Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Брянский государственный аграрный университет»

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Чемисов Н.Н.

Высокоуровневые методы информатики и программирования

Учебно-методическое пособие

Брянская область
2016

УДК 681.3:657(07)

Чемисов Н.Н. Высокоуровневые методы информатики и программирования. Учебно-методическое пособие.- Брянск: Издательство Брянский ГАУ, 2016.- 55 с.

В пособии собраны лабораторные задания по дисциплине «Высокоуровневые методы информатики и программирования».

Пособие предназначено для студентов направления подготовки Прикладная информатика профиль Прикладная информатика в экономике, а также студентов различных направлений высших учебных заведений, имеющих практические навыки работы с языками программирования.

Рекомендовано к изданию решением методической комиссии экономического факультета от 30.06.2016г., протокол №9.

Рецензент: к.э.н., доцент кафедры информационных систем и технологий Лысенкова С.Н.

Введение

В настоящее время разрабатываются и совершенствуются новые высокоуровневые методы создания программного обеспечения, использующие объектно-ориентированное компонентное программирование на основе обработки событий, графического интерфейса и взаимодействия с базами данных. Программирование все больше становится не просто кодированием алгоритмов, а процессом построения моделей, решаемых задач. Такой поход позволяет ускорить разработку сложного и надежного программного обеспечения. Для реализации такого подхода к программированию разными компаниями разрабатываются новые платформы, включающие наборы технологий и инструментов. Одним из наиболее новых и полных реализаций данного подхода является платформа .Net компании Microsoft. В этой платформе реализованы последние достижения высокоуровневых методов информатики и программирования. Изучение данной платформы позволит быстро и эффективно разрабатывать современное программное обеспечение для семейства операционных систем Windows, которые установлены на более чем 90% всех персональных компьютеров в мире. Кроме этого, изучение данной платформы позволит освоить новые методы построения и разработки программ, которые реализованы и в других платформах и технологиях. Целью данного пособия является обучение студентов базовым методам разработки современных прикладных программ (приложений) с использованием языка Visual Basic 2010 и библиотеки классов платформы .Net. Разработка таких приложений в значительной степени автоматизирована за счет использования системы разработки Visual Studio, которая облегчает процесс разработки, но при работе с ней нужны базовые знания языка программирования и способы решения разного типа задач.

Представленные материалы имеют целью формирование компетенций и освоение обучающимися видов профессиональной деятельности в соответствии с ФГОС ВО и ОПОП ВО по направлению подготовки 09.03.03 Прикладная информатика (уровень бакалавриата).

Лабораторная работа 1
Освоение среды разработки приложений MS Visual Basic 2010 Express

Запустить MS Visual Basic 2010 Express, выбрать пункт Создать проект…, затем Приложение Windows Form

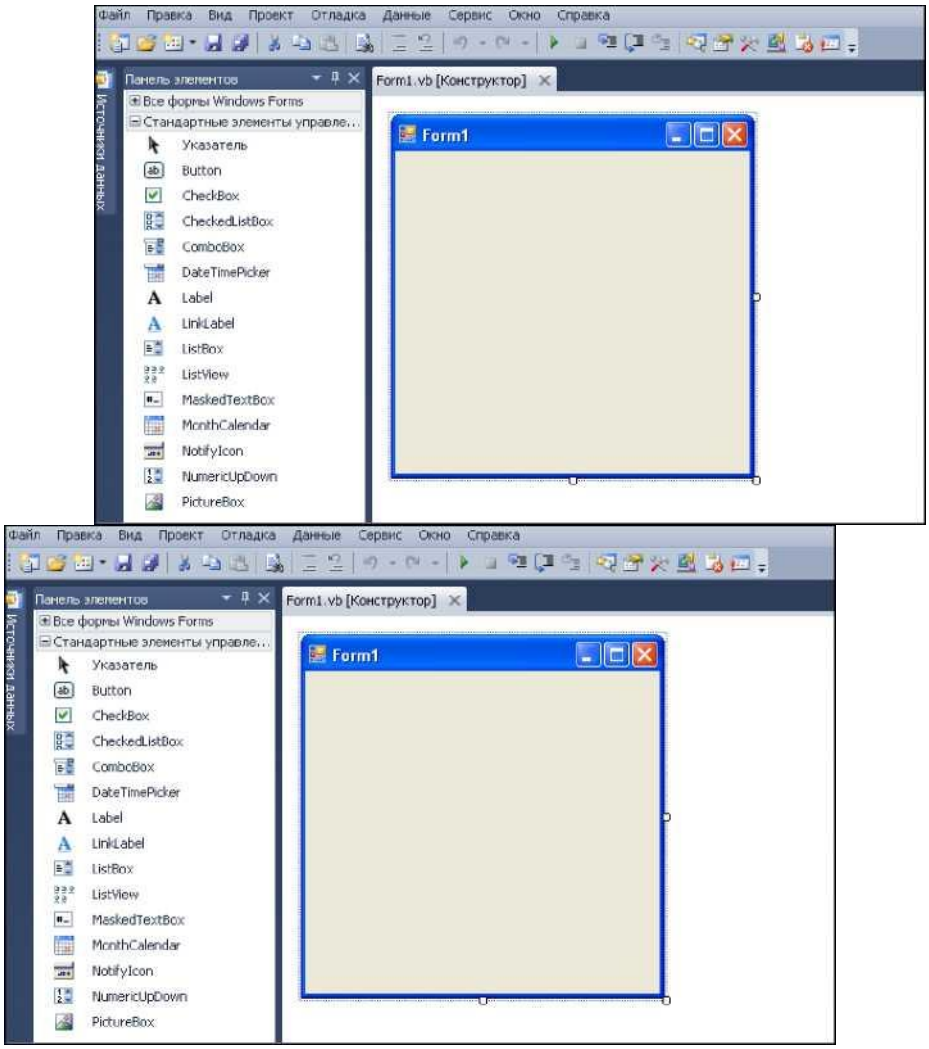
Упражнение 1. Конструктор форм.

Порядок работы:

В центре экрана расположено окно конструктора форм. Именно в этой рабочей области происходит визуальное конструирование макета формы и расположенных на ней элементов.

В среде Visual Basic здесь выводится либо изображение формы, либо окно программы (об окне программы будет рассказано в следующем разделе).

Обратите внимание на небольшие белые квадратики, расположенные в центре каждой стороны. Они называются маркерами размеров; перетаскивая их мышью, можно изменять размер формы.



Если вы хотите, чтобы размеры формы превышали размер окна конструктора форм, перетащите ее край через окно свойств и окно проекта - Visual Basic изменит размеры формы в соответствии с вашими требованиями. Часть формы будет скрыта окном свойств и окном проекта, и для работы с ней можно воспользоваться полосами прокрутки.

Упражнение 2. Редактирование свойств формы.

Порядок работы:

Редактирование свойств формы осуществляется в окне Свойства, которое имеет две кнопки: Алфавит и Категории. Редактировать свойства можно в любой из них. Рабочая область окна выглядит как таблица, состоящая из двух колонок: в первой определяется наименование свойства, а во второй его значение. Свойства формы могут быть следующих типов: логические свойства (могут принимать только значения True или False); свойства с фиксированным набором значений (допустимые значения ограничиваются некоторым списком); строковые свойства (содержат текстовые значения); Шестнадцатеричные свойства (определяют цвет); файловые свойства (значения представляют собой ссылки на файлы); свойства размера.

Логические свойства:

1. Если свойство может принимать только значения True или False, его можно изменить двойным щелчком на имени в первом столбце окна свойств. Задайте свойству MaximizeBox (возвращает и позволяет задать значение, определяющее наличие кнопки развёртывания окна в титульной строке окна формы.) формы Form1 значение False.
2. Запустите приложение щелчком по кнопке Начать отладку * - обратите внимание, что с вашей формы исчезла кнопка развертывания (в правом верхнем углу). Это бывает полезно, если вы не хотите, чтобы ваше приложение могло занимать весь экран.

Чтобы завершить работу программы кнопку Остановить отладку ■ на ПИ, или щелкните на значке [x] в правом верхнем углу формы Form1. В любом случае вы вернетесь в режим конструирования.

Свойства с фиксированным набором значений:

Если допустимые значения свойства ограничиваются некоторым списком (который называется перечислением), то двойные щелчки на имени свойства будут приводить к последовательному перебору всех допустимых значений. Если их количество велико, вероятно, быстрее будет выбрать нужное значение из раскрывающегося списка во втором столбце. Работу с такими свойствами можно продемонстрировать на примере свойства FormB orderStyl e (стиль рамки) формы Form1:

1. Щелкните на форме Form1, чтобы активизировать ее.
2. В окне свойств щелкните на кнопке со стрелкой справа от свойства FormBorderStyle. В открывшемся списке перечислены допустимые значения этого свойства:
   - Значение None удаляет рамку вокруг формы. Чаще всего применяется в заставках.
   - Значение Fixed Single создает тонкую рамку и запрещает произвольное изменение размеров окна (Однако при этом размеры окна можно изменять кнопками развертывания и восстановления).
   - По умолчанию свойство FormB orderStyle имеет значение Sizable. Оно применяется в тех случаях, когда пользователю разрешается изменять размеры окна.
   - Если по какой-либо причине вы не хотите, чтобы пользователь менял размеры диалогового окна, свойству FormB orderStyl e следует присвоить значение Fixed Dialog. В окнах сообщений, которые часто встречаются в среде Windows, используется именно этот тип рамки.
   - Наконец, если вы создаете "плавающую" панель инструментов, также называемую палитрой, свойству можно присвоить значение FixedToolWindow или SizableToolWindow, в зависимости от того, какая панель вам нужна.
3. Задайте для свойства FormBorderStyle значение Fixed Dialog.
4. Выполните команду Начать отладку, чтобы проверить работу формы. Обратите внимание - на ней отсутствуют кнопки изменения размера. Кроме того, вы не можете изменить размеры формы перетаскиванием ее границ.
5. После завершения работы с формой нажмите на ней кнопку [x].

Строковые свойства

Два самых распространенных свойства Name и Text - являются строковыми. Если значение свойства необходимо ввести с клавиатуры, то вместо того, чтобы щелкать во втором столбце, следует дважды щелкнуть на имени свойства. В этом случае во втором столбце выделяется текущее значение свойства (если оно существует), и вы можете просто набрать новый текст без предварительного удаления старого текста клавишами Delete или Backspace. Завершив ввод, желательно щелкнуть на форме или нажать клавишу Enter - при этом введенное значение предохраняется от нежелательных изменений, вызванных случайно нажатыми клавишами.

1. Щелкните на форме, чтобы сделать ее активным объектом.
2. В окне свойств дважды щелкните на свойстве Name (Имя) (оно находится в верхней части списка). При этом выделяется текущее значение этого свойства, Form1.
3. Введите текст frmMain. Нажмите клавишу Enter, чтобы задать для свойства Имя значение frmMain.
4. Дважды щелкните на свойстве Text; при этом выделяется текущее значение свойства, текст Form1.

Введите с клавиатуры новое значение поля Text - отображаемое для пользователя в режиме выполнения. Проследите изменения в рабочей области.

Шестнадцатеричные свойства:

Двойной щелчок на имени свойства (например, BackColor и остальные свойства, определяющие цвета) в первом столбце открывает диалоговое окно с двумя вкладками. На вкладке Палитра находится палитра, из которой можно выбрать нужный цвет вместо того, чтобы вводить его шестнадцатеричный код. Вкладка Система позволяет выбрать цвет на основании цветовой схемы, заданной в панели управления.

Изменение цвета фона формы:

1. Дважды щелкните на имени свойства BackColor в окне свойств.
2. Перейдите на вкладку Другой, чтобы вывести цветовую палитру и назначьте форме красный цвет фона.
3. Снова дважды щелкните на имени свойства BackColor и перейдите на вкладку Система.
4. Если выбрать Window, форма станет белой. Выберите из списка строку Button Face. Поскольку сейчас в Windows используются в основном объемные управляющие элементы, цвет фона формы может совпадать с цветом кнопок. В нашем случае это будет серый цвет с имитацией объема.

Файловые свойства

Чтобы выбрать нужный файл, дважды щелкните на имени свойства. Например, свойство Icon формы определяет значок, отображаемый при свертывании окна программы во время выполнения. В Visual Basic имеется достаточно значков, из которых можно выбрать нужный. Присвоение значка форме происходит следующим образом:

1. Дважды щелкните на имени свойства Icon формы frmMain. Открывается диалоговое окно Открыть.
2. Выберите значок - файл с расширением *.ico. Нажмите кнопку Открыть. Теперь в заголовке приложения появился новый значок.
3. Чтобы сбросить файловое свойство и вернуть ему значение (None), щелкните на втором столбце и нажмите клавишу Delete.

Свойства размера

Свойства Width (ширина формы) и Height (высота формы) задаются в свойстве Size, а Left (расстояние от левого края до формы), Top (расстояние от верха до края формы) - в свойстве Radding.

1. Выбрать свойство Height (высота формы) и установить значение 6000.

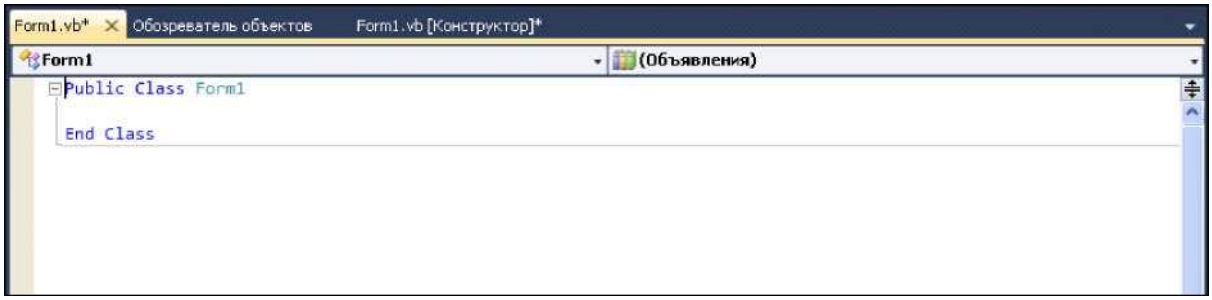
2. Изменить ЛКМ размеры формы и проследить изменения координат на ПИ и в окне свойств.

Упражнение 3. Окно просмотра объектов.
Порядок работы:

В окне просмотра объектов отображаются различные свойства, события и методы, которые были сделаны доступными для вас. Окно просмотра объектов вызывается клавишей F2.

Чтобы открыть окно программы, можно дважды щелкнуть на форме или элементе в окне макета формы.

Если двойной щелчок был сделан на элементе, загружается процедура элемента. Если двойной щелчок был сделан на форме, будет загружена процедура этой формы. Окно программы также открывается через меню Вид/Код.



После открытия окна программы вы можете перейти к любой процедуре любого объекта, находящегося на выделенной форме.

Упражнение 4. Старт, остановка, сохранение проекта.
Порядок работы:

1. стартовать проект (меню Отладка - Начать отладку, кнопка Начать отладку на ПИ, клавиша F5),
2. остановить выполнение проекта (меню Отладка - Остановить отладку, кнопка Остановить отладку на ПИ),
3. сохранить проект (меню Файл - Сохранить Form1 как).

Лабораторная работа 2.

Создание формы и ее элементов Упражнение 1. Создание формы

Порядок работы:

1.Создать 3 формы основными способами создания:

1 способ. Добавление нового объекта с помощью меню Проект пункта Добавить форму.

2. способ. На панели инструментов кнопка Добавить форму Windows.

2. Удалить последний объект из проекта: щелкнуть ПКМ по названию объекта в окне Группа проектов и в контекстном меню выбрать Удалить.

3. Произвести обязательное присвоение следующим свойствам формы (табл. 1):

Таблица 1

Свойства формы	Значение	Описание
Name (Имя)	frmMain	имя формы (принято к имени форм добавлять префикс “frm”)
Text	First form	заголовок формы
StartUpPosition	Center Screen	положение формы на экране при запуске приложения (в центре экрана)

4. Присвоить следующим свойствам предлагаемые значения и проследить изменения с формой (табл. 2)

Свойства формы	Значение	Описание
WindowsState	Maximized	увеличение размера формы до размера экрана после старта проекта
FormBorderStyle Text	None ММ	отсутствует граница формы и ее заголовок
BorderStyle Text Height Width	FixedSingle First Height +100 Width +100	граница формы появилась и ее можно изменять появился заголовок высота и ширина формы увеличилась на 100
BackColor	По желанию	цвет фона формы изменился
MinimizeBox	False	отсутствует кнопка свернуть в правой части заголовка

Упражнение 2. Основные управляющие элементы формы

Порядок работы:

1. Метка. Выполнить двойной щелчок ЛКМ на пиктограмме A (Label - метка) панели элементов управления.
2. Присвоить следующим свойствам появившейся метки предлагаемые значения (табл.)

Таблица 3

Свойства	Значение	Описание
Name	lblText	имя метки
Text	"Как прекрасен этот мир!"	текст, который располагается на метке
Font	шрифт Arial, курсив, размер 40	изменение типа и размера шрифта
BackColor	желтый цвет	изменение фона метки
ForeColor	красный цвет	изменение цвета названия метки
TextAlign	TopCenter	выравнивание текста по центру

3. Работа с графикой. Выполнить двойной щелчок ЛКМ на пиктограмме Image панели элементов управления и охватить созданным объектом-прямоугольником букву А.
4. В окне свойств для объекта Image свойству Stretch присвоить значение True (размер вставленного рисунка, см. следующий пункт, становится равным размеру Image).
5. Для свойства Picture объекта Image выбрать в диалоговом окне каталог MsVB6.3/Common/Grafics/Icons/Elements файл Sun.ico.
6. Вставить в буквы р рисунок Face05.ico, для этого на форме создать еще один элемент управления Image с рисунком Face05.ico.
8. Маркировать второй объект Image: использовать меню Правка пункт Копировать или контекстное меню, затем меню Правка пункт Вставить. На вопрос: "Хотите, чтобы объект стал элементом массива?", ответить: "Нет", копия объекта появится в верхнем левом углу формы, перетащить рисунки на буквы р и увеличить объекты так, чтобы заполнить букву.

