МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ГРАЖДАНСКОЙ АВИАЦИИ

В.Н. Агеев

ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Учебно-методическое пособие
по выполнению лабораторных работ

«Программирование»

для студентов I курса
направления 25.03.01
очной формы обучения

Москва
2017

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра прикладной математики
В.Н. Агеев

ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Учебно-методическое пособие
по выполнению лабораторных работ

«Программирование»

для студентов I курса
направления 25.03.01
очной формы обучения

Москва - 2017

ББК 6Ф6.5
А23
Рецензент д-р техн. наук А.А. Егорова

Агеев В.Н.

А23 Информатика и информационные технологии: учебно-методическое пособие по выполнению лабораторных работ «Программирование». – М.: МГТУ ГА, 2017. – 34 с.

Данное учебно-методическое пособие издается в соответствии с рабочей программой учебной дисциплины «Информатика и информационные технологии» по Учебному плану для студентов I курса направления 25.03.01 всех очной формы обучения.

Рассмотрено и одобрено на заседаниях кафедры 15.11.2016 г. и методического совета 20.12.2016 г.

Подписано в печать 03.02.2017 г.

Печать офсетная	Формат 60x84/16	1,15 уч.-изд. л.
1,96 усл.печ.л.	Заказ № 1725/138	Тираж 80 экз.

Московский государственный технический университет ГА
125993 Москва, Кронштадтский бульвар, д.20
ООО «ИПП «ИНСОФТ»
107140, г.Москва, 3-й Красносельский переулок, д.20, стр.1

Введение

Целью выполнения лабораторных работ является знакомство студентов с методами программирования в среде Visual Basic.

Правила подготовки и проведения лабораторных работ:

1. До выполнения работы студент должен изучить связанные с ней вопросы и знать порядок ее выполнения.

2. Перед началом работы преподавателю предъявляется оформленная теоретическая часть работы, включая необходимый иллюстративный материал (схемы, таблицы), а также, в некоторых случаях, тексты программ.

3. После выполнения экспериментальной части работы на ПК студент должен предъявить результаты расчетов преподавателю.

4. План построения отчета по лабораторной работе:

– название работы;
– цель работы;
– теоретическая часть;
– расчетная часть.

Отчет выполняется к началу очередного занятия. Законченные отчеты по лабораторным работам должны быть защищены студентом и подписаны преподавателем.

Лабораторная работа №1
Интегрированная среда разработки Visual Basic

Интегрированная среда разработки языка Visual Basic предоставляет пользователю удобный графический интерфейс в процессе разработки приложения. После запуска Visual Basic появляется диалоговое окно выбора режима (рис. 1). При создании нового проекта следует выбрать режим Standaed.EXE.

Рис.1. Окно выбора режима работы Visual Basic

Появится окно интегрированной среды разработки языка Visual Basic (рис. 2).

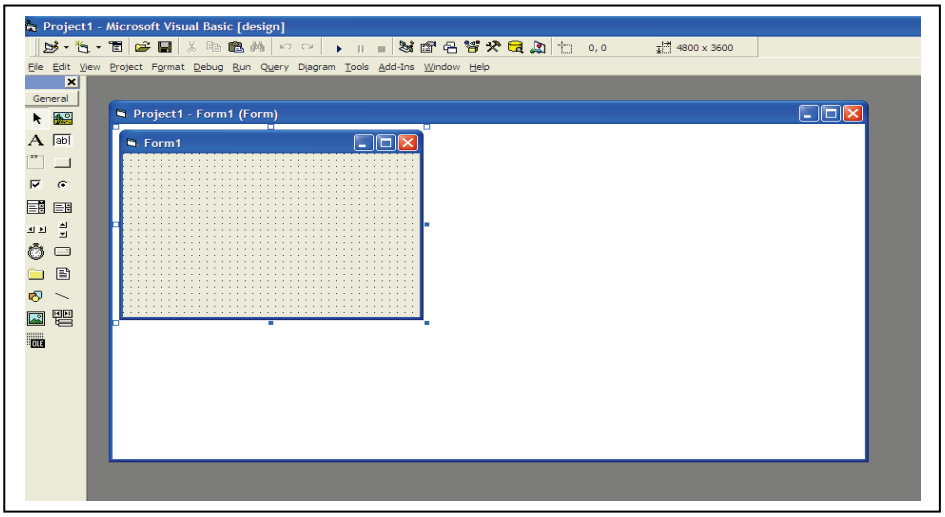


Рис. 2. Интегрированная среда разработки Visual Basic

Интегрированная среда разработки включает в себя:

Строку заголовка, которая состоит из имени проекта *Project1,* после которого через тире указана программная среда *Microsoft Visual Basic.* Далее, словом *[design]* указан текущий режим работы - проектирование. В режиме выполнения проекта текст в квадратных скобках заменяется на *[run]*. Кнопки управления окном расположены в правом углу строки.

