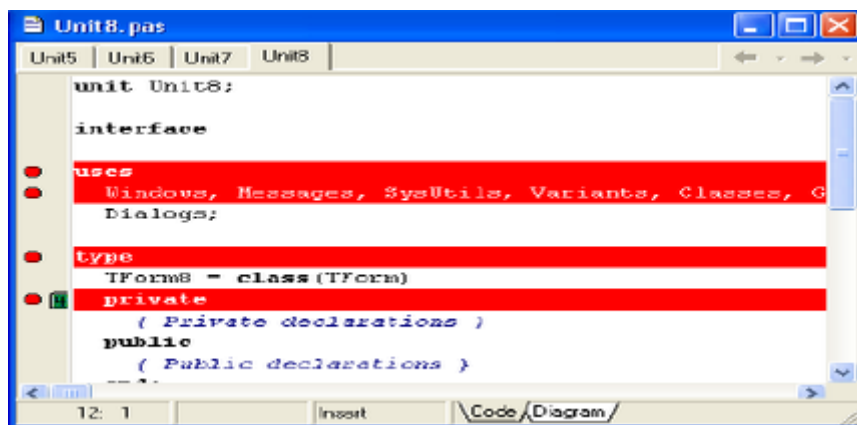
ignore subsequent exceptions - если переключатель установлен, игнорируются все возможные последующие исключения в текущем отладочном сеансе до очередной точки останова, в которой, возможно, это действие будет отменено.

Handle subsequent exceptions - после установки этого переключателя отменяется действие предыдущего переключателя и возобновляется обработка возможных исключений.

С помощью *Log message* вы можете указать произвольное сообщение, связанное с точкой останова, а с помощью *Eval expression* - вычислить некоторое выражение и поместить его результат в это сообщение.

Трассировка программы

Перед исполнением оператора, в котором установлена точка контрольного останова, работа программы будет прервана, управление получит среда Delphi, а в окне наблюдения отразится текущее значение наблюдаемых переменных и/или выражений. Теперь программист может проследивать работу программы по шагам с помощью клавиш *F7* и *F8* или инструментальных кнопок. При нажатии *F8* будут выполнены запрограммированные в текущей строке действия, и работа программы прервется перед выполнением следующей строки текста программы. Контрольная точка останова выделяется по умолчанию красным цветом, а текущая прослеживаемая строка - синим. Если программа остановлена в контрольной точке, т.е. когда текущая строка совпадает со строкой останова, строка выделяется красным цветом. Признаком текущей строки является особое выделение строки в служебной зоне слева в окне редактора.



Кстати, чтобы установить/снять точку контрольного останова, достаточно щелкнуть мышью по служебной зоне слева от нужной строки или установить в эту строку текстовый курсор и нажать *F5*.

При нажатии *F7* среда выполняет те же действия, что и при нажатии *F8*, однако, если в текущей строке содержится вызов подпрограммы пользователя, программа прервет свою работу перед выполнением первого исполняемого оператора этой подпрограммы, т.е. клавиша *F7* позволяет проследивать работу вызываемых подпрограмм.

После трассировки нужного фрагмента программы можно продолжить нормальную ее работу, нажав клавишу *F9*.

Порядок выполнения работы:

Задание 1. Разработайте в среде Delphi 7 проект в соответствии с индивидуальным заданием по варианту.

Задание 2. Разработайте план тестирования вашего программного комплекса. При подготовке контрольных тестов воспользуйтесь вышеописанными рекомендациями. Тесты должны отображать:

1. типичную ситуацию;
2. ненормальную ситуацию;
3. граничные значения;
4. затрагивающие причинно–следственные связи.

Одним из тестов является контрольный пример из раздела «Постановка задачи».

Задание 3. Воспользуйтесь опцией *Go to Cursor* для перехода в режим отладки. При остановке выполнения программы добавьте в окно просмотра (*Add Watch*) наименования нескольких интересующих вас переменных. Вызовите окно просмотра (*Watch*). Далее выполняйте программу по шагам (*F7* или *F8*). В окне просмотра можно наблюдать интересующие вас переменные. Если окно просмотра по какой-то причине исчезло с экрана, в него можно перейти с помощью клавиши *F6*. Для прекращения работы программы следует нажать комбинацию клавиш *Ctrl F2* или выполнить опцию *Program Reset*. Можно продолжить выполнение программы, нажав *Ctrl F9*.

Задание 4. Установите в программе несколько контрольных точек и запустите программу на выполнение. Просмотрите значения интересующих вас переменных с помощью окна просмотра, так же как это было предложено в предыдущем пункте.

Задание 5. Воспользуйтесь опцией *Go to Cursor* для перехода в режим отладки. При остановке выполнения программы откройте окно *Evaluate/modify* и просмотрите значения интересующих Вас переменных. Измените значение какой-нибудь переменной, записав новое в окно *New Value*.

Задание 6. Выполните такое же задания, используя точки останова (*Breakpoint*).

Задание 7. Ознакомьтесь с процессом выполнения программы при запуске ее с помощью опций *Trace Into (F7)* и *Step Over (F8)*.

Задание 8. Продолжите отладку разрабатываемой программы, используя навыки работы с отладчиком.

1. Дорохова, Т. Ю. Основы алгоритмизации и программирования: учебное пособие для СПО / Т. Ю. Дорохова, И. Е. Ильина. — Саратов, Москва: Профобразование, Ай Пи Ар Медиа, 2022. — 139 с. — ISBN 978-5-4488-1531-7, 978-5-4497-1718-4. — Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. — URL: <https://profspo.ru/books/122426>
2. Золин, А. Г. Программирование на C++: учебное пособие для СПО / А. Г. Золин, А. Е. Колоденкова, Е. А. Халикова. — Саратов: Профобразование, 2022. — 126 с. — ISBN 978-5-4488-1439-6. — Текст: электронный // ЭБС PROФобразование: [сайт]. — URL: <https://profspo.ru/books/116283>
3. Лебеденко, Л. Ф. Технологии программирования: учебно-методическое для СПО / Л. Ф. Лебеденко, О. И. Моренкова. — Саратов: Профобразование, 2021. — 108 с. — ISBN 978-5-4488-1204-0. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/106637>
4. Моренкова, О. И. Программирование на языке C/C++: практикум для СПО / О. И. Моренкова, Т. И. Парначева. — Саратов: Профобразование, 2021. — 102 с. — ISBN 978-5-4488-1192-0. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/106631>

Дополнительные учебные издания:

5. Кудинов, Ю. И. Основы алгоритмизации и программирования: учебное пособие для СПО / Ю. И. Кудинов, А. Ю. Келина. — 2-е изд. — Липецк, Саратов: Липецкий государственный технический университет, Профобразование, 2020. — 71 с. — ISBN 978-5-88247-956-4, 978-5-4488-0757-2. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/92834>
6. Токманцев, Т. Б. Алгоритмические языки и программирование: учебное пособие для СПО / Т. Б. Токманцев; под редакцией В. Б. Костоусова. — 2-е изд. — Саратов, Екатеринбург: Профобразование, Уральский федеральный университет, 2019. — 102 с. — ISBN 978-5-4488-0510-3, 978-5-7996-2899-4. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/87785>

Электронно-библиотечная система:

7. ЭБС «IPRbooks», ООО «Ай Пи Ар Медиа»

8. ЭБС «PROФобразование»

9. ЭБС «Book.ru»