9. Кнопка. Выполнить щелчок ЛКМ на изображении кнопки Button панели элементов управления.

10. Расположить указатель мыши на созданной форме, вид которого при этом изменится со стрелки на крест, и переместить его в то место формы, где будет находиться один из углов создаваемого элемента интерфейса (обычно выбирается левый верхний угол).

11. Нажать левую кнопку мыши и перемещать указатель до тех пор, пока изображение элемента не станет требуемых размеров, после чего следует отпустить кнопку и объект будет создан.

12. Присвоить следующим свойствам появившейся метки предлагаемые значения (табл. 4):

Таблица 4

Свойства	Значение	Описание
Name	cmdExit	имя кнопки (к имени кнопки добавляют префикс cmd)
Text	Выход	текст, который располагается на метке
FlatStyle	Standart	вид кнопки - объемный
BackColor	Белый	изменение фона кнопки
Font	шрифт Arial, жирный, размер 20	изменение типа и размера шрифта
Height	1000	высота кнопки

13. Стартовать проект, остановить выполнение и удалить форму.

Упражнение 3. Разработка первого приложения - Приложение Hello World Порядок работы:

1. Создать новую форму.
2. Измените размеры формы, перетащив ее края. В окончательном варианте она должна иметь размеры около 7 см в ширину и 5 см в высоту.

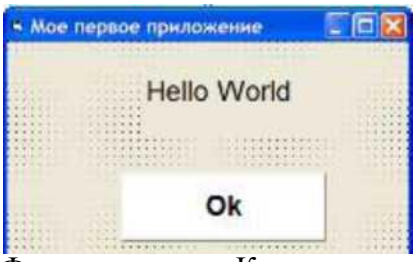


Рис. 1. Форма в режиме Конструктора форм

3. Создать в центре формы кнопку стандартного размера. Оттащить кнопку в нижнюю часть формы.

4. Создать на форме надпись (метка). Перетащите надпись так, чтобы она располагалась над кнопкой.

5. Выделить форму, щелкнув на ней мышью. О том, что выделена именно форма, а не один из размещенных на ней элементов, можно судить по содержимому окна свойств.

6. Задать значения двух свойств формы: Имя - frmHelloWorld и Text - Мое первое приложение.

7. Щелкните на элементе-надписи и задайте следующие свойства: Имя, Название, параметры шрифта, цвет шрифта, выравнивание.

8. Щелкните на элементе-кнопке и задайте следующие свойства: Имя - cmdOK, Название - Ок, стиль, параметры шрифта, фон, размеры. Форма должна выглядеть так, как показано на рисунке 1.

9. Дважды щелкните на кнопке cmdOK. Двойной щелчок на элементе (или форме) открывает окно программы со стандартным событием. Для кнопки стандартным является событие Click. На экране должен появиться шаблон процедуры (или заготовка) cmdOK_Click:

Private Sub cmdOk_Click(By Val sender As System.Object, ByVal e As System.EventArgs) Handles cmdOk.Click

End Sub

10. Не обращайте внимания на префикс Private Sub; сейчас важно лишь имя процедуры cmdOK_Click - оно означает, что после того, как пользователь нажмет кнопку cmdOK, будет выполнен код, находящийся в этой процедуре.

11. Введите между строками **Private Sub** и **End_Sub** запись: **End**

Когда пользователь нажимает кнопку cmdOK, происходит событие cmdOK_Click. В данном случае оно сообщает форме о том, что она должна выгрузить себя. Поскольку в нашем приложении нет других форм, выгрузка формы приводит к завершению приложения.

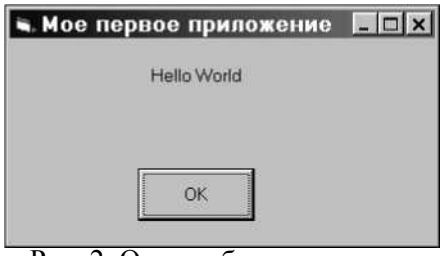


Рис. 2. Окно работающего приложения

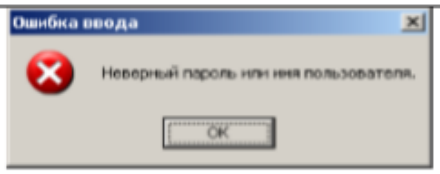
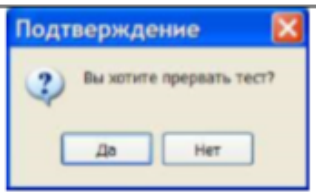
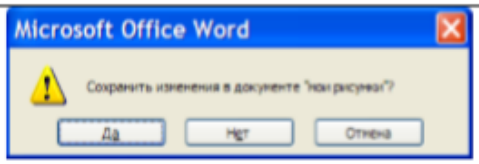
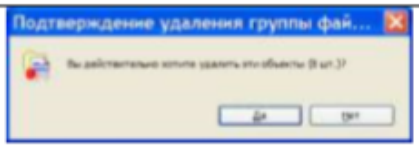
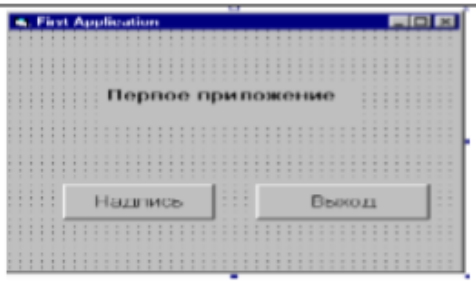
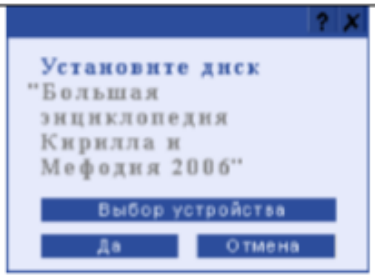
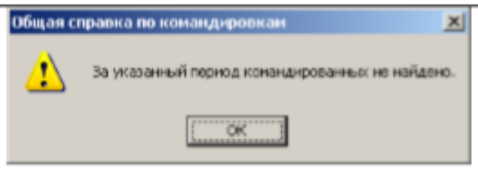
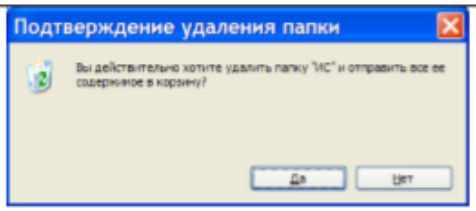
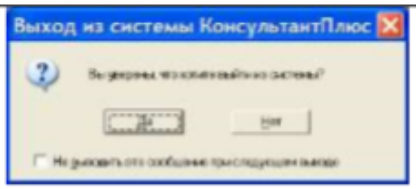
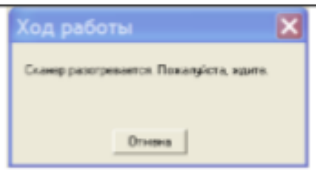
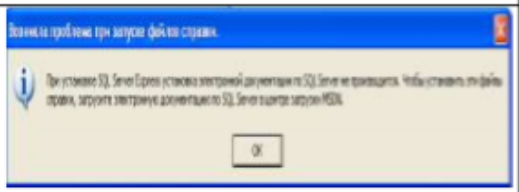
12. Перед сохранением проекта сохраняются все входящие в него файлы. В нашем приложении имеется всего одна форма, которой соответствует один файл. В сущности, файл проекта представляет собой список файлов компонентов. Выполните команду Файл - Сохранить Form1.vb как... и задать имя формы frmWorld.vb. Сохранить весь проект командой Файл - Сохранить все, задав имя проекта HelloWorld.

ЗАМЕЧАНИЕ. Обратите внимание - наша форма описывается тремя атрибутами: свойством Name (frmHelloWorld), свойством Text (Мое первое приложение) и именем файла (frmWorld.vb). Следует четко понимать, чем отличаются эти атрибуты: свойство Text выводится в заголовке формы, свойство Name служит для работы с формой в программе, а имя файла используется файлом проекта и операционной системой.

13. Выполните команду Начать отладку. Если все было сделано правильно, на экране появляется форма с сообщением Hello World (рис.2).

14. Завершите работу приложения кнопкой OK. Если у вас что-то не получилось, повторите описанные выше действия и найдите ошибку

Индивидуальные задания: создать форму согласно своему варианту

Вариант	Вид формы	Вариант	Вид формы
1		8	
2		9	
3		10	
4		11	
5		12	
6		13	
7		14	

Лабораторная работа №3
Интерактивные приложения в Visual Basic.

Цель работы: Создание Windows-приложений в среде Visual Basic 2010. Знакомство со средой Visual Basic 2010. Приобретение навыков проектирования графического интерфейса.

Задание

1. Создать новый проект в среде Visual Basic 2010.
2. Составить эскиз интерактивной формы (Рис. 1).
3. Составить программу вычисления заданного выражения (Табл. 4), предусмотрев ввод исходных данных через текстовые поля интерактивной формы и отображение результирующего значения посредством поля надписи той же интерактивной формы.
4. Выполнить сборку и компиляцию программы.
5. Запустить программу на выполнение, ввести исходные данные и получить результат вычисления заданного выражения. Записать полученные результаты и оформить отчет о проделанной работе.

Рисунок 1 Эскиз интерактивной формы Пример

Требуется создать Windows-приложение, позволяющее посредством спроектированного интерфейса в виде формы обеспечить ввод исходных данных и отображение результата вычисления следующего выражения:

$$Z = \frac{\sqrt{\frac{a}{b^2} + 5.68}}{(\sin x + b\cos y)^2}$$

где a = 114.6; b = 53.47 10^3; x = 36°; y = 0.87.

Порядок выполнения работы 1.

1. Создание нового проекта.
1.1. Запустить среду разработки MS Visual Studio 2010.
1.2. Создать новый проект, используя следующие меню: File - New Project.
1.3. Указать тип нового проекта: Visual Basic (Рис. 2.).
1.4. Указать шаблон, на основе которого создаётся проект: Windows Application.
1.5. Для подтверждения создания проекта нажать кнопку ОК.

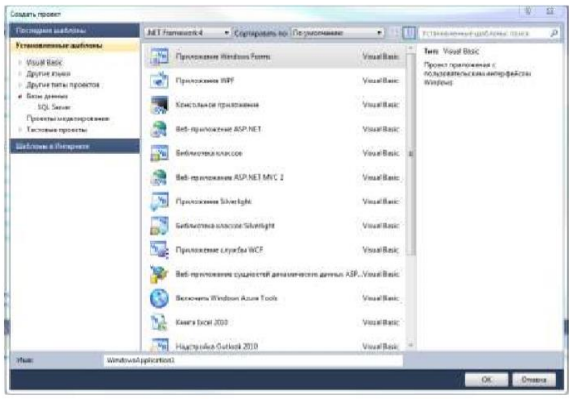


Рисунок 2 Создание нового проекта

1.6. В результате выполнения вышеописанных действий будет создана пустая форма (Рис. 3).

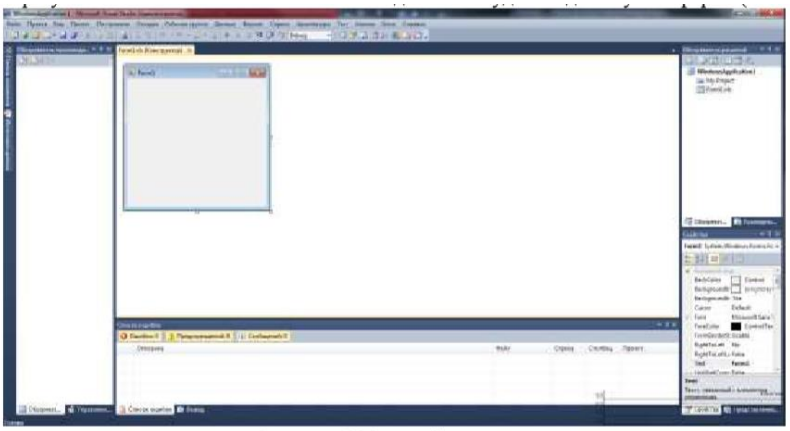


Рисунок 3 Пустая форма

2. Создание эскиза интерактивной формы.

2.1. Для ввода исходных данных: a, b, x, y и вывода результата вычисления выражения

$$Z = \frac{\sqrt{\frac{a}{b^2}+5.68}}{(\sin x + b\cos y)^2}$$

Необходимо разместить на форме элементы управления, как показано на Рис. 4. Заметим, что если в Вашем варианте используется менее четырѐх переменных, то и на проектируемой форме их должно быть меньше.

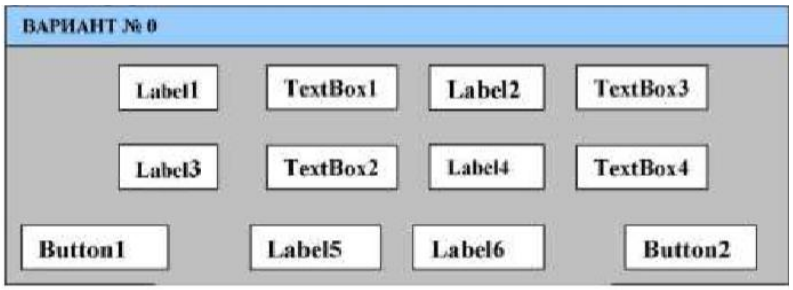


Рисунок 4 Размещение элементов управления

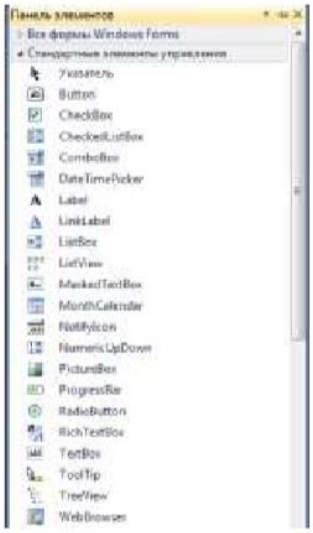


Рисунок 5 Панель Toolbox

Элементы управления Label (Label1 - Label6) представляют собой надписи и служат для отображения

текстовой информации на форме и вывода результата вычисления (a=, b=, x=, y=, z=, z).

Элементы управления TextBox (TextBox1 - TextBox4) представляют собой текстовые поля и служат для ввода информации на форме (a, b, x, y).

Элементы управления Button (Button1 - Button2) представляют собой кнопки и служат для выполнения расчѐта (Button1) и выхода из программы (Button2).

Для размещения элементов управления необходимо воспользоваться панелью Toolbox (Рис. 5) из меню View - Toolbox, перенося мышкой необходимые элементы с панели Toolbox на форму.

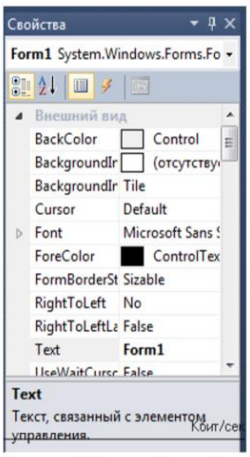


Рисунок 6 Панель Properties

После размещения всех необходимых элементов управления на форме необходимо задать их свойства через панель Properties (Рис. 6), которая появляется после одинарного щелчка мышью на нужном элементе управления, расположенном на форме. Каждый элемент управления имеет свой набор свойств. Свойства можно назначать не только элементам управления, но и форме.

2.2. Установите значения свойств BackColor и Text объекта Form1, как показано на Рис. 7

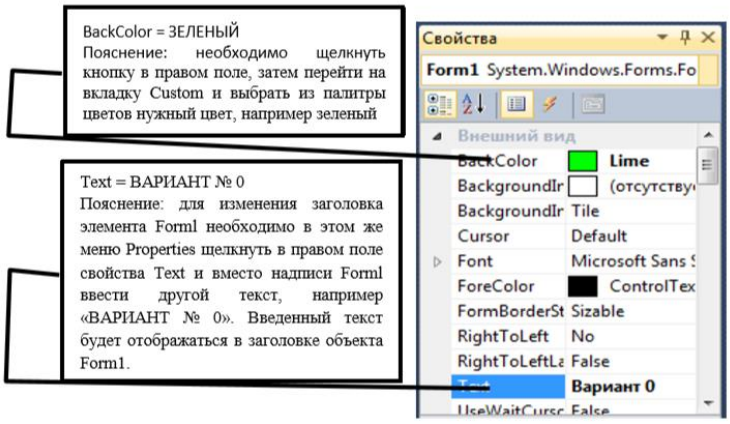


Рисунок 7 Свойства Form1

2.3. Установите значения свойств BackColor, Font и Text элемента - надписи Label1, размещенного на форме (Рис. 8).

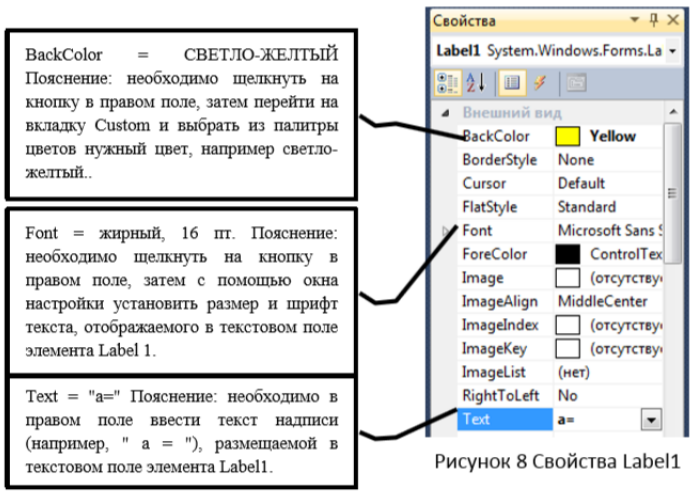


Рисунок 8 Свойства Label1

2.4. Аналогично установите для элементов Label2, Label3, Label4, Label5 и Label6 значения свойств, приведѐнные в Табл. 1.