Окно конструктора форм является основным рабочим окном и располагается в центре окна интегрированной среды разработки языка Visual Basic. Именно в этом окне происходит визуальное программирование графического интерфейса.

Таблица 1. Элементы управления, их основные свойства и методы

Элемент	Свойство	Событие
Command (Кнопка)	**Default** (При значении True кнопка активна по умолчанию и реагирует на нажатие клавиши Enter)	**Click** (Нажатие кнопки)
TextBox (Текстовое поле)	**Text** Принимает значение содержащегося в текстовом поле текста	**Change** (Изменение содержимого поля)
CheckBox (Флажок)	**Value** Принимает значения 0 – не отмечен, 1 – отмечен, 2 – недоступен	**Click** (Нажатие кнопки)
OptionButton (Переключатель)	**Value** Принимает значения **False** – не отмечен, **True** – отмечен	**Click** (Нажатие кнопки)
ListBox (Список)	**Text** Принимает значение выбранного элемента из списка **ListIndex** Принимает значение индекса выбранного элемента из списка **MultiSelect** 0 – множественный выбор невозможен, 1 – несколько элементов выбираются щелчком мыши	**Click** (Нажатие при выборе из списка)
ComboBox (Поле со списком)	**Text** Значение выбранного элемента списка **ListIndex** Значение индекса выбранного элемента	**Click** (Нажатие кнопки) **Change** (Изменение записи)
Timer (Таймер)	**Interval** Значение интервала времени в миллисекундах	**Timer** (Истекло заданное время)
DriveListBox (Список устройств)	**Drive** Значение выбранного диска	**Change** (Выбор диска из списка)
DirectoryListBox (Список папок)	**Path** Значение пути к выбранной папке	**Change** (Выбор из списка)
FileListBox (Список файлов)	**FileName** Имя выбранного файла **Pattern** Задает шаблон имени и расширения	**Click** (Выбор из списка)
PictureBox (Графическое окно)	**Picture** Название графического объекта	

В левой части окна Visual Basic располагается *Панель инструментов,* содержащая пиктограммы *управляющих элементов.* Стандартный набор управляющих элементов включает в себя 21 класс объектов: *Командная кнопка* (CommandButton), *Текстовое поле* (TextBox), *Метка* (Label) и т.д. (рис. 3).

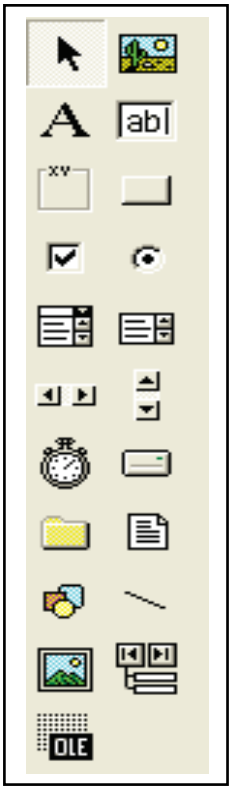


Рис.3. Панель инструментов

Выбрав щелчком мыши нужный элемент, можно поместить его на форму проектируемого приложения. Фактически, мы размещаем на форме экземпляры определенных классов объектов. Например, выбрав класс *Командная кнопка* (CommandButton), мы можем разместить на форме неограниченное количество экземпляров этого класса, т.е. кнопок Command1, Comma«d2, Command3 и т.д.

Выделив помещенный на форму элемент и нажав правую кнопку мыши можно перейти к окну свойств данного элемента. В качестве примера на рис.5.4 приведено окно свойств для элемента «кнопка» (Command1).

Окно содержит список объектов и список свойств, относящихся к выбранному объекту (форме или управляющему элементу на форме). На рисунке выбран объект Forml из класса Form.

Список свойств разделен на две колонки. В левой находятся имена свойств, а в правой – их значения. Установленные по умолчанию значения могут быть изменены. Свойством объекта является качественная или количественная характеристика этого объекта (размеры, цвет, шрифт и др.).

С каждой формой связан программный модуль, содержащий событийные и общие процедуры. Для перехода к режиму ввода и редактирования текста программы из основного меню необходимо выбрать пункт «View» и в развернувшемся меню – опцию «Code». Появится *окно программного кода* (рис.4).