Свойство	Значение
Label2.Text	b=
Label2.Font	жирный, 16 пт
Label2.BackColor	СВЕТЛО-ЖЕЛТЫЙ
Label3.Text	x=
Label3.Font	жирный, 16 пт.
Label3.BackColor	СВЕТЛО-ЖЕЛТЫЙ
Label4.Text	y=
Label4.Font	жирный, 16 пт.
Label4.BackColor	СВЕТЛО-ЖЕЛТЫЙ
Label5.Text	Z=
Label5.Font	жирный, 16 пт.
Label5.BackColor	СВЕТЛО-ЖЕЛТЫЙ
Label6.Font	жирный, 16 пт.
Label6.BackColor	СВЕТЛО-ЖЕЛТЫЙ

2.5. Установите значения свойства Font для элемента TextBox1, как показано на Рис. 9.

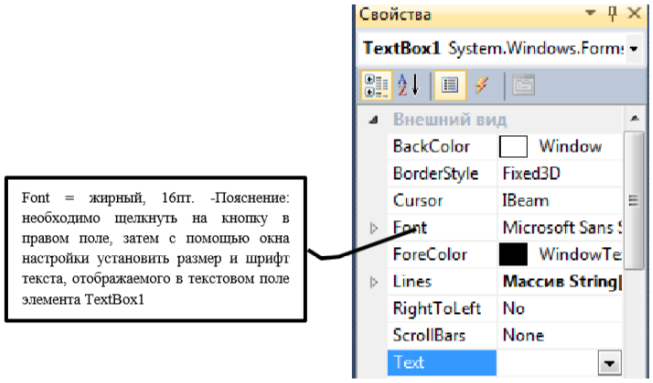


Рисунок 9 Свойства TextBox1

Аналогично установите для элементов TextBox2, TextBox3, TextBox4 значения свойств, приведѐнные в Табл. 2.

Свойство	Значение
TextBox2.Font	жирный, 16 пт.
TextBox3.Font	жирный, 16 пт.
TextBox4.Font	жирный, 16 пт.

2.6. Установите значения свойств BackColor, Font и Text для элемента Buttonl, как показано на Рис. 10.

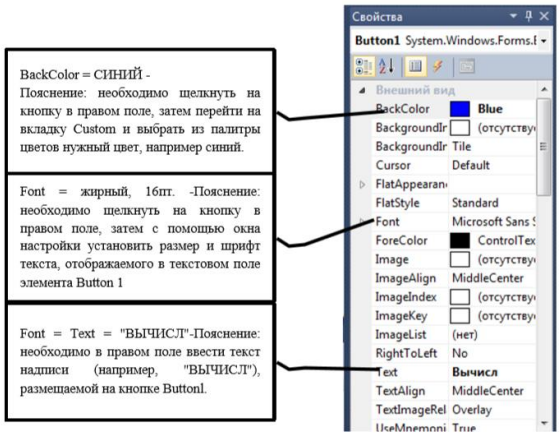


Рисунок 10 Свойства Buttonl

2.7. Аналогично установите для элемента Button2 значения свойств, приведѐнные в Табл. 3.

Свойство	Значение
Button2.Text	"ВЫХОД"
Button2.Font	жирный, 16 пт.
Button2.BackColor	КРАСНЫЙ

4. Написание программы (кода) включает в себя разработку кода для обработки события загрузки формы и нажатия кнопок «ВЫЧИСЛ» и «ВЫХОД» 3.1. Выполните двойной щелчок левой кнопкой мыши по пустому месту формы. В появившемся окне головного модуля Form1.vb будут присутствовать заголовок и концевик программы, обрабатывающей событие «ЗАГРУЗКА ФОРМЫ»

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

End Sub

Введите между этими строками код программы для обработки события «ЗАГРУЗКА ФОРМЫ»:

TextBoxl.Text = ""

TextBox2.Text = ""

TextBox3.Text = ""

TextBox4.Text = ""

Label6.Text = "

Пояснение: с помощью введенных операторов осуществляется чистка текстовых полей в элементах TextBox и Label6.

3.2. Выполните двойной щелчок левой кнопкой мыши по кнопке «ВЫЧИСЛ». В появившемся окне головного модуля Forml.vb будут присутствовать заголовок и концевик программы, обрабатывающей событие «НАЖАТИЕ КНОПКИ «ВЫЧИСЛ». Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click End Sub

Введите между этими строками код программы для обработки события:

```
Dim A, B, X, Y, X1, Z2, Z1, Z As Double
        A = Val(TextBox1.Text)
        B = Val(TextBox2.Text)
        X = Val(TextBox3.Text)
        Y = Val(TextBox4.Text)
        X1 = X * 3.14159 / 180          'преобразует значения из радианов в градусы
        Z2 = (Math.Sin(X1) + B * Math.Cos(Y)) ^ 2
        Z1 = Math.Sqrt(A / B) + 5.68
        Z = (Z1 / Z2)
        Label6.Text = CStr(Z)
```

Пояснение: Оператор Dim объявляет переменные A, B, X, Y, X1, Z2, Z1, Z как числовые переменные, имеющие тип числа с плавающей запятой двойной точности. Функция Val() осуществляет преобразование аргумента строкового типа в числовое значение. Функция CStr() осуществляет преобразование аргумента числового типа в строковое значение. Использование в вычислениях стандартных математических функций реализуется посредством выражений, в которых указано имя класса Math и следующее за ним через точку имя стандартной функции (например, Sin).

3.3. Выполните двойной щелчок левой кнопкой мыши на кнопку «ВЫХОД». В появившемся окне головного модуля Forml.vb будут присутствовать заголовок и концевик программы, обрабатывающей событие «НАЖАТИЕ КНОПКИ «ВЫХОД». Введите между этими строками код программы для обработки события.

```
Application.Exit()
```

Пояснение: для класса Application осуществляется вызов встроенного метода Exit().

4. Запустить приложение на выполнение можно командой Начать отладку из меню отладка.

4.1. После запуска приложения на выполнение автоматически появляется интерактивная форма с размещенными на ней элементами управления (Рис. 1.). Введите с клавиатуры в соответствующие поля на форме значения исходных данных, указанных в Вашем варианте (a = 114.6; b = 53 470; x = 36; y = 0.87).

4.2. Для выполнения вычислений нажмите кнопку «ВЫЧИСЛ». Результат расчета появится на форме в поле надписи Label6. Сеанс вычислений можно повторять многократно для различных значений исходных данных. Для завершения работы приложения необходимо нажать кнопку «ВЫХОД». Для выхода из программного комплекса Microsoft Visual Studio 2010 необходимо использовать команду Выход из меню Файл.

Табл. 4. Варианты к лабораторной работе №1.

№ задания	Выражение	Значения переменных
1	$Y = \sqrt[4]{a^3} + \frac{3\cos 35^\circ + \operatorname{arctg} 3^\circ}{\ln(a+10) - (b-x)^3} + \sqrt{b-x} - 7.128$	$a=3.17$ $b=5.34$ $x=4.1$
2	$Y = \frac{4\sin 35^\circ + \arccos 0.2}{(a-b)^2 + \ln(x+9)} + \sqrt{x} - \sqrt{a-b} + 0.29$	$a=8.31$ $b=4.50$ $x=2.71$
3	$Y = \sqrt[3]{x^3} + \frac{5\sin 34^\circ + \operatorname{tg} 34^\circ}{\ln(x+10) - (a-b)^3} + \sqrt{a-b} - 4.023$	$a=8.31$ $b=4.50$ $x=4.412$
4	$Y = \sqrt[4]{(a-b)^3} + \frac{2\arcsin 0.9 + 5\cos 28^\circ}{\ln(a-b) + \sqrt{0.9}} - 6.493$	$a=5.34$ $b=4.10$
5	$Y = \frac{2\arccos 0.1 - \sin 33^\circ}{\sqrt{a} + \ln a} \sqrt[5]{a^5} + tg33^\circ - 2.201$	$a=2.412$
6	$Y = \sqrt[3]{a^2} + \frac{2\cos x + \operatorname{arctg} 10^\circ}{\ln(a^4+10) + \sqrt{a}} - 1.671$	$a=2.412$ $x=29^\circ$
7	$Y = \sqrt[4]{a^3} + \frac{4\cos x + \operatorname{arctg} 3}{\ln(a+10) - (b-4.1)^3} + \sqrt{b-4.1} - 9.767$	$a=2.71$ $b=5.34$ $x=33^\circ$
8	$Y = \frac{2\sin x + \arccos 0.2}{(a-4.5)^2 + \ln(b+9)} \sqrt[3]{b} - \sqrt{a-4.5} + 2.214$	$a=7.31$ $b=2.17$ $x=21^\circ$
9	$Y = \sqrt[3]{a^2} + \frac{2\arccos 0.8 + \sin x}{\ln(a^2+10) - \sqrt{0.8}} - 1.665$	$a=3.115$ $x=51^\circ$
10	$Y = \frac{2\arcsin 0.8 - \cos x}{\sqrt{a} + \ln(a^*+1)} - \sqrt[3]{a^3} + \operatorname{tg} 44^\circ + 1.83$	$a=4.115$ $x=44^\circ$
11	$Y = \sqrt[3]{a^2} + \frac{2\sin 73^\circ + \operatorname{arctg} 9}{\ln(a^3+5) + \sqrt{a}} - 3.313$	$a=8.133$
12	$Y = \sqrt[3]{a^2} + \frac{4\sin x + \operatorname{arctg} 0.3}{\ln(a^2+10) - \sqrt{a-4}} - 4.875$	$a=8.71$ $x=78^\circ$
13	$Y = \frac{2\cos 37^\circ + \arcsin 0.8}{(a-2.7)^2 - \ln(b+10)} \sqrt[3]{b} - \sqrt{a-2.7} + 3.67$	$a=8.735$ $b=2.41$
14	$Y = \sqrt[5]{a^3} + \frac{2\arcsin 0.2 - 5\cos x}{\ln(a^2+6) - \sqrt{0.2}} - 1.015$	$a=3.891$ $x=64^\circ$
15	$Y = \frac{4\operatorname{arctg} 8 + 5\sin x}{\ln(a^2+1) - \sqrt{8}} + \sqrt[3]{a} - 5.601$	$a=8.735$ $x=22^\circ$
16	$Y = \sqrt[4]{a^3} - \frac{2\arcsin 0.13 + \sqrt{a}}{5\operatorname{tg} x - \ln(a^2+10)} - 5.882$	$a=12.13$ $x=14^\circ$

17	$V = \frac{U\sqrt[4]{\lvert 3x - y\rvert} + 2.8\,\mathrm{arctg}\, y^{\frac{3}{2}} - \cos 3x}{(3x - y^3)^4 - U \cdot \sin 51.3^\circ} + 1.7$	$u = 2.95 \cdot 10^4$ $y = 0.88$ $x = 1.22 \cdot 10^3$
18	$Y = \frac{a \ln 3x^{\frac{2}{3}} + b^3 \sin 2x - \sqrt[6]{3x^{\frac{2}{3}}}}{a\,\mathrm{tg}\, 2x + \lvert 3x^{\frac{2}{3}} - b^4 \rvert} + 1.794$	$a = 6.35$ $b = 1.5 \cdot 10^2$ $x = 0.56$
19	$Z = \frac{l\,\mathrm{tg}(2x^2 - 0.25) + m\sin^2 52.1^\circ}{(2x^2 - 0.25)^{\frac{4}{3}} + lmx} + 1.538$	$l = 2.7 \cdot 10^3$ $m = 6.54$ $x = 0.82$
20	$R = \frac{a\sqrt{b^2 - 0.8\sin 3.2x} + tg(0.8\sin 3.2x)}{\ln(b^2 - 0.8\sin 3.2x) + e^a} - 1041.29$	$a = 3.42$ $b = 152 \cdot 10^2$ $x = 14.5^\circ$
21	$P = \frac{a\sin^2(2x) + \sqrt{a^2 2b}}{\ln(a^2 + 2b) + a^{-2b}} + a^2 - 39.248$	$a = 4.78$ $b = 2.83 \cdot 10^2$ $x = 35.5^\circ$
22	$M = \frac{le^{-0.45x} + \sqrt{0.45x} + n\cos(3y + 15^\circ)}{e\,\mathrm{tg}(3y + 15^\circ) - n\ln 0.45x} - 4595.752$	$l = 2.8 \cdot 10^4$ $n = 1.45$ $x = 2.2$ $y = 8.2^\circ$
23	$T = \frac{2p\sin^3(x/4) - \sqrt{4p^2 + q} \cdot \mathrm{tg}(x/4)}{\ln(4p^2 + q) + q\cos x - 2.78} + 125.833$	$p = 3.7 \cdot 10^2$ $q = -21.2$ $b = 2.17$ $x = 132^\circ$
24	$Z = \frac{a^2\sqrt{b + 0.45\cos 2.7x} + \mathrm{tg}(0.45\cos 2.7x)}{(b + 0.45\cos 2.7x)^{4/3} + e^{-b}} - 14179.162$	$a = -1.5 \cdot 10^2$ $b = 0.98$ $x = 15.2^\circ$
25	$W = \frac{u^3 e^{2x-1} - v\,\mathrm{tg}(2x - 1)}{u^3 \ln 2x + v\cos 41.2^\circ} + 1.709$	$u = 2.72$ $v = 1.045 \cdot 10^3$ $x = 0.24$
26	$P = \frac{\sqrt{m^2 - 2.83 * 10^{-2} * n^{3/2} + 0.48m^2}}{\sin(2.83 * 10^{-2} * n^{3/2}) + m^2 * \cos 28.3^\circ} + 2.404$	$m = 5.48$ $n = 0.75 \cdot 10^2$ $q = 2.8$
27	$S = \frac{0.52p^2 + 4.2 * 10^{-3} e^{2q} - (\cos 0.52m)^2}{\sqrt{0.52p^2 + 4.2 * 10^{-3} e^{2q} + 753 * e^{2q}}} - 195.553$	$p = 2.8 \cdot 10^2$ $q = 0.35$ $m = 37.2^\circ$
28	$W = \frac{473.2U^2 - 0.82V^{-0.54} + \cos U^3}{\sqrt{473.2U^2 - 0.82V^{-0.54} + \sin U^3}} - 5979.319$	$U = 2.75 \cdot 10^2$ $V = 1.54$
29	$r = \frac{\sqrt{p^{0.32} * tg^2 0.75 * q + e^{0.75q}}}{622.7 * 10^2 * p^{0.32} + 0.75 * q} + 2.9$	$p = 5.75$ $q = 1.3$
30	$q = \frac{pe^{0.25x} + \sqrt{0.25x} - tg^2 2y}{p^{3/4} \ln(0.25x + 7.3) + 2.2p^{3/4}} - 1.538$	$p = 1.3 \cdot 10^3$ $x = 4.75$ $y = 23.2^\circ$

Лабораторная работа № 4

Тема: Windows приложение в Visual Basic: опрос-шутка.

Цель работы: Познакомиться с некоторыми возможностями работы с мышью и диалоговыми окнами.

Задание 1-го уровня

1. Создать новый проект.

2. Составить эскиз интерактивной формы - опроса (Рис. 5.1).

3. Задать значения свойств элементов управления, размещенных на интерактивной форме.

4. Для каждого элемента управления написать программный код, соответствующий событию активизации (нажатия) элемента управления, так, чтобы при наведении курсора мыши на кнопку «ДА» кнопка «убегала» (меняла своѐ местоположение на форме), а при выборе кнопки «НЕТ» появлялось сообщение: «Мы рады, что Вы считаете, что в повышении размера стипендии необходимости нет!»

5. Осуществить сборку и компиляцию модулей проекта. 6. Проверить работоспособность приложения.

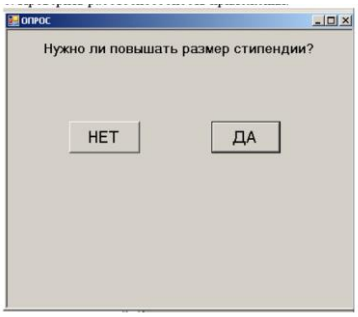


Рис. 1. Эскиз интерактивной формы

Задание 2-го уровня

Доработайте приложение из первого задания так, чтобы при наведении курсора мыши на кнопку «НЕТ» эта кнопка становилась в 2 раза больше, а при «снятии» курсора мыши с кнопки возвращалась к прежним размерам, а также чтобы при «снятии» курсора мыши с кнопки «ДА» эта кнопка возвращалась на прежнее место на форме.

Задание 3-го уровня

Доработайте приложение из первого и второго заданий таким образом, чтобы при каждом десятом наведении курсора мыши на кнопку «ДА» появлялось сообщение: «Поздравляем, Вы достойны повышения стипендии!».

Порядок выполнения работы (1-й уровень)

1. Создать новый проект командой New Project из меню File (порядок создания нового проекта подробно описан в лабораторной работе № 3).

2. Создать эскиз интерактивной формы.

2.1. Используя панель инструментов ToolBox, разместить на форме элементы управления (кнопки - Button1 ÷ Button3 и надпись – Label1), как показано на Рис. 5.2.

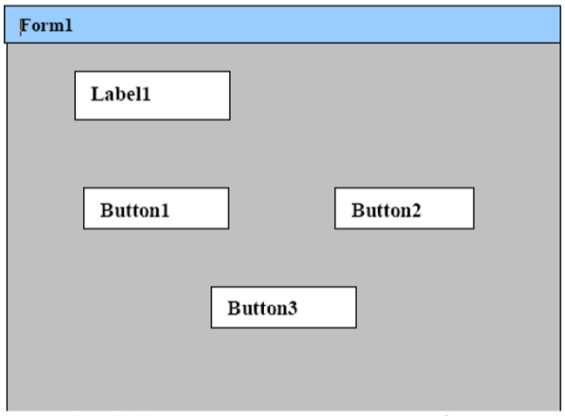


Рис. 5.2. Размещение элементов управления на форме

3. После размещения всех необходимых элементов управления на форме необходимо задать их свойства через панель Properties, которая появляется после одинарного щелчка мышью по нужному элементу управления, расположенному на форме. Каждый элемент управления имеет свой набор свойств. Свойства можно назначать не только элементам управления, но и форме.

3.1. Установите значения свойств Cursor, MaximizeBox, Size и Text объекта Form1, как показано на Рис. 5.3.

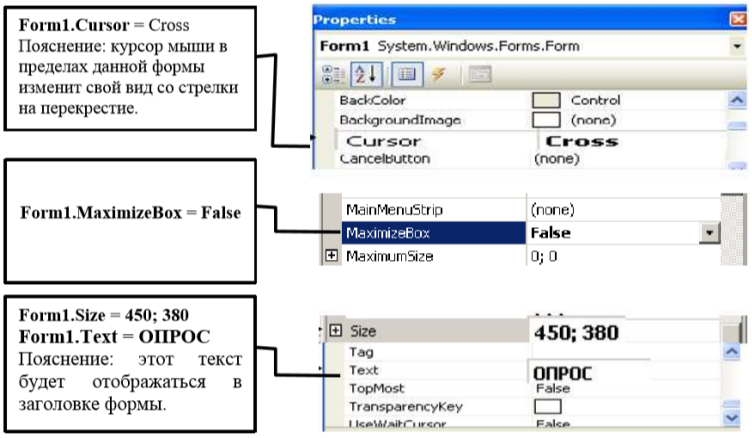


Рис. 5.3. Свойства Form1

3.2. Установите значения свойств элемента – надпись (Label), как указано в Табл. 5.1.

Табл. 5.1

Свойство	**Значение**
Label1.Text	Нужно ли повышать размер стипендии?
Label1.Font.Size	12
Label1.Font.Bold	TRUE
Label1.TextAlign	MiddleCenter

3.3. Установите значения свойств элементов – кнопок (Button), как указано в Табл. 5.2.

Свойство	**Значение**
Button1.Text	НЕТ
Button1.Font.Size	15
Button2.Text	ДА
Button2.Font.Size	15
Button3.Text	Закрыть

В результате изменения свойств вышеперечисленных объектов форма Form1 примет вид, указанный на Рис. 5.4.

4. Написание программы (кода) включает в себя разработку кода обработки событий для кнопок «НЕТ», «ДА» и «Закрыть».

4.1. Обработка события - нажатие кнопки «НЕТ». Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке Button1 и ввести код:

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click MessageBox.Show("Мы рады, что Вы считаете, что в повышении размера стипендии необходимости нет!", "Благодарим Вас за участие в опросе") End Sub

Пояснение: MessageBox служит для вывода сообщений. В скобках указываются текст сообщения и заголовок сообщения. Также можно указать тип сообщения, иконку сообщения и ряд других параметров.

4.2. Обработка события – наведение на кнопку «ДА». Для этого необходимо выполнить двойной щелчок левой кнопкой мыши по кнопке Button2 и в редакторе кода выбрать метод MouseMove (Рис. 5.4). Этот метод в отличие от метода Click срабатывает не на нажатие, а на наведение курсора мыши. Введите код, описывающий реакцию приложения на наведение курсора мыши на кнопку «ДА»:

Private Sub Button2_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles Button1.MouseMove Dim x, y As Single x = Rnd() y = Rnd() Button1.Location = New Point(x * (Me.Width * 0.9 - Button1.Width), y * (Me.Height * 0.9 - Button1.Height)) End Sub

Пояснение: свойство Location задаèт координаты положения кнопки Button1 относительно верхнего левого угла формы. Чтобы кнопка не оказалась за пределами формы, нужно контролировать размеры формы по ширине (Me.Width) и по высоте (Me.Height). Из показателей размеров формы, умноженных на понижающий коэффициент 0.9, вычитаем соответствующие показатели размеров кнопки (Button1.Width и Button1.Height). Для того, чтобы кнопка перемещалась на случайную позицию, генерируем две псевдослучайных величины (X, Y), изменяющиеся в пределах от 0 до 1, и домножаем на них координаты кнопки Button1.

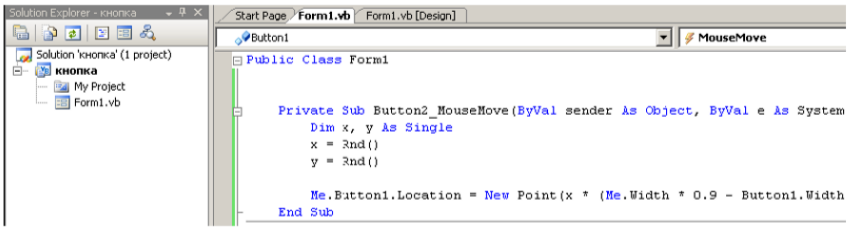


Рис. 5.4. Выбор метода MouseMove для Button1

4.3. Кнопка «Закрыть» должна закрывать приложение. Код для обработки события – нажатия кнопки «Закрыть» напишите самостоятельно.

5. Сборка и компиляция модулей проекта выполняется командой Build WindowsApplication из меню Build. Убедитесь, что приложение откомпилировалось без ошибок, в противном случае проверьте правильность написания кода. Сохраните Ваш проект командой Save All из меню File. 6. Запустить приложение на выполнение можно командой Start Debugging из меню Debug. Проверьте приложение на работоспособность. Покажите преподавателю результаты Вашей работы. Для выхода из программного комплекса Microsoft Visual Studio 2010 необходимо использовать маршрут главного меню: File → Exit.

Пояснения для выполнения задания 2-го уровня

1. Для обработки событий: для наведения и снятия курсора используйте методы MouseHover и MouseLeave. 2. Размеры кнопки можно определить через свойства .Size.Width и .Size.Height. Задать новые размеры кнопки можно используя конструкцию: Button.Size = New Size(a, b), где a и b новые размеры кнопки (ширина и высота).

Лабораторная работа № 5

Тема: Просмотр каталогов и файлов

Цель работы: Познакомиться с некоторыми возможностями работы с текстовыми и графическими файлами.

Для того чтобы иметь представление о системе, необходим список логических дисков, который можно получить с помощью метода GetLogicalDrives класса Environment. Например, следующая строка программы размещает имена дисков в массиве строк L_D. Количество дисков определяется с помощью функции Length. Dim L_D() As String = Environment.GetLogicalDrives() Dim Nd As Integer = L_D. Length

Какие папки размещены в данном каталоге, можно узнать с помощью метода GetDirectories класса SystemIO.Directory. Например, строка ниже помещает список всех подкаталогов корневого каталога в массив строк S_F. Dim S_F() As String = IO.Directory.GetDirectoriesC'cA") Dim Nf As Integer = S_F. Length Указанный метод дает возможность отфильтровать папки по маске, например, выбрать те, которые начинаются с буквы W: Dim W() As String = IO.Directory.GetDirectories("c:\,"W*")

Просмотр и выбор в дереве каталогов осуществляется с помощью класса FolderBrowserDialog, который предоставляет способ выдачи приглашения пользователя для просмотра, выбора, а также создания папки. Этот класс используется, когда необходимо разрешить пользователю выбирать только папки, но не файлы. Обзор папок осуществляется с помощью дерева. Могут выбираться только папки из файловой системы; выбор виртуальных папок невозможен. В примере кода ниже создается новый объект класса FolderBrowserDialog и устанавливается имя корневой папки RootFolder, с которой начинается просмотр, например, Мой компьютер: Dim FolderBrowserDialog1 As New FolderBrowserDialog() FolderBrowserDialog1 .RootFolder = _ Environment.SpecialFolder.MyComputer Можно использовать свойство ShowNewFolderButton, чтобы указать, имеется ли у пользователя возможность создания новых папок с помощью кнопки Создать папку, например, следующая строка запретит отображение этой кнопки: FolderBrowserDialogl.ShowNewFolderButton = False Свойство Description предоставляет пользователю текстовую подсказку. FolderBrowserDialog1.Description = "выберите папку" Метод ShowDialog служит для отображения диалогового окна с деревом каталогов (рис. 10.1), в котором пользователь выбирает папку. Если пользователь выбрал папку, то результатом диалога является DialogResultOK, а свойство SelectedPath будет содержать путь к выбранной папке. В строках кода ниже путь отображается в поле Label:

```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        If FolderBrowserDialog1.ShowDialog() = DialogResult.OK Then
            Label1.Text = FolderBrowserDialog1.SelectedPath
        End If
    End Sub
End Class
```

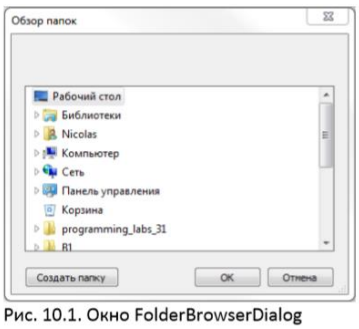


Рис. 10.1. Окно FolderBrowserDialog

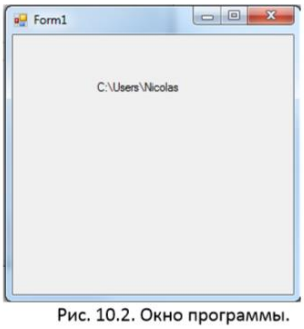


Рис. 10.2. Окно программы.

Объект FolderBrowserDialog является модальным диалоговым окном. Поэтому, когда это окно отображается, остальная часть приложения блокируется до тех пор, пока

пользователь не выберет папку. Программа должна скрыть или закрыть диалоговое окно перед тем, как разрешить ввод в вызывающую программу.

Пример программы Данный пример кода может быть использован, например, в качестве обработки события Button.Click. В нем с помощью метода ShowDialog класса FolderBrowserDialog осуществляется выбор папки, полное имя которой выводится в поле Label. Для доступа к указанному классу требуется строка Imports SystemIO. В качестве корневого каталога указан Мой компьютер.

Пример 10.1

```
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Dim FolderBrowserDialog1 As New FolderBrowserDialog()
        FolderBrowserDialog1.Description = "выберите папку"
        FolderBrowserDialog1.ShowNewFolderButton = False
        FolderBrowserDialog1.RootFolder = Environment.SpecialFolder.MyComputer
        If FolderBrowserDialog1.ShowDialog() = DialogResult.OK Then
            Label1.Text = FolderBrowserDialog1.SelectedPath
        Else
            Label1.Text = ""
        End If
    End Sub
End Class
```

Выбор файла в каталоге

Для того чтобы получить список файлов из произвольной папки, можно применить метод GetFiles класса System.IO.Directory. Например, ниже список файлов помещается в массив строк FL, а количество файлов - в переменную N. Dim FL() As String = IO.Directory.GetFiles("c:\") Dim N As Integer = FL. Length Метод GetFiles позволяет отфильтровать файлы по маске, например, выбрать те, которые начинаются с буквы S: Dim FL() As String = System.IO.Directory.GetFiles("c:\","S*") Dim N As Integer = FL.Length

Для выбора произвольного файла из папки служит класс OpenFileDialog. Этот класс позволяет проверить, существует ли файл, и открыть его. Свойство ShowReadOnly определяет, отображается ли в диалоговом окне флажок «Доступно только для чтения». Свойство ReadOnlyChecked показывает, установлен ли флажок «Доступно только для чтения». Значительная часть возможностей этого класса находится в классе FileDialog.

После создания нового объекта класса OpenFileDialog требуется указать папку, в которой пользователь начнет выбор файла.

Dim openFileDialogl As New OpenFileDialog() openFileDialogl.InitialDirectory = "C:\"

Сократить количество просматриваемых файлов можно с помощью маски, которая состоит из пар описание-маска, разделенных вертикальной чертой. Например, ниже задаются два варианта маски. openFileDialogl.Filter = "текстовые файлы .txt|*.txt|все файлы|*.*" openFileDialogl .FilterIndex = 2 Можно также изменить заголовок диалогового окна: openFileDialogl.Title = "Выбор текстового файла" Метод ShowDialog служит для отображения диалогового окна с деревом каталогов (рис. 10.3), в котором пользователь выбирает файл. Если пользователь выбрал файл, то результатом диалога является DialogResultOK, а свойство SafeFileName будет содержать имя выбранного файла. If openFileDialog1.ShowDialog() = DialogResult.OK Then Labell .Text = openFileDialogl.SafeFileName End If

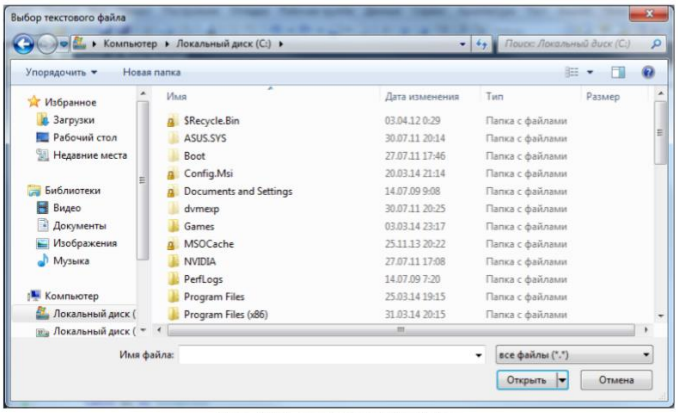


Рис. 10.3. Окно OpenFiledialog

Объект OpenFileDialog является модальным диалоговым окном. Поэтому, когда это окно отображается, остальная часть приложения блокируется до тех пор, пока пользователь не выберет папку. Программа должна скрыть или закрыть диалоговое окно перед тем, как разрешить ввод в вызывающую программу. Если следует предоставить пользователю возможность выбрать папку, а не файл, используйте объект FolderBrowserDialog.

Пример программы В следующем примере кода создается экземпляр класса OpenFileDialog. Чтобы использовать указанный класс, в программу обязательно должно быть добавлено пространство имен System.IO. С помощью метода ShowDialog отображается диалоговое окно, в котором осуществляется выбор текстового файла. Для окна указывается заголовок Title, папка InitialDirectory, с которой начинается выбор, а также строка фильтра Filter, состоящая из пар подсказка\маска. Имя файла отображается в компоненте Label. Содержимое текстового файла читается в поле TextBox. Для решения проблем с распознаванием русских букв указывается соответствующая кодовая таблица Text.Encoding.GetEncoding(1251).

Пример 10.2

```
Imports System.IO
Public Class Form1

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Dim fs As FileStream = Nothing
        Dim openFileDialogl As New OpenFileDialog()
        Dim INI_FLD As String = "C:\_Pro"
        openFileDialogl.Title = "Выбор текстового файла"
        openFileDialogl.InitialDirectory = INI_FLD
        openFileDialogl.Filter = "текстовые файлы .txt|*.txt|все файлы|*.*"
        openFileDialogl.FilterIndex = 2
        openFileDialogl.RestoreDirectory = True
        openFileDialogl.FileName = ""
        If openFileDialogl.ShowDialog() = DialogResult.OK Then
        Try
            fs = openFileDialogl.OpenFile()
            If (fs IsNot Nothing) Then
                Label1.Text = openFileDialogl.SafeFileName
                Dim fi As New StreamReader(fs, _
                System.Text.Encoding.GetEncoding(1251))
                TextBox1.Text = fi.ReadToEnd()
                fi.Close()
            End If
        Catch el As Exception
            Label1.Text = el.Message
        Finally
            fs.Close()
        End Try
        End If
    End Sub
End Class
```

Выбор файла в каталоге

Сохранение файла в папке Для сохранения файла служит класс SaveFileDialog, который позволяет выбрать папку для сохранения, открыть и перезаписать существующий файл или создать новый. Этот класс не может наследоваться. Значительная часть возможностей этого класса находится в классе FileDialog.

После создания нового объекта класса SaveFileDialog требуется указать папку, в которой пользователь начнет выбор файла. Dim SaveFileDialogl As New SaveFileDialog() SaveFileDialogl.InitialDirectory = "C:\Tmp\"

Сократить количество просматриваемых файлов можно с помощью маски, которая состоит из пар описание-маска, разделенных вертикальной чертой. Например, ниже задаются два варианта маски. SaveFileDialogl.Filter = "текстовые файлы .Ш|*.Ш|все файлы|*.*" SaveFileDialogl .Filterlndex = 2

Можно также изменить заголовок диалогового окна: SaveFileDialogl.Title = "Выбор текстового файла"

Метод ShowDialog служит для отображения диалогового окна (рис. 6.3), в котором пользователь выбирает папку. Если пользователь подтвердил сохранение файла, то результатом диалога является DialogResultOK. Функция OpenFile открывает или создает файл для записи нового содержимого. Свойство FileName содержит имя записанного файла.

If SaveFileDialogl.ShowDialog() = DialogResult.OK Then fs = SaveFileDialog1.OpenFile() Labell .Text = SaveFileDialogl .FileName Объект SaveFileDialog является модальным диалоговым окном. Поэтому, когда это окно отображается, остальная часть приложения блокируется до тех пор, пока пользователь не выберет папку. Программа должна скрыть или закрыть диалоговое окно перед тем, как разрешить ввод в вызывающую программу.