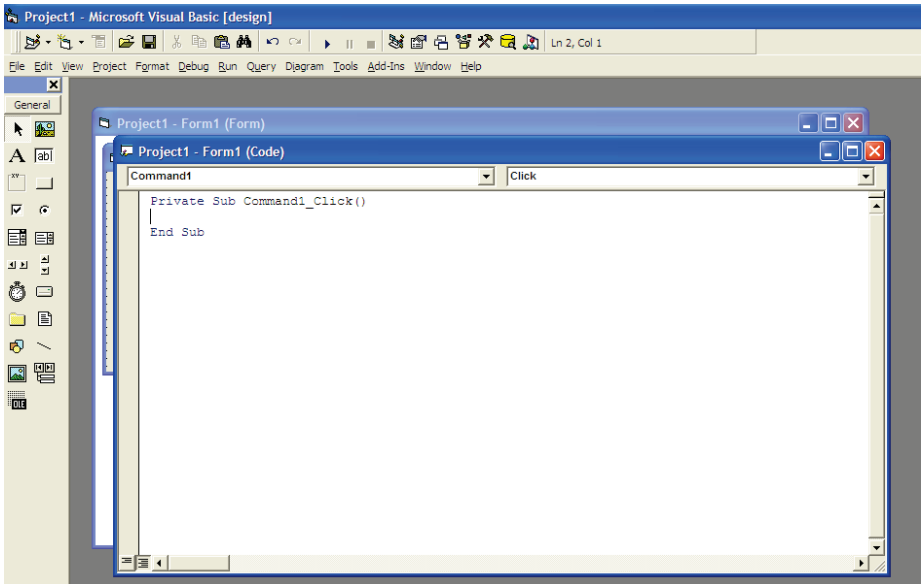
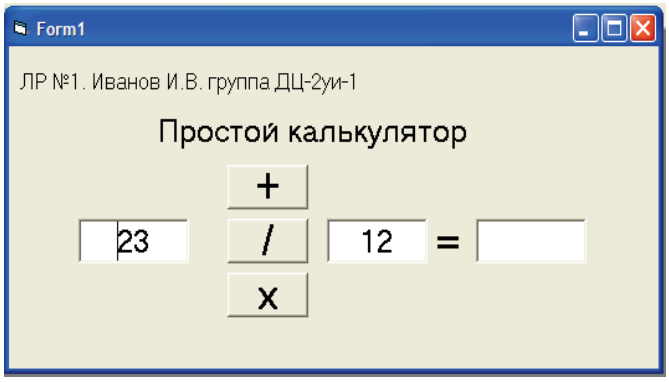


Рис.4. Окно программного кода

Сразу под строкой заголовка окна программного кода размещаются два раскрывающихся списка. Левый список содержит перечень объектов проекта (объектов размещенных на форме), а правый – перечень событий, доступных для данного объекта. При выборе из левого списка объект, а из правого – событие в рабочем окне появится заготовка процедуры, имя которой состоит из имени объекта и названия события. Например, для кнопки «Command1» и события «Click» процедура получит имя *Command1_Click*.

Задание.

Разработать проект «Простой калькулятор». Создать форму с тремя текстовыми окнами и тремя кнопками с надписями «+», «/» и «*». После ввода в первые два окна двух чисел и нажатии одной из кнопок в третьем окне отображается результат: сумма, частное или произведение этих чисел.



Лабораторная работа № 2
Программирование линейных вычислительных процессов

В объектно-ориентированных языках программирования и, в частности, в языке Visual Basic, *переменные* играют такую же важную роль, как и в алгоритмических языках программирования. Переменные предназначены для хранения и обработки данных в программах.

Переменные задаются *именами,* которые определяют области памяти, в которых хранятся их *значения.* Значениями переменных могут быть *данные* различных типов (целые или вещественные числа, последовательности символов, логические значения и т.д.).

Тип переменных определяется типом данных, которые могут быть значениями переменных. Значениями переменных числовых типов (Byte, Integer, Long, Single, Double) являются числа, логических (Boolean) — True или False, строковых (String) — последовательности символов и т.д. Обозначения типов переменных являются ключевыми словами языка и поэтому выделяются.

Имя каждой переменной уникально и не может меняться в процессе выполнения программы. Имя переменной может состоять из различных символов (латинские и русские буквы, цифры и т.д.), но должно обязательно начинаться с буквы и не должно включать знак точка «.». Количество символов в имени не может быть более 255.

Наборы однотипных переменных могут быть объединены в массивы, объединенные одним именем. Массивы бывают *одномерные,* которые можно представить в форме одномерной таблицы, и *двумерные,* которые можно представить в форме двумерной таблицы. Массивы могут быть различных типов: *числовые, строковые* и т.д.

Массив состоит из пронумерованной последовательности элементов. Номера в этой последовательности называются *индексом.* Каждый из этих элементов является переменной, т.е. обладает именем и значением, и поэтому массив можно назвать *переменной с индексом.*

Для объявления типа переменной можно воспользоваться также оператором определения переменной. Синтаксис (правило записи) этого оператора следующий:

Dim *ИмяПеременной* [.**As** *ТипПеременной*]

Здесь **Dim** и **As** ключевые слова языка Visual Basic и поэтому они выделяются жирным шрифтом. Назначение этого оператора — объявить переменную, т.е. задать ее имя и тип, однако объявление типа может отсутствовать. С помощью одного оператора можно объявить сразу несколько переменных, например:

Dim *ИмяПеременной* **As Integer,** *ИмяПеременной* **As String**

Объявление массива производится аналогично объявлению переменных, необходимо только дополнительно указать диапазон изменения индекса. Например, объявление строкового массива, содержащего 33 элемента, производится следующим образом:

Dim *ИмяМассива* **(** 1 **To** 33**) As String**

Переменные, значения которых не меняются в процессе выполнения программы, называются *константами.* Синтаксис объявления констант следующий:

Const *ИмяКонстанты* [**As** Тип]= *ЗначениеКонстанты*

Переменная может получить или изменить значение с помощью *оператора присваивания.* Синтаксис этого оператора следующий:

ИмяПеременной = Выражение

При выполнении оператора присваивания переменная, имя которой указано слева от знака равенства, получает значение, равное значению выражения (арифметического, строкового или логического), которое находится справа от знака равенства.