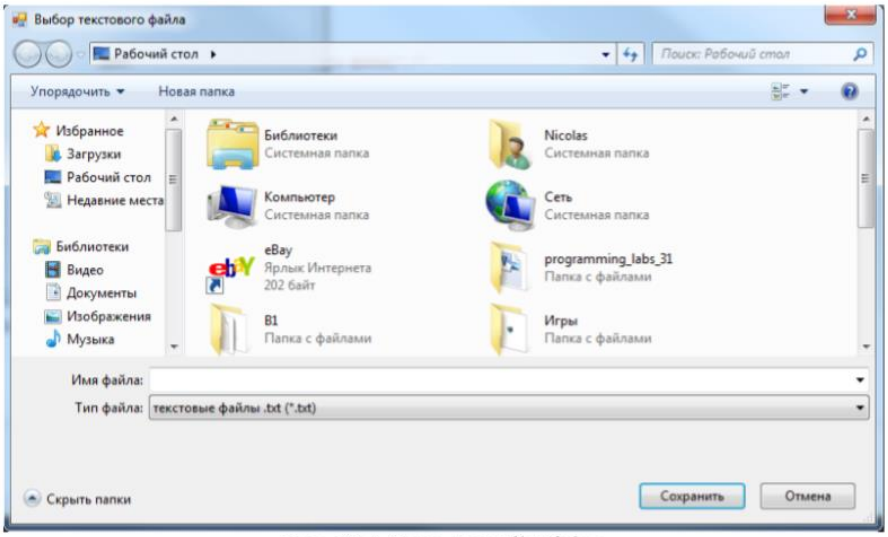
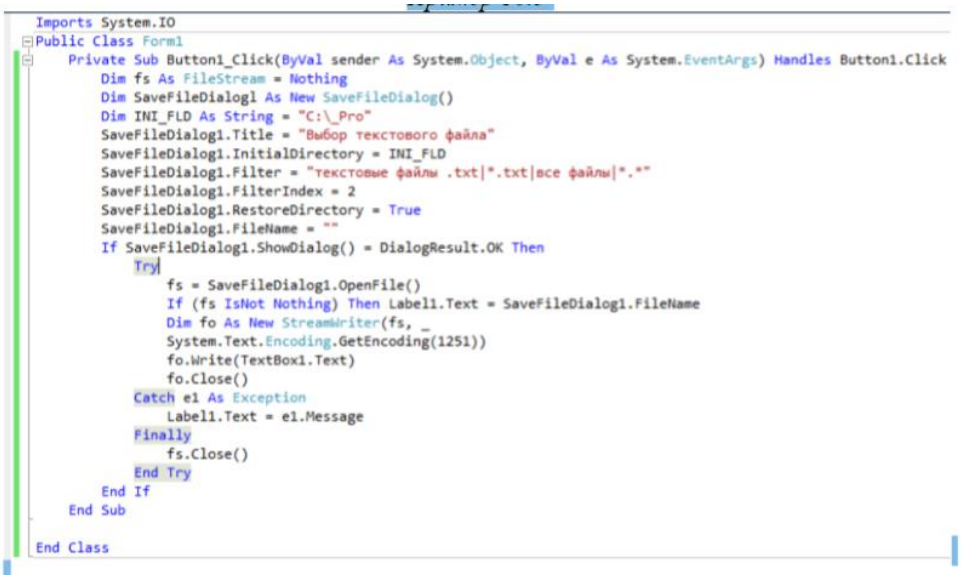


Рис. 10.4. Окно SaveFileDialog

Пример программы

В следующем примере кода показано создание SaveFileDialog, задание элементов, вызов диалогового окна с помощью метода ShowDialog и сохранение текущего файла. В созданный текстовый файл выводится содержимое компонента TextBox. Для решения проблем с распознаванием русских букв указывается соответствующая кодовая таблица Text.Encoding.GetEncoding(1251). В примере предполагается, что используется форма, на которой размещена кнопка.

Пример 10.3



```
Imports System.IO
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim fs As FileStream = Nothing
        Dim SaveFileDialog1 As New SaveFileDialog()
        Dim INI_FLD As String = "C:\_Pro"
        SaveFileDialog1.Title = "Выбор текстового файла"
        SaveFileDialog1.InitialDirectory = INI_FLD
        SaveFileDialog1.Filter = "текстовые файлы .txt|*.txt|все файлы|*.*"
        SaveFileDialog1.FilterIndex = 2
        SaveFileDialog1.RestoreDirectory = True
        SaveFileDialog1.FileName = ""
        If SaveFileDialog1.ShowDialog() = DialogResult.OK Then
            Try
                fs = SaveFileDialog1.OpenFile()
                If (fs IsNot Nothing) Then Label1.Text = SaveFileDialog1.FileName
                Dim fo As New StreamWriter(fs, _
                System.Text.Encoding.GetEncoding(1251))
                fo.Write(TextBox1.Text)
                fo.Close()
            Catch e1 As Exception
                Label1.Text = e1.Message
            Finally
                fs.Close()
            End Try
        End If
    End Sub

End Class
```

Просмотр графических файлов

Приводим фрагмент кода, предназначенный для просмотра в окне PictureBox графического файла с именем, полученным из диалога openFileDialog. Параметр PictureBoxSizeMode.Zoom означает, что рисунок будет пропорционально растянут в окне PictureBox.

PictureBoxl.Visible = False

PictureBoxl .Image = New Bitmap(openFileDialogl .FileName)

PictureBoxl.SizeMode = PictureBoxSizeMode.Zoom

PictureBoxl.Visible = True

Для удобства список файлов можно разместить в компоненте ListBox. Список файлов помещаем в массив FL с помощью метода GetFiles класса System.IO.Directory. Строки в ListBox очищаются при помощи функции Items.Clear, а содержимое в них добавляется функцией Items.Add.

Dim FL() As String = IO.Directory.GetFiles("c:\Foto","*.jpg")

Dim N As Integer = FL.Length

ListBoxl.Items.Clear()

For i As Integer = 0 To N - l ListBoxl

Items.Add(FL(i)) Next

Произвольный элемент в списке можно выбрать по его номеру. Нумерация начинается с нуля. Количество строк в списке определяется с помощью функции Items.Count. Для выбора номера может быть использован компонент NumericUpDown. Номер элемента списка указывается в квадратных скобках. numericUpDownl .Minimum = 0 numericUpDownl .Maximum = listBoxl .Items.Count-l Выбор номера элемента списка может осуществляться с помощью компонента NumericUpDown, а содержимое выбранной таким образом строки ListBox.Items отображаться в Label. Label1.Text = ListBox1.Items(NumericUpDown1.Value).ToString()

Пример программы Приводим ниже текст программы, в которой файл выбирается с помощью класса OpenFileDialog.

```
Imports System.IO
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim fs As FileStream = Nothing
        Dim openFileDialog1 As New OpenFileDialog()
        Dim INI_FLD As String = "C:\Foto\"
        openFileDialog1.Title = "Выбор графического файла"
        openFileDialog1.InitialDirectory = INI_FLD
        openFileDialog1.Filter = "фото jpg|*.jpg"
        openFileDialog1.FilterIndex = 1
        openFileDialog1.RestoreDirectory = True
        openFileDialog1.FileName = ""
        If openFileDialog1.ShowDialog() = DialogResult.OK Then
            Try
                fs = openFileDialog1.OpenFile()
                If (fs IsNot Nothing) Then
                    Label1.Text = openFileDialog1.FileName
                    PictureBox1.Visible = False
                    PictureBox1.Image = New Bitmap(Label1.Text)
                    PictureBox1.SizeMode = PictureBoxSizeMode.Zoom
                    PictureBox1.Visible = True
                End If
            Catch e1 As Exception
                Label1.Text = e1.ToString
            Finally

                fs.Close()
            End Try
        End if
    End Sub
End Class
```

Рис. 10.5. Пример окна программы.

Варианты заданий Вариант 1. Разработать программу, которая позволит выбрать и просмотреть произвольный текстовый файл (*.txt) или графический (*.bmp) из произвольной папки.

Вариант 2. Разработать программу, которая позволит выбрать и просмотреть произвольный графический файл (*.jpg) из произвольной папки.

Вариант 3. Разработать программу, которая позволит выбрать и просмотреть произвольный графический файл (*.bmp) из произвольной папки.

Вариант 4. Разработать программу, которая позволит выбрать и просмотреть произвольный графический файл (*.gif) из произвольной папки.

Вариант 5. Разработать программу, которая позволит получить список текстовых файлов (*.txt) из произвольной папки, а также просмотреть любой файл из списка.

Вариант 6. Разработать программу, которая позволит получить список графических файлов (*.jpg) из произвольной папки, а также просмотреть любой файл из списка.

Вариант 7. Разработать программу, которая позволит получить список графических файлов (*.bmp) из произвольной папки, а также просмотреть любой файл из списка.

Вариант 8. Разработать программу, которая позволит получить список графических файлов (*.gif) из произвольной папки, а также просмотреть любой файл из списка.

Вариант 9. Разработать программу, которая позволит выбрать и просмотреть произвольный текстовый (*.txt) или графический (*.jpg) файл из произвольной папки.

Вариант 10. Разработать программу, которая позволит выбрать и просмотреть произвольный текстовый (*.txt) или графический (*.bmp) файл из произвольной папки.

Вариант 11. Разработать программу, которая позволит выбрать и просмотреть произвольный текстовый (*.txt) или графический (*.gif) файл из произвольной папки.

Вариант 12. Разработать программу, которая позволит получить список текстовых (*.txt) и графических (*.jpg) файлов из произвольной папки, а также просмотреть любой файл из списка.

Вариант 13. Разработать программу, которая позволит получить список текстовых (*.txt) и графических (*.bmp) файлов из произвольной папки, а также просмотреть любой файл из списка.

Вариант 14. Разработать программу, которая позволит получить список текстовых (*.txt) и графических (*.gif) файлов из произвольной папки, а также просмотреть любой файл из списка.

Вариант 15. Разработать программу, которая позволит получить список графических файлов (*.bmp и *.jpg) из произвольной папки, а также просмотреть любой файл из списка.

Порядок выполнения лабораторной работы

- Создать папку с набором тестовых данных.
- Создать новый проект - приложение Windows Forms.
- Вынести на форму необходимые визуальные компоненты.
- Добавить программный код обработки событий. - Компилировать программу.

Лабораторная работа 6
Language Integrated Query

Цель работы: Познакомиться с некоторыми возможностями интегрированного языка запросов.

Интегрированный язык запросов (Language Integrated Query) позволяет оперировать с массивами, XML-документами и другими источниками данных. Операции над данными выполняются с помощью q-операторов (Query-операторы). Для того чтобы использовать LINQ, нужно добавить ссылку на пространства имен ***System.Linq*** и ***System.Linq.Xml.*** LINQ добавляет возможности запросов в Visual Basic, обеспечивает простые и мощные способы обработки всех видов данных. В запросах независимо от типа данных используется унифицированный синтаксис. LINQ позволяет запрашивать данные из базы данных SQL Server, XML, массивов и коллекций в памяти, наборов данных ADO.NET или любого другого локального или удаленного источника данных, который поддерживает LINQ.

Операторы LINQ предоставляют возможности для анализа числовых массивов. В частности, операторы Count, Max, Min, Sum дают количество элементов, максимальное, минимальное значения, а также сумму элементов массива. Например, в следующей строке определяется количество элементов массива D:

Dim c As Integer = D.Count()

В табл. 11.1 представлены примеры простых функций, предназначенных для анализа массива. Таблица 11.1 Функции, предназначенные для анализа массива.

Таблица 11.1 Функции, предназначенные для анализа массива.

Функция	Действие
Average	Среднее арифметическое
Count	Количество элементов
First	Первый элемент
Last	Последний элемент
Max	Максимальный элемент
Min	Минимальный элемент
Sum	Сумма элементов

Более сложные процедуры выполняются с помощью строки запроса, которая имеет следующий формат: Dim результат = From переменная In массив Where условие. Параметр запроса результат содержит те элементы исходного массива, которые удовлетворяют условию. Переменная – это имя переменной, которая участвует в построении условия. Например, следующий запрос выбирает те элементы числового массива, которые больше 10.

```
Dim M1 = From x In M Where x>10
```

Другой запрос выбирает те элементы, которые совпадают с минимальным или максимальным элементом:

```
Dim M2 = From x In M Where x = M.Max Or x = M.Min
```

Отобранные таким образом элементы можно вывести на экран, например в поле ListBox:

```
For Each a in M2
ListBox1.Items.Add(a.ToString)
Next
```

Для упорядочения элементов требуется другая форма запроса: Dim результат = From переменная In массив Order By параметр Здесь параметр может принимать значения Descending (по убыванию) или Ascending (по возрастанию). Например, следующий запрос сортирует элементы по убыванию:

Dim M2 = From x In M Order By Descending Еще одна форма запроса позволяет исключить повторяющиеся элементы.

Dim результат = From переменная In массив Distinct Например, следующий запрос отбирает только оригинальные элементы из массива, игнорируя повторы:

Dim M3 = From x In M2 Distinct Используя q-операторы, легко выполнить какие-либо вычисления для всех элементов массива, например найти долю каждого элемента в процентах.

Пример программы В примере для исходного целочисленного массива выполняется сортировка, фильтр, поиск элементов, определяются минимум, максимум, количество элементов и др. Результат отображается в компоненте ListBox.

```
Public Class Form1
    Private s As String
    Private A() As Integer = {7, 3, 6, 3, 5, 7, 9, 1, 4, 8, 5, 4, 0, 2, 3}
    Private Property b1 As IEnumerable(Of Integer)
    Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        s = "исходный массив = "
        For Each x In A
            s = s + x.ToString + " "
        Next
        ListBox1.Items.Add(s)
    End Sub

    Private Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
        s = "по возрастанию = "
        Dim b = From c In A Order By c Ascending
        For Each d In b
            s = s + d.ToString + " "
        Next
        ListBox1.Items.Add(s)
        s = "по убыванию = "
        b = From c In A Order By c Descending
        For Each d In b
            s = s + d.ToString + " "
        Next
        ListBox1.Items.Add(s)
        s = "не мин и не макс = "
        b1 = From c In A Where c > A.Min And c < A.Max
        For Each d In b1
            s = s + d.ToString + " "
        Next
        ListBox1.Items.Add(s)
        s = "неповторяющиеся эл. = "
        b1 = From c In A Distinct
        For Each d In b1
            s = s + d.ToString + " "
        Next
        ListBox1.Items.Add(s)
        s = "больше 6, меньше 4 = "
        b1 = From c In A Where c > 6 Or c < 4
        For Each d In b1
            s = s + d.ToString + " "
        Next
        ListBox1.Items.Add(s)
        ListBox1.Items.Add("min = " + A.Min.ToString)
        ListBox1.Items.Add("max = " + A.Max.ToString)
        ListBox1.Items.Add("avg = " + A.Average.ToString)
        ListBox1.Items.Add("first = " + A.First.ToString)
        ListBox1.Items.Add("last = " + A.Last.ToString)
        ListBox1.Items.Add("sum = " + A.Sum.ToString)
        ListBox1.Items.Add("count = " + A.Count.ToString)
    End Sub
End Class
```

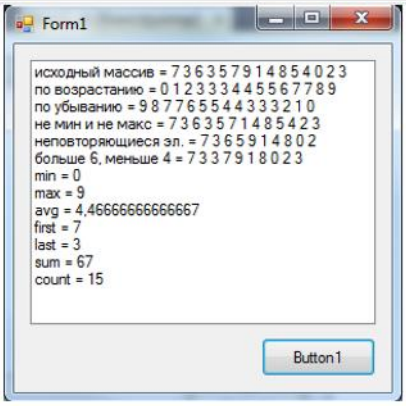


Рис10.1 Код и внешний вид программы.

Обработка массива строк

С массивом строк можно выполнять процедуры, аналогичные тем, что действуют с числами. Например, максимальный элемент в массиве строк – это тот, который должен

находиться на первом месте по порядку возрастания (строка начинается с буквы А), а минимальный элемент – это тот, который стоит на последнем месте по алфавиту. Пусть требуется в массиве строк S найти все элементы, содержащие подстроку S1. Для получения результатов создается запрос со словом Where. Фильтрация строк массива выполняется с помощью функции Contains.

Dim FS = From c In S Where c.Contains(S1)

Если требуется в массиве строк S найти все элементы, совпадающие со строкой S1, то фильтрация выполняется командами Where и Equals.

Dim FS = From c In S Where c.Equals(S1)

Отобразить результат запроса, например, в поле ListBox можно с помощью короткого цикла:

For Each a in FS
ListBox1.Items.Add(a)
Next

С помощью запроса можно выполнить некоторое действие для каждого элемента массива. Например, запрос в следующей строке получает новый массив, в котором элементы равны удвоенным элементам исходного массива.

Dim B = From x In A Select x * 2

Пример программы

В примере исходные данные отображаются в одном ListBox, а результаты запросов выводятся в другом ListBox. Выполняются сортировка и несколько вариантов фильтра. Один из вариантов предполагает, что пользователь будет вводить кусок строки в поле TextBox, а после каждого изменения этого поля будет выполняться запрос.

```
Public Class Form1
    Dim m0() As [String] = {"Ву", "Ли", "По", "Ас", "Мо", "Шу", "Лу",
    "Ка", "Яр", "Ал", "До", "Ву", "Ус", "Шу"}

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Label1.Text = "исходный массив"
        For Each f As Object In m0
            ListBox1.Items.Add(f)
        Next
        Label2.Text = ""
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Label2.Text = "без повторов"
        Dim b = From a In m0 Distinct
        ListBox2.Items.Clear()
        For Each f As Object In b
            ListBox2.Items.Add(f)
        Next
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        Label2.Text = "содержит у"
        ListBox2.Items.Clear()
        Dim c = From a In m0 Where a.Contains("у")
        For Each f As Object In c
            ListBox2.Items.Add(f)
        Next
    End Sub
```

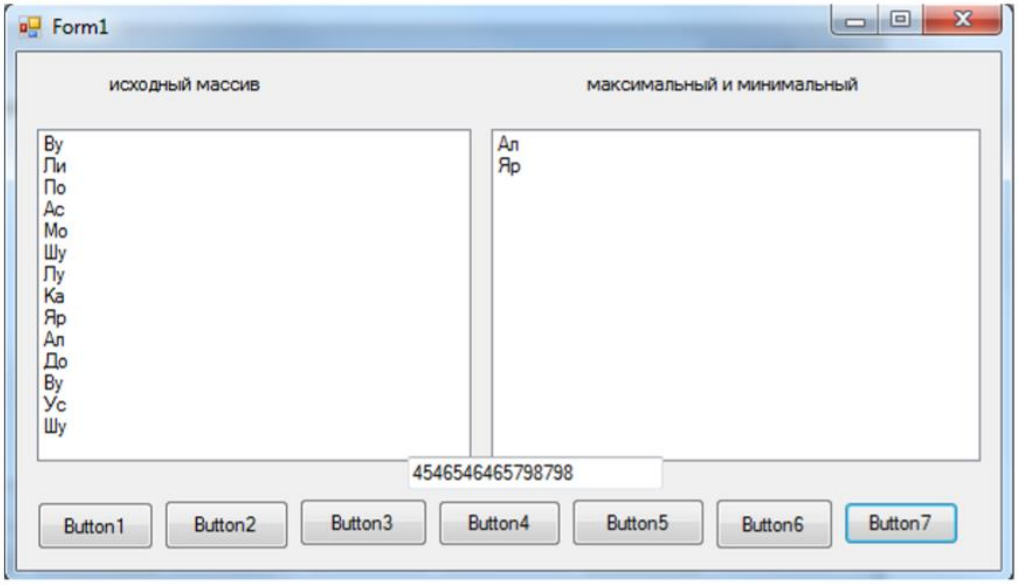
```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    Label2.Text = "содержит у и У"
    ListBox2.Items.Clear()
    Dim c = From a In m0 Where a.Contains("у") Or a.Contains("У")
    For Each f As Object In c
        ListBox2.Items.Add(f)
    Next
End Sub

Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    Label2.Text = "Не содержит у и У"
    ListBox2.Items.Clear()
    Dim c = From a In m0 Where Not (a.Contains("у") Or _
    a.Contains("У"))
    For Each f As Object In c
        ListBox2.Items.Add(f)
    Next
End Sub

Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
    Label2.Text = "Select - результат"
    ListBox2.Items.Clear()
    Dim d = From a In m0 Select a + "10"
    For Each f As Object In d
        ListBox2.Items.Add(f)
    Next
End Sub

Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button6.Click
    Label2.Text = "первый и последний "
    ListBox2.Items.Clear()
    ListBox2.Items.Add(m0.First)
    ListBox2.Items.Add(m0.Last)
End Sub

Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChange
    If Me.Text.Length > 0 Then
        Label2.Text = "Произвольный выбор"
        ListBox2.Items.Clear()
        Dim d = From a In m0 Where a.Contains(TextBox1.Text)
        For Each f In d
            ListBox2.Items.Add(f)
        Next
    End If
End Sub
```



Исходный код и внешний вид программы

Задания для лабораторной работы

Разработать программу, выполняющую анализ массива строк. Исходные данные подготовить в соответствии с вариантом задания и разместить в текстовом файле. Результат отобразить на экране, а также разместить в текстовом файле.

Вариант 1. Дан список фамилий. Сортировать его по возрастанию. Найти фамилии, содержащие произвольную подстроку.

Вариант 2. Дан список фамилий. Сортировать его по убыванию. Найти фамилии, содержащие произвольную подстроку без учета регистра.

Вариант 3. Дан список фамилий. Сортировать его по возрастанию. Найти фамилии, совпадающие с указанной произвольно фамилией.

Вариант 4. Дан список фамилий. Сортировать его по убыванию. Найти фамилии, совпадающие с указанной произвольно фамилией без учета регистра.

Вариант 5. Даны 2 списка фамилий. Сортировать их по возрастанию, затем объединить.

Вариант 6. Даны 2 списка фамилий. Сортировать их по убыванию, затем объединить.

Вариант 7. Даны 2 списка фамилий. Объединить фамилии в один список, затем сортировать их по возрастанию.

Вариант 8. Даны 2 списка фамилий. Объединить фамилии в один список, затем сортировать их по убыванию.

Лабораторная работа 7
Файлы произвольного доступа

Кроме базовых типов данных, таких как Integer, Long и т.п., VB поддерживает также типы данных, определяемые пользователем. Они могут быть созданы как на основе базовых типов данных, так и на основе типов ранее определенных пользователем. Пользовательский тип данных, называемый структурой, широко применяются при построении файлов произвольного доступа.

Для определения пользовательского типа (структуры) данных используется ключевое слово Structure:

[Public/Private]Structure Имя_типа
Dim/Public/Private Элемент1 **As** Тип
Dim/Public/Private [Элемент2 **As** Тип]
End Structure

Структура не может быть объявлена внутри процедуры или в функции. Она может быть объявлена только в начале проекта, а также в форме или в модуле до первой процедуры. Определив собственный тип данных, Вы можете использовать его для объявления переменных этого типа. Эти переменные могут быть локальными, переменными области формы или модуля, а также глобальными. Переменную пользовательского типа называют записью. Отдельные компоненты этой переменной называют полями записи. Переменная пользовательского типа является структурированной. Она подобно массиву включает в свой состав отдельные элементы, но в отличие от массива, ее элементы могут иметь разный тип:

```
Public Class Form1
    Structure T
        Dim Название As String
        Dim Цена As Decimal
        Dim Код As Long
    End Structure
    Dim Инструмент As T
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Инструмент.Название = "Отвертка"
        Инструмент.Цена = 120
    End Sub
End Class
```

В этом примере определяется тип данных T. Затем объявляется переменная Инструмент типа T, а конкретные значения составляющих этой переменной устанавливаются в процедуре Buttonl_Click. Доступ к элементам переменной пользовательского типа осуществляется, по аналогии с доступом к свойствам, путем указания точки после имени переменной. При этом переменные одинакового типа можно присваивать не поэлементно, а напрямую.

Запись в файл и чтение из файла
FilePut(file_number, value[, record_number]),
FileGet(file_number, value[, record_number])

Функции используются для записи и чтения записей из файла с произвольным доступом. Каждой из них может быть передан номер записи, с которой Вы хотите работать. В аргументе record_number задается номер записи, а в аргументе value – переменная, содержащая записываемую в файл запись или принимающая запись, которая считывается из файла.

Fргумент record_number не обязателен; если он не задан, будет записываться или считываться текущая запись. После записи или чтения очередной записи текущей становится следующая запись. Таким образом, вызвав функцию FilePut() десять раз подряд без указания номера записи, Вы последовательно создадите или перезапишете первые десять записей файла с произвольным доступом. Точно так же, вызвав десять раз подряд функцию FileGet() без указания номера записи, Вы последовательно прочитаете первые десять записей файла. Несколько слов о принципах обработки файлов с произвольным доступом. Предположим, Вы хотите создать файл с произвольным

доступом для хранения списка товаров. Пусть информация о каждом из товаров содержится в структуре Товар, определяемой следующим образом:

```
Public Class Form1
    Structure Товар
        Dim Код As String
        Dim Название As String
        Dim Цена As Decimal
    End Structure
End Class
```

Структура Товар будет использоваться для хранения информации о товаре перед ее записью в файл. Начнем с объявления переменой типа Товар:

```
Dim тов As Товар
```

Теперь можно присвоить значения полям структуры Тов:

```
тов.Код = "TV00180-A"
тов.Название = "SONY Trinitron TV"
тов.Цена = 799.99
```

Для записи данных, хранящихся в переменной Тов, в файл с произвольным доступом, мы воспользуемся функцией FilePut(). Конечно, файл сначала нужно создать с помощью инструкций:

```
fn = FreeFile()
FileOpen(fn, "c:\products.dat", OpenMode.Random)
```

Как видите, последний аргумент, в котором задается длина записи, здесь опущен. Записи содержат строки, следовательно, имеют переменную длину. Информация о длине каждой строки хранится вместе с самой строкой, так что определять длину каждой записи не понадобится. Далее переменная Тов записывается в файл с помощью инструкции:

```
FilePut(fn, тов)
```

Обратите внимание, что номер записи, в которой сохраняются данные, не указан. В этом случае выполняется обработка текущей записи. После окончания выполнения операции текущей становится следующая запись. Вы можете изменить значения полей и сохранить в файле следующую запись с помощью точно такой же инструкции. После записи всех необходимых данных файл закрывается с помощью инструкции

```
FileClose(fn)
```

Для того чтобы прочитать данные из файла, нужно открыть его с применением той же функции FileOpen(), с помощью которой мы открывали его для записи данных:

```
fn = FreeFile()
FileOpen(fn, "c:\products.dat", OpenMode.Random)
```

Затем обычно выполняется цикл чтения записей. Следующий фрагмент кода демонстрирует, как записать в файл с произвольным доступом записи разной длины с помощью функции FilePut() и как их прочитать с помощью функции FileGet(). Прежде всего, необходимо добавить в форме до объявления первой процедуры следующее объявление структуры:

```
Public Class Form1
    Structure Товар
        Dim Код As String
        Dim Название As String
        Dim Цена As Decimal
    End Structure
```

Затем следует поместить на форму кнопку и ввести в обработчике ее события Click такие инструкции:

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim fn As Integer
        Dim Тов As Товар
        Тов.Код = "TV00180-A"
        Тов.Название = "SONY Trinitron TV"
        Тов.Цена = 799.99
        fn = FreeFile()
        FileOpen(fn, "c:\products.dat", OpenMode.Random)
        FilePut(fn, Тов)
        Тов.Код = "TV-RCA"
        Тов.Название = "This is an RCA Trinitron TV"
        Тов.Цена = 699.99
        FilePut(fn, Тов)
        Тов.Код = "TV810X"
        Тов.Название = "Real cheap BIG Trinitron TV"
        Тов.Цена = 399.99
        FilePut(fn, Тов)
        FileClose(fn)
        fn = FreeFile()
        FileOpen(fn, "c:\products.dat", OpenMode.Random)
        FileGet(fn, Тов, 2)
        FileClose(fn)
        Console.WriteLine(Тов.Код)
        Console.WriteLine(Тов.Название)
        Console.WriteLine(Тов.Цена)
    End Sub
End Class
```

Коды и описания различных товаров представляют собой строки разной длины. Первая часть приведенного кода записывает три записи в файл с произвольным доступом и закрывает его. Вторая часть кода считывает из файла вторую запись и выводит ее поля в окне Output. Итак, функции FilePut() и FileGet() позволяют создавать записи со строками, которые имеют переменную длину. Функции FilePut() и FileGet() берут на себя ответственность за работу с такими строками, предоставляя Вам доступ к данным файлов с произвольным доступом и используя запись в качестве базовой единицы длины.

Функция Seek

Seek(file_number[, position]) Функция Seek() возвращает текущую позицию указателя ввода-вывода в заданном файле, но лишь при условии, что вызвана без аргумента position. Для файла с произвольным доступом значением функция возвращает номер последней записи, из которой производилось чтение или в которую производилась запись. Но если речь идет о двоичных файлах, это номер последнего прочитанного или записанного байта. Задав в функции Seek() аргумент position, можно установить текущую позицию ввода/вывода. Например, для того чтобы в файле с произвольным доступом перейти к началу третьей записи, нужно выполнить такую инструкцию:

```
Seek(fNum, 3)
```

Проект «Файл произвольного доступа»

Создайте новый проект, который для файла произвольного доступа должен позволять открытие файла, создание файла, отображение его содержимое в текстовом поле, а также

добавление в него новых записей. Файлы произвольного доступа обычно применяют для хранения структурированной информации. Все записи файла должны иметь одинаковую структуру. Применим файл для хранения ведомости совершенных продаж. Каждая запись ведомости содержит для проданного следующие поля:

код;

название;

цена;

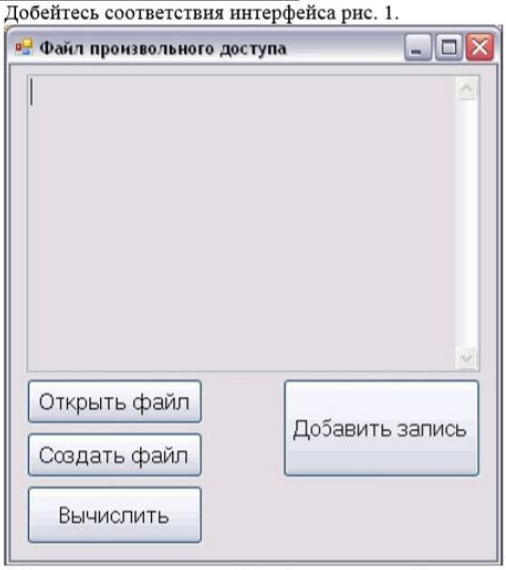
количество;

дата.

Приступите к разработке нового проекта. 1. Создайте новый проект с именем ФайлПроизвольногоДоступа, следуя приложению 1. 2. Разместите на форме текстовое поле и три кнопки. Установите значения свойств Text управляющих элементов:

Button1.Text	Открыть файл
Button2.Text	Создать файл
Button3.Text	Добавить запись

Добейтесь соответствия интерфейса рис. 1.



3. Введите внутри класса Form1, но до первой процедуры формы код:

```
Structure Ведомость
    Dim Код As String
    Dim Название As String
    Dim Цена As Decimal
    Dim Количество As Integer
    Dim Дата As Date
End Structure
Dim ИмяФайла As String, Запись As Ведомость
Dim k As Integer
```

Здесь в строках 1 – 7 объявляется структура Ведомость. Затем в строках 8 и 9 объявляется переменные, действующие во всех процедурах формы, в том числе переменная Запись типа Ведомость.

5. Подпрограмма Button1_Click открывает существующий файл произвольного доступа для чтения и отображает содержащуюся в нем информацию в текстовом поле TextBox1. При отображении записи в текстовом поле с помощью метода ToString выполняется преобразование в строку значений отдельных компонент записи, тип которых отличается от строкового. Дважды щелкните на кнопке Button1 и введите код тела подпрограммы Button1_Click (без начальной строки и конечной строки, которые созданы системой):

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    TextBox1.Clear()
    ИмяФайла = InputBox( _
    "Введите полное имя файла." & " Пример: d:\ИмяРабочейПапки\ИмяФайла")
    If ИмяФайла = "" Then
        MsgBox("Не задано имя файла. Повторите!")
        Exit Sub
    End If
    k = FreeFile()
    Try
        FileOpen(k, ИмяФайла, OpenMode.Random, OpenAccess.Read)
    Catch
        MsgBox( _
        "Ошибка при открытии файла " & ИмяФайла)
        ИмяФайла = ""
        FileClose(k)
        Exit Sub
    End Try
    Do Until EOF(k)
        FileGet(k, Запись)
        TextBox1.AppendText(Запись.Код.ToString & vbTab)
        TextBox1.AppendText(Запись.Название & vbTab)
        TextBox1.AppendText(Запись.Цена.ToString & vbTab)
        TextBox1.AppendText(Запись.Количество.ToString & vbTab)
        TextBox1.AppendText(Запись.Дата.ToString)
        TextBox1.AppendText(vbCrLf)
    Loop
    FileClose(k)
End Sub
```

В строке 2 этой подпрограммы задается полное имя файла, который затем будет открыт. Строка 3 проверяет, если файла является пустой строкой (пользователь забыл его ввести), то работа подпрограммы заканчивается. В строке 9 выполняется открытие файла для произвольного доступа для чтения. При открытии файла возможна генерация исключения. Строки 8 – 15 предусматривают обработку исключения, предупреждают о его возникновении и прекращают выполнение подпрограммы. В этом случае следует разобраться в причине возникновения ошибки и повторным нажатием на кнопке Button1 снова попытаться открыть файл. Строки 16 – 24 обеспечивают отображение в текстовом поле TextBox1 содержащейся в файле информации. 5. Подпрограмма Button2_Click позволяет создать новый файл с заданным именем. Если этот файл уже существует, то он будет открыт для записи и закрыт. Если же такого файла нет, то он будет создан и закрыт. Указатель записи-чтения устанавливается в позицию 1. Дважды щелкните на кнопке Button2 и введите код тела подпрограммы Button2_Click:

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    TextBox1.Clear()
    ИмяФайла = InputBox( _
    "Введите полное имя файла. " & "Пример: d:\ИмяРабочейПапки\ИмяФайла")
    If ИмяФайла = "" Then
        MsgBox("Не задано имя файла. Повторите!")
        Exit Sub
    End If
    k = FreeFile()
    Try
        FileOpen(k, ИмяФайла, OpenMode.Random, OpenAccess.Write)
        FileClose(k)
    Catch
        MsgBox("Ошибка при открытии файла " & ИмяФайла)
        ИмяФайла = ""
    End Try
End Sub
```

В строках 3 – 6 проверяется, что пользователь не забыл ввести имя создаваемого файла. В строке 9 файл открывается для записи. Если создаваемый файл не существовал, то он будет создан. В следующей строке 10 файл закрывается. При открытии файла проверяется корректность его имени. Возможное исключение, возникающее при открытии файла, обрабатывается инструкциями в строках 8 – 14.

6. Подпрограмма Button3_Click обеспечивает запись в заданную позицию файла произвольного доступа или в конец этого файла, а также отображение всего файла в текстовом поле TextBox1 после выполнения записи. Дважды щелкните на кнопке Button3 и введите код тела подпрограммы Button3_Click:

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    Dim p As Long, s As Ведомость = Nothing
    If Имяфайла = "" Then
        MsgBox("Сначала файл нужно открыть или создать!")
        Exit Sub
    End If
    Try
        p = InputBox("Введите номер записи. Если запись" & " нужно добавить в конец файла, то введите 0")
    Catch
        MsgBox("Строка не может трактоваться как число!")
        Exit Sub
    End Try
    If p < 0 Then
        MsgBox("Некорректный номер записи = " & p.ToString & "?")
        Exit Sub
    Else
        Try
            Запись.Код = InputBox("Введите код товара")
            Запись.Название = InputBox("Введите название товара")
            Запись.Цена = InputBox("Введите цену товара")
            Запись.Количество = InputBox("Введите количество товара")
            Запись.Дата = InputBox("Введите дату в формате дд.мм.гг")
        Catch
            MsgBox("Некорректные данные. Повторите!")
            Exit Sub
        End Try
        k = FreeFile()
        FileOpen(k, Имяфайла, OpenMode.Random, OpenAccess.ReadWrite)
        If p = 0 Then
            Do Until EOF(k)
                FileGet(k, s)
            Loop
               FilePut(k, Запись)
               TextBox1.AppendText(Запись.Код.ToString & vbTab)
               TextBox1.AppendText(Запись.Название & vbTab)
               TextBox1.AppendText(Запись.Цена.ToString & vbTab)
               TextBox1.AppendText(Запись.Количество.ToString & vbTab)
               TextBox1.AppendText(Запись.Дата.ToString)
               TextBox1.AppendText(vbCrLf)
           Else
               FilePut(k, Запись, p)
               TextBox1.Clear()
               Seek(k, 1)
               Do Until EOF(k)
                   FileGet(k, Запись)
                   TextBox1.AppendText(Запись.Код.ToString & vbTab)
                   TextBox1.AppendText(Запись.Название & vbTab)
                   TextBox1.AppendText(Запись.Цена.ToString & vbTab)
                   TextBox1.AppendText(Запись.Количество.ToString & vbTab)
                   TextBox1.AppendText(Запись.Дата.ToString)
                   TextBox1.AppendText(vbCrLf)
               Loop
           End If
           FileClose(k)
       End If
   End Sub
End Class
```