Кроме четырех арифметических действий при записи арифметического выражения можно использовать стандартные функции, представленные в следующей таблице.

Словесное описание	**Запись в Visual Basic**
Возведение числа X в степень Y	X^Y
Извлечение квадратного корня из X	Sqr(X)
Остаток от целочисленного деления X на Y	X mod Y
Целочисленное частное от деления X на Y	X div Y
Абсолютное значение (модуль) числа X	Abs (X)
Синус угла X, заданного в радианах	Sin (X)
Косинус угла X, заданного в радианах	Cos (X)
Тангенс X	Tg (X)
Арктангенс X	Atn (X)
Ближайшее целое числа X	Int (X)
Целая часть числа X	Fix (X)
Экспонента числа X	Exp (X)
Натуральный логарифм X	Log (X)

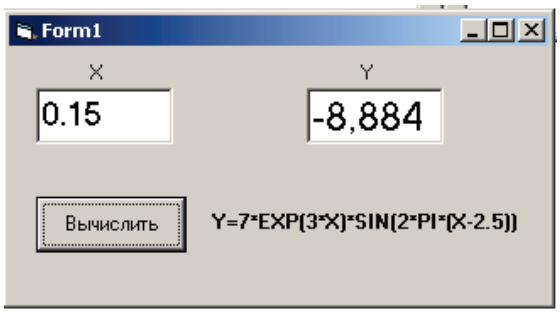
Пример: вычислить значение функции $y = 7 \cdot e^{3x} \cdot \sin(2 \cdot \pi \cdot (x - 2.5))$ для x, введенного в поле **Text1**.

Решение.

Программный код:

```
Private Sub Command1_Click()
PI = 3.1415
X = Val(Text1.Text)
Y = 7 * Exp(3 * X) * Sin(2 * PI * (X - 2.5))
Text2.Text = Y
End Sub
```

Экранная форма:



Задание:

1) создать экранную форму, разместив на ней элементы управления: кнопку и два текстовых поля – одно для ввода числа, другое – для вывода результата вычислений;

2) поместить на форму поясняющие надписи около этих элементов, а также фамилию, номер группы, текущую дату и номер варианта;

4) создать процедуру, запускаемую при нажатии кнопки на форме и выполняющую вычисления по заданной формуле.

Варианты задания

№	функция	аргумент
1	$(x^3 - 8) \cdot \cos(2 \cdot \pi \cdot x - 3)$	0, 25
2	$3 \cdot (x^2 - 4) \cdot (x^2 - 1)$	-2,53
3	$x \cdot (x^3 - 2 \cdot x^2 - x + 2)$	0,5
4	$8 \cdot e^{2x-1} \cdot \sin(2 \cdot \pi \cdot (x + 2))$	-1,1
5	$2 \cdot e^{2(x+1)}) \cdot \cos(3 \cdot \pi \cdot (x + 1))$	0, 25
6	$10 \cdot e^{2x} \cdot \sin(2 \cdot \pi \cdot (x + 0.5))$	-0,4
7	$x \cdot (x^2 + x - 2)$	1,2
8	$5 \cdot e^{3x} \cdot \cos(3 \cdot \pi \cdot x + 5)$	0,25
9	$4 \cdot \cos(x^2) \cdot \sin(2 \cdot \pi \cdot x)$	0,28
10	$3 \cdot x^3 \cdot \cos(4 \cdot \pi \cdot x - 2)$	-0,9
11	$4 \cdot (2 \cdot x^2 - 5x + 3) \cdot (2 \cdot x^2 - 9 \cdot x + 10)$	0,3
12	$2 \cdot e^{-3x} \cdot \cos(2 \cdot \pi \cdot x + 1)$	0,37
13	$4 \cdot x^3 \cdot \sin(2 \cdot \pi \cdot x - 3)$	-1,25
14	$x^4 - 4 \cdot x^3 + x^2 - 6x$	-2,15