В строках 2 – 5 проверяется, что имя файла не является пустой строкой. Это означает, что файл ранее был открыт или создан. В строке 7 задается номер позиции файла, в которую предполагается произвести запись. Строки 6 – 11 выполняют обработку исключения, которое может возникнуть при задании номера позиции записи. В строке 12 проверяется корректность заданного номера позиции записи. Он не должен быть отрицательным. Если новую запись следует добавить в конец файла, то следует задать номер позиции, равный нулю. В строках 17 – 21 тела подпрограммы задаются значения отдельных компонент новой записи. Строки 16 – 25 обеспечивают обработку исключения, которое может возникнуть изза ошибок пользователя при задании значений компонент записи. В строке 27 выполняется открытие для записи и чтения файла произвольного доступа с заданным именем. Если задан номер позиции, равный нулю, то строки 29 – 38 обеспечивают запись в конец файла. А также отображение сделанной в файле записи в текстовом поле TextBox1. При этом строки 29 – 31 предназначены исключительно для позиционирования указателя записи-чтения в конце файла. Непосредственно запись в

файл выполняется в строке 32. В строке 40 выполняется запись в заданную позицию файла, когда заданный номер позиции записи положителен. В строке 42 устанавливается позиция записи-чтения файла, равной 1. Затем вся информация из файла считывается в предварительно очищенное строкой 41 текстовое поле TextBox1 (строки 43 – 51).

7. Сохраните проект и проверьте его работу. Для этого с помощью кнопки Создать файл создайте в своей рабочей папке файл с именем f2.dat. Затем с помощью кнопки Добавить запись запишите в файл не менее 5 записей. При этом все записи файла, включая только что добавленную запись, должны отобразиться в текстовом поле. Наконец проверьте, как файл открывается кнопкой Открыть файл. После нажатия на эту кнопку в текстовом поле должны отобразиться все записи файла.

8. Доработайте проект. Добавьте на форму надпись и еще одну кнопку. При нажатии на эту кнопку должны быть вычислена и отображена в надписи сумма продаж, выполненных в заданную дату. В текстовом поле должны быть отображены только те записи файла, компонента Дата которых совпадает с заданной датой.

9. Сохраните проект и проверьте его работу.

10. Попробуйте ответить на вопросы для контроля. Покажите результаты работы преподавателю.

11. Закройте приложение и удалите на диске d свою рабочую папку

Вопросы для контроля

1. Какие типы доступа к файлам поддерживаются в VB.NET?
2. Какие типы доступа к файлу делают возможным замену отдельной записи без перезаписи всего файла?
3. Для чего применяется функция FreeFile?
4. Каково назначение функции FileOpen?
5. Каково назначение функции FileClose?
6. Каково назначение функции FilePut?
7. Каково назначение функции FileGet?
8. Для чего применяется функция EOF?
9. Для чего применяется функция Seek?
10. Как можно изменить одну запись в файле произвольного доступа?

Лабораторная работа №8
Построение графиков и диаграмм

Машинная графика всегда являлась самым популярным упражнением программистов. При наличии готовых к использованию компонентов графика стала вполне доступной. Хорошим заданием для ее освоения является построение графика таблично заданной функции. Для построения рисунка задаем процедуру Grafik обработки события Paint. Эта процедура содержит код построения рисунка, который будет выполняться при наступлении некоторых событий.

Me.Paint = new PaintEventHandler(Grafik)

Благодаря этой функции после изменения размеров окна формы рисунок может быть обновлен командой

Me.Refresh()

Для вывода графических объектов необходимо задать графическую поверхность G_P, в которой будет отображаться график.

Dim G_P As Graphics

G_P = e.Graphics()

На указанной поверхности должна отображаться и вспомогательная информация, в частности заголовок и подписи данных. Необходимо описать шрифт этих элементов графика, например, в строковой переменной:

Dim hF As F = New Font("Tahoma", 14)

Dim H As String = "График"

Для того чтобы оценить размеры области для вывода, используем свойство ClientSize.Width, означающее ширину рабочей области формы. Свойство MeasureString().Width означает ширину строки текста с учетом выбранного шрифта.

Dim w As Integer = G_P.MeasureString(H, hF).Width

Dim x As Integer = (Me.ClientSize.Width - w) / 2

Метод DrawString предназначен для вывода строки текста на графическую поверхность. G_P.DrawString(H, hF, System.Drawing.Brushes.Black, x, 5) Важной частью программы построения графика является определение шага по X, а также настройка масштаба. Для определения шага нужно размер рабочей области формы Me.ClientSize.Width разделить на количество точек графика. Например, если массив D содержит значения функции, а количество его элементов содержится в свойстве D.Length, то шаг с учетом отступа от границы формы равен Dim s_x As Integer = (Me.ClientSize.Width -40)/d.Length Для определения масштабного множителя определяются максимальное и минимальное значения элементов массива. Масшабный множитель вычисляется на основании вертикального размера рабочей области формы Me.ClientSize.Height с учетом отступа от границ: Dim m As Integer = (Me.ClientSize.Height - 100) / (max-min)

Координата X определяется путем умножения номера точки i на постоянный шаг, а Y вычисляется в соответствии с масшабом и с учетом отступа от границы формы: X = 8 + i * S_X

Y = Me.ClientSize.Height - 20 – m * d(i)

Экранные координаты – это целые числа. Точки изображаются прямоугольниками DrawRectangle, а график – отрезками прямой DrawLine. Над каждой точкой выводится соответствующее числовое значение.

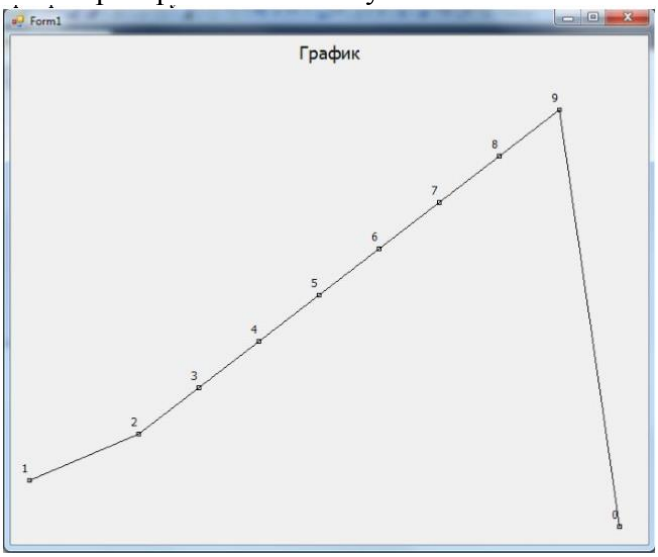
G_P.DrawRectangle(System.Drawing.Pens.Black,x2-2,y2-2,4,4)

G_P.DrawLine(System.Drawing.Pens.Black, x1, y1, x2, y2)

G_P.DrawString(Convert.ToString(d(i)), dF, _ System.Drawing.Brushes.Black, x2 - 10, y2 - 20)

Пример программы

Исходные данные – это 10 вещественных чисел, хранящихся в 10 строках текстового файла. Для использования методов ввода данных из текстового файла в начале программы делается ссылка на пространство имен System.IO.



```
Imports System.IO
Public Class Form1
    Private d(10) As Double
    Private Sub Form1_Paint(ByVal sender As Object, ByVal e As System.Windows.Forms.PaintEventArgs) Handles MyBase.Paint
        Dim g As Graphics ' графическая поверхность
        g = e.Graphics() ' подписи данных
        Dim dF As Font = New Font("Tahoma", 9) ' заголовок
        Dim hF As Font = New Font("Tahoma", 14)
        Dim H As String = "График"
        Dim w As Integer = g.MeasureString(H, hF).Width
        Dim x As Integer = (Me.ClientSize.Width - w) / 2
        g.DrawString(H, hF, System.Drawing.Brushes.Black, x, 7) ' шаг по X
        Dim sw As Integer = (Me.ClientSize.Width - 40) / (d.Length - 1) ' размеры реальной области
        Dim max As Double = d(1)
        Dim min As Double = d(1)
        For i As Integer = 1 To d.Length - 1
            If d(i) > max Then max = d(i)
            If d(i) < min Then min = d(i)
        Next
        Dim m As Integer = (Me.ClientSize.Height - 100) / (max - min) ' масштабный множитель
        Dim x1, y1, x2, y2 As Integer ' первая точка графика
        x1 = 20
        y1 = Me.ClientSize.Height - 20 - m * d(1)
        g.DrawRectangle(System.Drawing.Pens.Black, x1 - 2, y1 - 2, 4, 4)
        g.DrawString(Convert.ToString(d(1)), dF, System.Drawing.Brushes.Black, x1 - 10, y1 - 20) ' остальные точки
        For i As Integer = 2 To d.Length - 1
            x2 = 8 + i * sw
            y2 = Me.ClientSize.Height - 20 - m * d(i)
            g.DrawRectangle(System.Drawing.Pens.Black, x2 - 2, y2 - 2, 4, 4)
            g.DrawLine(System.Drawing.Pens.Black, x1, y1, x2, y2)
            g.DrawString(Convert.ToString(d(i)), dF, _
            System.Drawing.Brushes.Black, x2 - 10, y2 - 20)
            x1 = x2
            y1 = y2
        Next
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        Dim fs As FileStream ' подготовка исходных данных
        Dim FN As String = "C:\d.txt" ' путь в текущий каталог
        Dim i As Integer = 0 ' поток для чтения
        Try
            fs = New FileStream(FN, FileMode.Open, FileAccess.Read) ' читаем числа в массив
            Dim sr As New StreamReader(fs)
            Dim t As String = sr.ReadLine()
            While i < d.Length - 1
                i = i + 1
                d(i) = Convert.ToDouble(t)
                t = sr.ReadLine()
            End While
            sr.Close()
            Dim Paint = New PaintEventHandler(AddressOf Form1_Paint) ' задаем функцию обработки события Paint
        Catch ex As FileNotFoundException
            MessageBox.Show(ex.ToString())
        Catch ex As Exception
            MessageBox.Show(ex.ToString())
        Finally
            fs.Close()
        End Try
    End Sub
End Class
```

Задание на тему «Построение графика»

 Разработать программу, вычерчивающую график. Исходные данные должны извлекаться из текстового файла. Количество точек – произвольно.

Порядок выполнения лабораторной работы Создать папку с набором тестовых данных.

 Создать новый проект – приложение Windows Forms.

 Вынести на форму необходимые визуальные компоненты.

 Добавить программный код обработки событий.

 Компилировать программу. Тестировать программу. Создать документ с описанием работы программы.

Лабораторная работа №9
Работа с принтерами

Цели работы:

 научится печатать графические изображения;

 научится печатать текст;

 научится печатать многостраничные документы;

 научится использовать диалоговые окна Print (Печать), Page Setup (Параметры страницы) и Print Preview (Предварительный просмотр).

Использование класса PrintDocument

Большинство приложений для Microsoft Windows позволяют пользователям печатать созданные ими документы. В том, что касается возможностей печати в программах, Visual Basic .NET значительно превосходит Visual Basic 6, хотя новая функциональность дается не даром. Получить вывод на печать в программах на Visual Basic .NET - нетривиальная задача, и методы, используемые вами, зависят от типа и количества генерируемого вывода для печати. Однако во всех классах базовым механизмом, который управляет печатью в Visual Basic .NET, является класс PrintDocument, который вы можете использовать в проекте, добавив в форму элемент управления PrintDocument, или определив его программно с помощью нескольких строк кода на Visual Basic. Класс PrintDocument предоставляет для печати текста и графики несколько полезных объектов. Объект PrinterSettings содержит настройки принтера по умолчанию, объект PageSettings содержит настройки печати для конкретной страницы, а объект PrintPageEventArgs содержит событийную информацию о печатаемой странице. Класс PrintDocument расположен в пространстве имен System.Drawing.Printing. Если вы добавляете в форму элемент управления PrintDocument, некоторые из объектов класса PrintDocument автоматически встраиваются в ваш проект, но вам по-прежнему требуется добавить в верхнюю часть кода формы следующий оператор Imports:

```
Imports System.Drawing.Printing
```

Он определяет PrintPageEventArgs и другие значения. Чтобы узнать, как использовать класс PrintDocument в вашей программе, выполните следующее упражнение. В нем вы научитесь добавлять в ваш проект элемент управления PrintDocument и использовать его для печати графического файла. Использование элемента управления PrintDocument

1. Запустите Microsoft Visual Studio . В среде разработки Visual Studio появится пустая форма.

2. Используйте элемент управления Label и нарисуйте в верхней части формы метку. Сделайте метку достаточно широкой, чтобы отображать строку, содержащую указания для пользователя.

3. Используйте элемент управления TextBox и нарисуйте под меткой текстовое поле. Это поле будет использоваться для ввода имени графического файла, который вы хотите открыть. Для этого будет достаточно однострочного текстового поля.

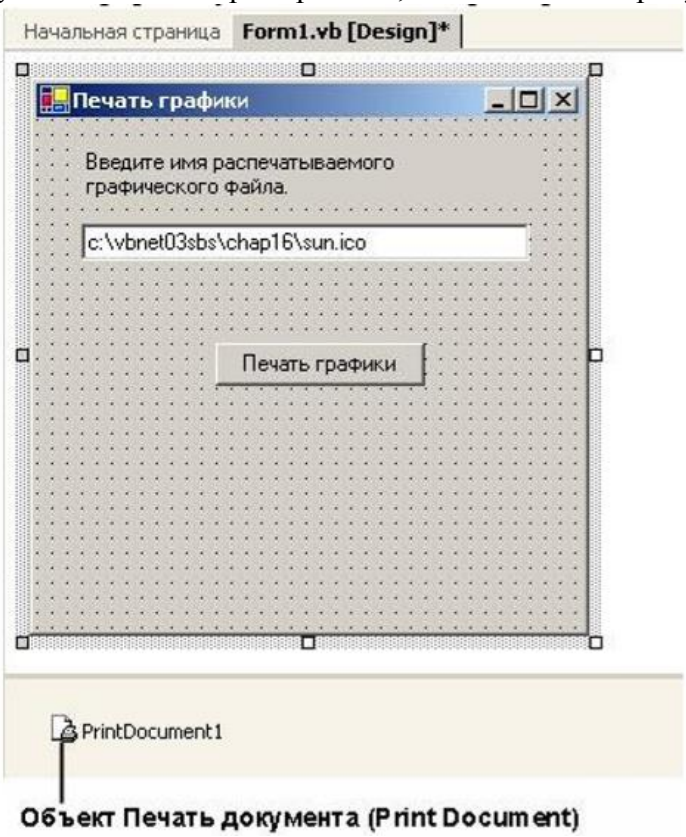
4. Используйте элемент управления Button и нарисуйте под объектом текстового поля кнопку. Эта кнопка будет печатать графический файл. Теперь добавьте в вашу форму элемент управления PrintDocument.

5. Прокрутите вниз панель Windows Forms окна Области элементов, пока не увидите элемент управления PrintDocument, а затем сделайте на нем двойной щелчок мышью. Аналогично элементу управления Timer, элемент управления PrintDocument во время выполнения программы не отображается, так что при его создании он помещается в область компонент под формой. Теперь ваш проект имеет доступ к классу PrintDocument и его объектам печати.

6. Установите для объектов формы следующие свойства:

Объект	Свойство	Установка
Label1	Text	Введите имя распечатываемого графического файла
TextBox1	Text	"путь к файлу"
Button1	Text	Печать графики
Form1	Text	Печать графики

7. Ваша форма будет выглядеть примерно так, как показано на рисунке ниже.



8. Теперь добавьте код программы, необходимый для печати графического файла (растрового изображения, значка, метафайла, файла JPEG и т.д.).

9. Сделайте двойной щелчок мышью на кнопке Печать графики. В Редакторе кода появится процедура события Button1_Click.

10. Прокрутите код формы в самый верх, а затем введите следующий оператор программы:

```
Imports System.Drawing.Printing
```

Этот оператор Imports объявляет пространство имен System.Drawing.Printing, которое требуется для определения объекта PrintPageEventArgs в процедуре PrintGraphic. Процедура PrintGraphic будет добавлена на одном из следующих шагов. (Другие объекты PrintDocument получат свои определения из элемента управления PrintDocument.)

11. Теперь снова прокрутите код вниз до процедуры события Button1_Click, а затем введите следующий код программы:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    'Печать с использованием обработчика ошибок для перехвата проблем
    Try
        AddHandler PrintDocument1.PrintPage, AddressOf Me.PrintGraphic
        PrintDocument1.Print()  'печать графики
    Catch ex As Exception       'перехват исключения печати
        MessageBox.Show("Извините, возникли проблемы с печатью", ex.ToString())
    End Try
End Sub
```

Этот код использует оператор AddHandler, который указывает, что при возникновении события PrintPage объекта PrintDocument1 должен вызываться обработчик события PrintGraphic. Он обрабатывает системные события, которые с технической точки зрения не являются ошибками, а представляют собой ключевые действия в жизненном

цикле объекта. В данном случае указанный обработчик события связан со службами печати, и запрос к нему содержит конкретную информацию о печатаемой странице, текущих установках принтера и других атрибутах класса PrintDocument. Технически, оператор AddressOf используется для указания обработчика события PrintGraphic, определяя его внутренний адрес и сохраняя его. Оператор AddressOf неявно создает объект, известный как делегат.

При возникновении события он передает вызовы в соответствующий обработчик события. Третья строка этого кода использует для отправки запроса на печать в процедуру события PrintGraphic, которую вы создадите в следующем шаге, метод Print объекта PrintDocument1. Заметьте, что в блоке Catch я использую несколько иной синтаксис. Здесь объявляется переменная ex типа Exception, которая содержит подробное сообщение о любой возникающей ошибке. Использование типа Exception является еще одним способом получения условий, которые привели к возникновению ошибки.

12. Прокрутите процедуру события Button1_Click вверх до области глобальных объявлений, расположенной под меткой "Windows Form Designer generated code" ("Код, автоматически созданный конструктором форм Windows"). Введите следующее объявление процедуры:

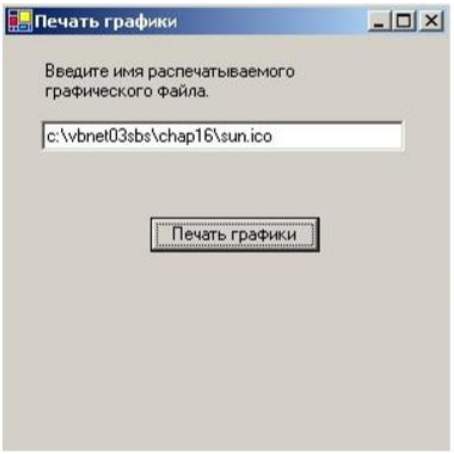
```
Private Sub PrintGraphic(ByVal sender As Object, ByVal ev As PrintPageEventArgs)
    '  Создаем графику с помощью DrawImage
    ev.Graphics.DrawImage(Image.FromFile(TextBox1.Text), ev.Graphics.VisibleClipBounds)
    '  Указываем, что это последняя печатаемая страница
    ev.HasMorePages = False
End Sub
```

Это процедура обрабатывает событие печати, генерируемое методом PrintDocument1.Print. Объявлена Sub-процедура в коде формы, но вы также можете объявить ее как процедуру общего назначения и поместить в стандартный модуль. Обратите внимание на переменную ev в списке аргументов процедуры PrintGraphic. Это важный носитель информации о текущей печатаемой странице. Она имеет тип PrintPageEventArgs из пространства имен System.Drawing.Printing. Чтобы собственно напечатать графику, процедура использует метод Graphics.DrawImage, ассоциированный с текущей печатаемой страницей. Это метод загружает графический файл, имя которого указано в свойстве Text объекта TextBox1. Вы можете изменить это значение во время выполнения программы и напечатать любой другой графический файл.) Наконец я установил свойство ev.HasMorePages на значение False, чтобы Visual Basic понял, что задание на печать не содержит много страниц.

13. Чтобы сохранить изменения, щелкните на кнопке Save All (Сохранить все) на панели инструментов. Теперь запустите эту программу. Найдите в вашей системе графические файлы, которые вы хотите распечатать. (Просто запомните их пути и введите их в текстовое поле.)

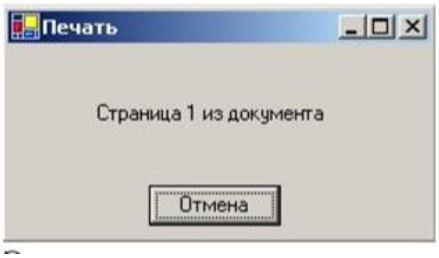
Запуск программы Print Graphics 1.

Щелкните на кнопке Start (Начать) панели инструментов. Ваша программа запустится на выполнение в среде разработки. Вы увидите такую форму.



2. Включите ваш принтер и убедитесь, что в нем есть бумага.

3. Если вы установили файлы примеров в папку по умолчанию c:\vbnet03sbs, щелкните на кнопке Печать графики и распечатайте значок Sun.ico. Если вы не использовали расположение файлов примеров по умолчанию, или если вы хотите распечатать другой графический файл, измените содержимое текстового поля, а затем щелкните на кнопке Print Graphic. Метод DrawImage увеличит изображение до максимального размера, который может быть напечатан на вашем принтере на одной странице, а затем отправит его на принтер. (Эта функция увеличения позволяет лучше рассмотреть изображение.) Если вы хотите изменить расположение или размер распечатки, поищите в справочной системе Visual Basic термин "Graphics.DrawImage", а затем изучите различные доступные аргументы и их значения и измените код программы. Если вы внимательно посмотрите, то увидите, что когда Visual Basic отправляет ваше задание печати на принтер, появляется следующее диалоговое окно.



Это окно состояния также является результатом работы класса PrintDocument и предоставляет пользователям профессионально выглядящий интерфейс печати, включая номер для каждой печатаемой страницы.

4. Введите другие пути, а затем щелкните на кнопке Печать графики, чтобы получить новые распечатки.

5. Когда закончите экспериментировать с этой программой, щелкните на кнопке Закрыть формы. Программа остановится.

Печать текста из объекта Text Box

Вы познакомились с элементом управления PrintDocument и печатью графики. Теперь попробуйте использовать аналогичную методику для печати содержимого текстового поля формы Visual Basic. В следующем упражнении вы создадите проект, который с помощью класса PrintDocument печатает текст, но на этот раз вы определите класс с помощью кода программы, и не будете добавлять элемент управления PrintDocument в вашу форму. В дополнение к этому вы будете использовать для отправки всего содержимого объекта текстового поля на принтер по умолчанию метод Graphics.DrawString. Примечание. Следующая программа разработана для печати только

одной страницы текста или менее. Чтобы печатать несколько страниц, вы должны добавить дополнительный код программы, который изучается далее.

Использование метода Graphics.DrawString для печати текста

1. Создайте новый проект с именем My Print Text. Появится пустая форма.

2. Используйте элемент управления Label и нарисуйте в верхней части формы метку. Эта метка также будет отображать строку с инструкциями для пользователя.

3. Используйте элемент управления TextBox и нарисуйте под меткой текстовое поле. Это поле будет содержать текст, который вы хотите распечатать.

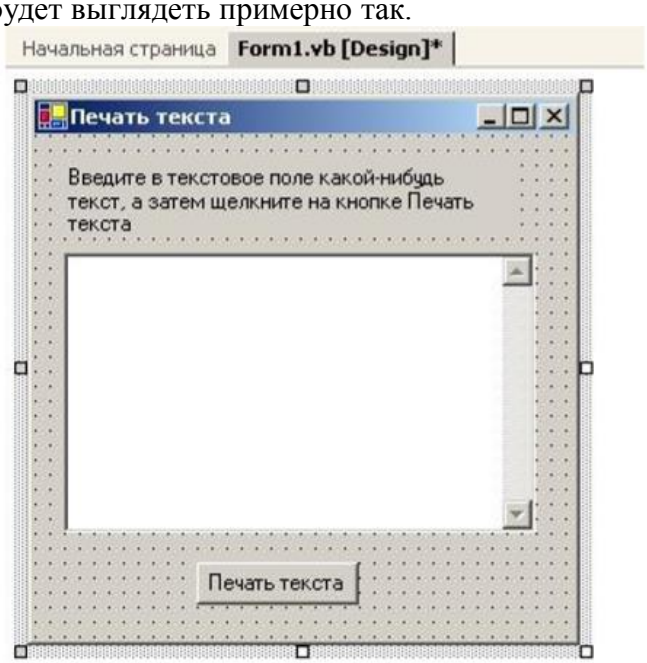
4. Установите свойство Multiline этого текстового поля на значение True, а затем увеличьте текстовое поле так, чтобы оно стало достаточно большим для ввода нескольких строк текста.

5. Используйте элемент управления Button и нарисуйте под объектом текстового поля объект кнопки. Этот объект кнопки будет печатать текстовый файл.

6. Установите для объектов формы следующие свойства:

Объект	Свойство	Установка
Label1	Text	Введите в объект текстового поля какой-нибудь текст, а затем щелкните на кнопке Печать текста.
TextBox1	ScrollBars	Vertical
	Text	пустой (empty)
Button1	Text	Печать текста
Form1	Text	Печать текста

7. Ваша форма будет выглядеть примерно так.



8. Теперь добавьте код программы, необходимый для печати содержимого текстового поля.

9. Сделайте двойной щелчок мышью на кнопке Печать текста. В Редакторе кода появится процедура события Button1_Click.

10. Прокрутите код формы в самый верх, а затем введите следующее объявление Imports:

```
Imports System.Drawing.Printing
```

Оно определяет пространство имен System.Drawing.Printing, которое требуется для определения класса PrintDocument и необходимых объектов этого класса.

11. Теперь снова прокрутите код вниз до процедуры события Button1_Click, а затем введите следующий код программы:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    'Печать с использованием обработчика ошибок для перехвата проблем
    Try
        'Объявляем переменную PrintDoc типа PrintDocument
        Dim PrintDoc As New PrintDocument
        AddHandler PrintDoc.PrintPage, AddressOf Me.PrintText
        PrintDoc.Print()        'печать текста
    Catch ex As Exception   'перехват исключения печати
        MessageBox.Show("Извините, возникли проблемы с печатью", ex.ToString())
    End Try

End Sub
```

Вместо того чтобы добавлять элемент управления PrintDocument в вашу форму, на этот раз вы просто создали PrintDocument программно. Для этого вы использовали ключевое слово Dim и тип PrintDocument, который был определен в вашей программе при определении пространства имен System.Drawing.Printing. После этого переменная PrintDoc представляет объект PrintDocument и используется для объявления обработчика ошибок и печати текстового документа. Обратите внимание, что для ясности я переименовал процедуру, которая будет обрабатывать событие печати в PrintText (вместо PrintGraphic).

12. Прокрутите процедуру события Button1_Click в Редакторе кода вверх до области глобальных объявлений. Введите следующее объявление Sub-процедуры:

```
Public Class Form1
    'Sub для печати текста
    Private Sub PrintText(ByVal sender As Object, _
     ByVal ev As PrintPageEventArgs)
        'Используем DrawString для создания текста в объекте Graphics
        ev.Graphics.DrawString(TextBox1.Text, New Font("Arial", _
        11, FontStyle.Regular), Brushes.Black, 120, 120)
        'Указываем, что это последняя печатаемая страница
        ev.HasMorePages = False
    End Sub
```

Это процедура, которая обрабатывает событие печати, генерируемое методом PrintDoc.Print. Изменения по сравнению с процедурой PrintGraphic из предыдущего упражнения также выделены жирным шрифтом. При печати текста вы используете новый метод. Вместо использования Graphics.DrawImage, который рисует графическое изображение, вы должны использовать Graphics.DrawString, который печатает текстовую строку. Я указал текст для печати в свойстве Text объекта текстового поля, базовое форматирование (Arial, 11 пунктов, обычный стиль, черный цвет) и координаты (x, y) начала печати на странице, равные (120, 120). Это придает печатной странице внешний вид, похожий на текстовое поле на экране. Как и в прошлый раз, я установил свойство ev.HasMorePages на значение False, чтобы указать, что задание на печать не содержит нескольких страниц.

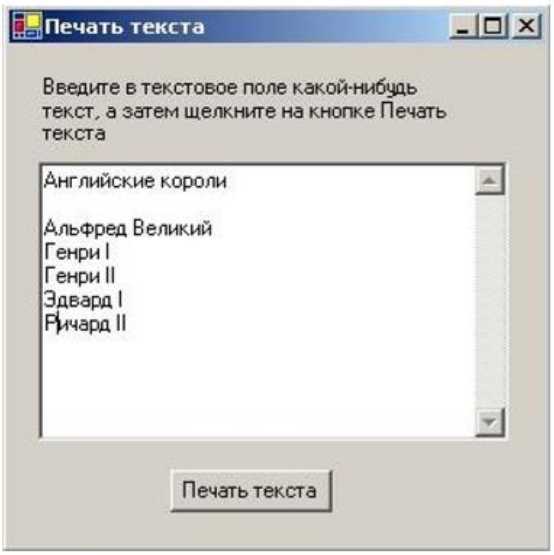
13. Чтобы сохранить изменения, щелкните на кнопке Save All (Сохранить все) на панели инструментов. Запустите программу, чтобы увидеть, как печатается объект текстового поля.

Запуск программы Print Text

1. Щелкните на кнопке Start (Начать) панели инструментов. Ваша программа запустится в среде разработки.

2. Проверьте, что принтер включен.

3. Введите в текстовое поле какой-нибудь текст. Если вы введете несколько строк, убедитесь, что в конце каждой строки вы ввели возврат каретки. Перенос строк в этой демонстрационной программе не поддерживается - слишком длинные строки будут вылезать за правую границу печати. Ваша форма должна выглядеть примерно так.



4. Щелкните на кнопке Печать текста. Программа отобразит диалоговое окно печати и распечатает содержимое вашего текстового поля.

5. Измените текстовое поле и снова попробуйте его распечатать.

6. Когда вы закончите, щелкните на кнопке Закрыть формы. Программа остановится. Теперь вы знаете, как печатать текст и графику.

Печать многостраничных текстовых файлов Методики печати, которые вы только что изучили, используются для простых текстовых документов. Они обладают некоторыми ограничениями. Во-первых, метод, который я использовал, не позволяет печатать длинные строки - другими словами, текст, который выходит за рамки полей страницы. В отличие от объекта текстового поля, объект PrintDocument автоматически не переносит строки, когда они выходят за границы печати. Если у вас есть файлы, не содержащие в конце строк символ возврата каретки, вы должны написать код, который обрабатывает эти длинные строки. Второе ограничение состоит в том, что программа Print Text не может печатать более одной страницы текста. На самом деле, она вообще не понимает, что такое страница текста - процедура печати просто посылает текст на принтер по умолчанию. Если текстовый блок слишком длинный для одной страницы, то оставшийся текст просто не будет напечатан. Чтобы справиться с многостраничной печатью, вы должны создать виртуальную страницу текста, которая называется PrintPage, а затем добавлять на нее текст до тех пор, пока она не заполнится. Когда эта страница заполнится, она будет отправлена на принтер. Этот процесс продолжается до тех пор, пока весь печатаемый текст не будет отправлен на принтер. После этого задание печати заканчивается. Если обработка этих ограничений кажется сложной, не отчаивайтесь - существует несколько удобных механизмов, которые помогают создавать на Visual Basic виртуальные текстовые страницы и печатать текстовые файлы с длинными строками и несколькими страницами текста. Первый механизм - это событие PrintPage, которое возникает при печати страницы. PrintPage получает аргумент типа PrintPageEventArgs, который предоставляет вам размеры и характеристики текущей

страницы принтера. Еще одним механизмом является метод Graphics.MeasureString. Метод MeasureString может быть использован для определения того, сколько символов и строк может поместиться в прямоугольной области страницы. Используя эти, а также некоторые другие механизмы, можно довольно просто создать процедуры, которые выполняют многостраничную печать. Чтобы создать программу с именем Print File, которая открывает текстовые файлы любой длины и печатает их, выполните следующие действия. Программа Print File также демонстрирует, как использовать элементы управления RichTextBox, PrintDialog и OpenFileDialog. Элемент управления RichTextBox является более мощной версией элемента управления TextBox,

который вы использовали только что для отображения текста. Элемент управления PrintDialog отображает стандартное диалоговое окно Print (Печать), так что вы можете указать различные параметры принтера. Элемент управления OpenFileDialog позволяет выбирать текст для печати.

СОДЕРЖАНИЕ

Литература

1. Кудинов, Ю.И. Основы алгоритмизации и программирования. Ч. 1 : учеб. пособие /— Липецк: ЛГТУ, 2013. - http://rucont.ru/efd/303216

2. Макаров А.В. Common Intermediate Language и системное программирование в Microsoft.NET [Электронный ресурс]— Электрон. текстовые данные.- М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.-164 с.— Режим доступа: http://www.iprbookshop.ru/56316.html.

3. Биллиг В.А. Основы программирования на C#. – М.: Бином. Лабора- тория знаний, 2006. – 483 с.

2. Нортроп Т., Уилдермьюс Ш., Райан Б. Основы разработки приложений на платформе Microsoft .Net Framework / Пер. с англ. – М.: Издательскоторговый дом «Русская Редакция», 2007. – 864 с.

3. Волченков, Н.Г. Программирование на языке VB6: учеб, пособие. Часть 1 / Н.Г. Волченков. - М.: Инфра М, 2000. - 99 с.

4. Алексеев, Д.В. Компьютерное моделирование физических задач в Microsoft Visual Basic — Москва : СОЛОН-Пресс, 2009. — 528 с.

5. Зеньковский, В.А. Программирование на Visual Basic 6.5 и Visual Basic.Net. — Москва : СОЛОН-Пресс, 2006. — 248 с.

Учебное издание

Чемисов Николай Николаевич

Высокоуровневые методы информатики и программирования

Учебно-методическое пособие

Компьютерный набор произвел Чемисов Н.Н.

Редактор Лебедева Е.М.

Подписано к печати Формат 60x84. 1/16. Бумага печатная

Усл.п.л. 4,0. Тираж 50 экз. Изд.№

Издательство Брянского государственного аграрного университета

243365, Брянская обл., Выгоничский район, п. Кокино, БГАУ