№	функция	аргумент
15	$3 \cdot x^2 \cdot \cos(3 \cdot \pi \cdot x - 1)$	0,6
16	$5 \cdot x^3 \cdot \cos(4 \cdot \pi \cdot x - 3)$	1,2
17	$5 \cdot x^3 \cdot \sin(\pi \cdot x - 3)$	-0,55
18	$4 \cdot e^{2x+1} \cdot \sin(3 \cdot \pi \cdot (x + 1.5))$	-1,34
19	$3 \cdot x^2 \cdot \sin(3 \cdot (\pi \cdot x - 1))$	-0,95
20	$(x^2 - 3 \cdot x + 2) \cdot (x^2 - 7 \cdot x + 12)$	0,45
21	$7 \cdot e^{3x} \cdot \sin(2 \cdot \pi \cdot (x - 2.5))$	-1,25
22	$2 \cdot x^2 \cdot \cos(3 \cdot \pi \cdot x + 2)$	-0,25
23	$x \cdot (x^2 - 0{,}25) \cdot (x^2 - 3 \cdot x + 2)$	0,28
24	$3 \cdot e^{2x-1} \cdot \sin(\pi \cdot x + 2)$	-1,3
25	$4 \cdot \cos(x^2) \cdot \sin(5 \cdot \pi \cdot x)$	1,8

Лабораторная работа № 3
Программирование разветвляющихся и циклических процессов

Изменить последовательность выполнения операторов можно с помощью операторов ветвления.

Одним из них является условный оператор **If** … **Then**. Он имеет несколько форматов записи.

1. Однострочный формат

If *условие* **Then** *Оператор1* [**Else** *Оператор2*]

(в квадратные скобки взята необязательная часть). Если *условие* истинно, *Оператор1* выполняется, в противном случае он пропускается и выполняется *Оператор2*, если он указан.

2. Блочный формат.

If *условие* **Then**
Операторы1
[**Else**
Операторы2]
End If

Эта форма применяется, когда нужно при выполнении заданного условия выполнить не один, а несколько операторов.

Еще одним оператором ветвления является **Select Case**.

Select Case *Выражение*
Case *Список_значений1*
Операторы1
Case *Список_значений2*
Операторы2
…….
[**Case Else** *Список_значений3*
Операторы3]
End Case

Здесь в качестве параметра *Выражение* может быть как имя переменной, так и арифметическое выражение, а параметры *Список_значений1*, *Список_значений2* и т.д. могут быть как одиночными числовыми значениями, так и списками – несколькими числами, разделенными запятыми.

Для многократного выполнения одного или нескольких операторов применяются операторы цикла. Имеется несколько типов оператора цикла.

1. Оператор цикла типа пересчета:

For *Счетчик* = *Начальное_значение* **To** *Конечное_значение* [**Step** *Шаг*]
Операторы
Next [*Счетчик*]

Здесь *Счетчик* – параметр цикла, переменная целого или вещественного типа, *Начальное_значение* и *Конечное_значение* – числа, задающие

границы интервала изменения параметра цикла, *Шаг* – шаг изменения параметра цикла, если он не указан, то по умолчанию он принимается равным 1. *Операторы* – один или несколько операторов, которые повторяются до тех пор, пока *Счетчик* не достигнет конечного значения.

2. Оператор повторений с предусловием **Do While … Loop**:

Do While *Условие*
Операторы
Loop

Указанные *Операторы* повторяются до тех пор, пока *Условие* остается истинным.

3. Оператор повторений с постусловием **Do … Loop While**:

Do
Операторы
Loop While *Условие*

Операторы повторяются до тех пор, пока *Условие* остается истинным. Отличие от предыдущего случая в том, что если *Условие* к моменту начала цикла ложно (имеет значение **False**), указанные операторы будут хотя бы один раз выполнены.

4. Оператор повторений с предусловием **Do Until … Loop**:

Do While *Условие*
Операторы
Loop

Операторы повторяются до тех пор, пока *Условие* не станет истинным.

5. Оператор повторений с постусловием **Do … Loop Until**:

Do
Операторы
Loop Until *Условие*

Здесь *Операторы* повторяются до тех пор, пока *Условие* не станет истинным. Отличие от предыдущего случая в том, что если *Условие* к моменту начала цикла уже является истинным (имеет значение **True**), указанные операторы будут хотя бы один раз выполнены.

Задание:

1) создать экранную форму, разместив на ней элементы управления: кнопку и два текстовых поля – одно для ввода чисел, другое – для вывода результата вычислений;

2) поместить на форму поясняющие надписи около этих элементов, а также фамилию, номер группы, текущую дату и номер варианта;

4) создать процедуру, запускаемую при нажатии кнопки на форме и выполняющую необходимые вычисления.

Варианты задания

1. В заданном массиве подсчитать количество четных и нечетных чисел:

234	122	213	245	279	220	334	303	217	160

В зависимости от полученного результата вывести сообщение «четных больше», «нечетных больше» или «одинаковое количество».

2. Для заданного массива значений случайной величины вычислить среднее арифметическое и сформировать новый массив из квадратов разностей элементов заданного массива и средним арифметическим

28	22	21	15	27	28	33	19	21	22

3. В заданном массиве отыскать наибольший элемент, вывести его номер и значение

–23	12	21	15	–29	22	33	13	–2	16

4. Вычислить длины векторов A и B:

3	0	–1	5	9

2	4	4	–2	2

В зависимости от полученного результата вывести сообщение «длина A больше», «длина B больше» или «длины одинаковы».

5. Упорядочить заданный массив по возрастанию

18	22	12	13	27	28	33	19	20	12

6. Даны три вектора:

A	1	3	5	B	1	2	4	C	2	3	3

Определить, какой из векторов, A или B, ближе по направлению к вектору C.

7. В заданном массиве подсчитать количество чисел, делящихся без остатка на 3:

28	22	21	15	27	28	33	19	21	22

8. Вводится целое число N. Вычислить и вывести значение $Z=N!=1\cdot2\cdot\ldots\cdot N$. Учесть, что по определению 0!=1. Предварительно сделать проверку знака N и если N<0 вычислений не проводить, а вывести сообщение “N<0”. Провести расчеты для N=4.

9. Упорядочить заданный массив по убыванию

28	12	24	13	27	29	31	21	20	14

10. Вводятся три числа A, B и C. Определить действительные корни X1 и X2 квадратного уравнения $Ax^2+Bx+C=0$. Если действительных корней нет, вывести соответствующее сообщение. Для расчетов взять A=1, B= –1, C= –6.

11. В заданном массиве отыскать наименьший элемент, вывести его номер и значение.

21	11	21	15	22	15	28	13	23	16

12. В заданном массиве поменять местами наибольший и наименьший элементы

22	28	12	13	27	18	33	19	21	15

13. Найти наибольшее значение функции $y=x^2–A\cdot x$ на отрезке [0, A], изменяя аргумент x от 0 до A с шагом h . Вывести значения x_{max} и y_{max}. Для расчетов взять A=2, h=0.2

14. В заданном массиве подсчитать количество чисел, кратных 5:

234	122	213	245	279	220	334	303	217	160

15. Вводятся 6 чисел – координаты вершин треугольника: x_1, y_1, x_2, y_2, x_3, y_3. Вычислить его площадь. Для расчетов взять числа (0,0,1,1,2,0).

16. Разделить заданный массив на два, один из которых содержит только отрицательные числа исходного массива, а другой – нулевые и положительные:

–23	22	21	45	–27	–20	–34	30	27	–16

17. Для заданного массива вычислить среднее геометрическое Z и сформировать новый массив, вычитая из заданных значений величину Z:

28	22	21	15	27	28	33	19	21	22

18. Нормализовать заданный вектор, то есть сделать его длину равной единице. Для этого надо вычислить длину вектора и разделить на нее все элементы заданного вектора:

1	1	2	3	1

19. Даны три вектора:

A	2	3	2	B	1	2	5	C	2	3	4

Определить, какой из векторов, A или B, дальше по направлению отстоит от вектора C.

20. Вводятся 6 чисел – координаты вершин треугольника: x_1, y_1, x_2, y_2, x_3, y_3. Вычислить его периметр. Для расчетов взять числа (1,0,1,1,2,1).

21. Разложить заданный массиве на два, первый состоит из четных значений исходного массива, второй – из нечетных:

234	122	213	245	279	220	334	303	217	160

22. Найти наименьшее значение функции $y=A^2/16+A\cdot x-x^2$ на отрезке [0, A], изменяя аргумент x от 0 до A с шагом h. Вывести значения x_{min} и y_{min}. Для расчетов взять A=1, h=0.1

23. В заданном массиве подсчитать количество чисел, делящихся без остатка на 7:

28	21	26	14	63	28	33	19	21	22

24. Вводятся 6 чисел – координаты A, B, C вершин треугольника. Вычислить его площадь. Для расчетов взять числа A=(0,0,), B=(1,1) C=(2,0).

25. Нормализовать заданный вектор, то есть сделать его длину равной единице. Для этого надо вычислить длину вектора и разделить на нее все элементы заданного вектора:

2	2	2	0	2

Лабораторная работа № 4
Действия с символьными строками

Во многих случаях требуется выполнить действия с символьными строками. Например, требуется определить количество символов в строке или выделить подстроку из заданной строки.

В таблице приведены примеры использования некоторых функций для работы с символьными строками.

Выражение	Комментарий
N = Len (S)	Длина строки S присваивается числовой переменной N
S1 = Left (S, n)	Строковая переменная S1 получает n левых символов S
S1 = Right (S, n)	Строковая переменная S1 получает n правых символов S
S1 = Mid (S, n, m)	Строковая переменная S1 получает n символов S, начиная с m-го
N =Val (S)	Числовая переменная N получает значение числа из строки S
S = Str (N)	Строковая переменная S получает символьное представление числа N
N = Asc (C)	N получает значение ASCII-кода символа C
C = Chr (N)	C получает значение символа с кодом ASCII
S1 = Ucase (S)	Преобразование всех букв строки S в прописные
S1 = Lcase (S)	Преобразование всех букв строки S в строчные

Следует помнить, что коды заглавных и прописных букв разные. Коды заглавных букв от «А» до «Я» (без буквы «Ё») идут подряд от 192 до 223, а прописных – от 224 до 255 (т.е. для код каждой прописной буквы на 32 больше кода заглавной. Исключение – буква «Ё» (код 168) и «ё» (код 184).

Коды с 0 до 31 относятся не к символам, отображаемым на экране, а служат для обозначения управляющих команд. Так, например код 13 означает «возврат каретки», то есть переход к началу строки, а код 10 – переход на следующую строку.

Порядок выполнения:

1) создать экранную форму, разместив на ней необходимые элементы управления;

2) поместить на форму поясняющие надписи около этих элементов, а также фамилию, номер группы, текущую дату и номер варианта;

3) создать процедуры, реализующие разработанный алгоритм.

Варианты задания.

1. В текстовое поле вводится строка из трех чисел, разделенных пробелами. При нажатии кнопки «Решение» запускается процедура, которая вычисляет сумму этих чисел и выводит результат во второе текстовое поле. Если строка содержит не цифровые символы, то вместо суммы в это поле выводится сообщение «Строка содержит ошибку».

2. В текстовое поле вводится строка из цифр. При нажатии кнопки «Пуск» запускается процедура, которая подсчитывает количество цифр во введенной строке и вычисляет их сумму. Результаты выводятся в соответствующие текстовые поля. Если строка содержит не цифровые символы, подсчитывается только количество введенных символов, а сумма не вычисляется.

3. В текстовое поле вводится строка из нескольких слов. При нажатии кнопки «Обработка» запускается процедура, которая подсчитывает, сколько раз в этой строке появляются буквы «а» и «е» (как строчных, так и прописных). Результаты выводятся в соответствующие текстовые поля.

4. В текстовое поле вводится натуральное число. При нажатии кнопки «Перевод» запускается процедура, которая преобразует это число в двоичное, восьмеричное и 16-ричное. Результаты выводятся в соответствующие текстовые поля. Если строка содержит не цифровые символы, то выводится сообщение об ошибке.

5. В текстовое поле вводится натуральное число. При нажатии кнопки «Пуск» запускается процедура, которая определяет, есть ли в этом числе одинаковые цифры, и если есть, то сколько раз они там появляются. Результаты выводятся в виде таблицы. Если одинаковых цифр в числе нет, то вместо таблицы выводится сообщение об этом.

6. В два текстовых поля вводятся две строки символов. При нажатии кнопки «Сравнение» запускается процедура, которая определяет, есть ли в этих строках одинаковые символы, и если есть, то они выводятся в текстовое окно.

7. В два текстовых поля вводятся по строке символов, причем вторая строка короче первой. При нажатии кнопки «Поиск» запускается процедура, которая определяет, входит ли вторая строка в первую в качестве подстроки. Результат выводится в текстовое окно. В случае, если вторая строка окажется длиннее первой, выводится сообщение об ошибке.

8. В текстовое поле вводится строка, состоящая из букв и цифр. При нажатии кнопки «Обработка» запускается процедура, которая из данной строки удаляет цифры и формирует новую строку, в которой остаются только введенные буквы. Результат выводится в во второе текстовое поле.

9. В текстовое поле вводится строка, состоящая из русских строчных букв. При нажатии кнопки «ОК» запускается процедура, которая из данной строки формирует другую из тех же букв, но упорядоченных по алфавиту (например: «казак» ⇒ «аазкк»). Результат выводится во второе текстовое поле.

10. При нажатии кнопки «Пуск» запускается процедура, которая с помощью датчика случайных чисел формирует массив из 10 чисел в диапазоне от 0 до 100. Эти числа выводятся в текстовое окно. При нажатии кнопки «Обработка» запускается другая процедура, которая в сформированном массиве определяет наибольшее и наименьшее значения и выводит их в текстовые окна.
Указание. Случайное число из интервала (0, 1) возвращает функция rnd. Пример: переменная X получит значение из интервала (0, 100) после выполнения оператора X=100*rnd

11. В текстовое поле вводится строка из трех слов, разделенных пробелами. При нажатии кнопки «ОК» запускается процедура, результатом работы которой является новая строка, отличающаяся от введенной тем, что первое и третье слова меняются местами. Эта строка выводится во второе текстовое поле. Если введенная строка содержит меньше трех слов, новая строка не формируется, вместо нее во второе поле выводится сообщение «В строке должно быть три слова!».

12. В текстовое поле вводится строка из русских букв. При нажатии кнопки «Решение» запускается процедура, которая подсчитывает количество гласных и согласных букв в строке. Результат выводится во второе текстовое поле.

13. В текстовое поле вводится строка из трех чисел, разделенных пробелами. При нажатии кнопки «Решение» запускается процедура, которая определяет наибольшее и наименьшее из этих чисел и выводит результат во второе текстовое поле. Если строка содержит не цифровые символы, то вместо суммы в это поле выводится сообщение «Строка содержит ошибку».

14. В текстовое поле вводится целое число. При нажатии кнопки «Пуск» запускается процедура, которая определяет, каких цифр в числе больше – четных или нечетных. Соответствующее сообщение выводится в текстовое окно.

15. В текстовое поле вводится строка из русских букв прописных и заглавных. При нажатии кнопки «Обработка» запускается процедура, которая формирует из этой строки новую, в которой все прописные буквы заменены заглавными, а заглавные – прописными. Новая строка выводится во второе окно.

16. В текстовое поле вводится натуральное число в восьмеричной системе счисления (Цифры от 0 до 7). При нажатии кнопки «Перевод» запускается процедура, которая преобразует это число в десятичную систему. Результат выводится во второе текстовое поле. Если введенная строка содержит цифры 8 или 9, то выводится сообщение «Число не восьмеричное».

17. В текстовое поле вводится натуральное число. При нажатии кнопки «Пуск» запускается процедура, которая определяет, делится ли это число без остатка на 2, 3, 5 и 7. Для вывода слов «Да» или «Нет» используются четыре текстовых поля, рядом с которыми помещены поясняющие надписи («число делится на 2», «число делится на 3» и т.д.). Если во введенной строке кроме цифр есть другие символы (точка, буквы), то выводится сообщение об ошибке.

18. В два текстовых поля вводятся две строки символов одинаковой длины. При нажатии кнопки «ОК» запускается процедура, которая формирует новую строку, в которой символы, стоящие на нечетных местах берутся из первой введенной строки, а стоящие на четных – из второй (например: «парк», «лего» $\Rightarrow$ «перо»). Эта строка выводится в третье текстовое поле. Если введенные строки имеют разную дину, выводится сообщение об ошибке.

19. В первое текстовое поля вводится строка из русских букв, во второе – одна буква. При нажатии кнопки «Поиск» запускается процедура, которая определяет, встречается ли эта буква в строке и если да, то сколько раз. Результат выводится в третье текстовое поле.

20. В текстовое поле вводится строка, состоящая из русских и латинских букв. При нажатии кнопки «Обработка» запускается процедура, которая из данной строки удаляет латинские буквы и формирует новую строку, в которой остаются только русские буквы. Результат выводится в во второе текстовое поле.

21. При нажатии кнопки «Пуск» запускается процедура, которая с помощью датчика случайных чисел формирует массив из 10 чисел в диапазоне от -100 до 100. Эти числа выводятся в текстовое окно. При нажатии кнопки «Обработка» запускается другая процедура, которая в сформированном массиве определяет количество отрицательных, положительных и нулевых значений и результат выводит в текстовые окна.

Указание. Случайное число из интервала (0, 1) возвращает функция rnd. Пример: переменная S получит значение из интервала (–100,100) после выполнения оператора S=200*rnd – 100.

22. В текстовое поле вводится строка из трех слов, разделенных запятыми. При нажатии кнопки «ОК» запускается процедура, результатом работы которой является новая строка, отличающаяся от введенной тем, что второе и третье слова меняются местами, а разделяющие слова запятые заменяются пробелами. Эта строка выводится во второе текстовое поле. Если введенная строка содержит меньше трех слов, новая строка не формируется, вместо нее во второе поле выводится сообщение «В строке должно быть три слова!».

23. В текстовое поле вводится строка из русских и латинских букв, а также цифр. При нажатии кнопки «Решение» запускается процедура, которая подсчитывает количество цифр, русских и латинских букв в строке. Результат выводится во второе текстовое поле.

24. В текстовое поле вводится натуральное число N. При нажатии кнопки «Пуск» запускается процедура, которая с помощью датчика случайных чисел формирует массив из N целых чисел в диапазоне от 192 до 233 (коды заглавных букв от А до Я). Эти числа выводятся в текстовое окно. При нажатии кнопки «Обработка» запускается другая процедура, которая преобразует последовательность числовых кодов в строку символов и выводит ее в текстовое окно.

Лабораторная работа № 5
Построение графиков

На форме или в графическом поле можно рисовать различные графические примитивы с использованием графических методов. В табл. 5.6 приведены примеры использования этих методов. В качестве объекта *object*, куда выводятся графические примитивы, может служить сама форма (в этом случае имя объекта *Form* можно не указывать) или графическое окно *PictureBox*.

Таблица 1. Графические примитивы

Наименование	Синтаксис и комментарии
Точка	object.Pset (X,Y), C X,Y – координаты точки, C – цвет.
Окружность	object.circle (X, Y) , R, C X, Y – координаты центра в выбранной системе координат, R – радиус, C – цвет.
Дуга окружности	object.circle (X, Y) , R, C, A, B X, Y – координаты центра, R – радиус, C – цвет. A, B – углы дуги в радианах. дуга строится против часовой стрелки от A к B.
Круговой сектор	object.circle (X, Y) , R, C, –A, –B Минус перед углами означает, что из центра к концам дуги строятся отрезки прямых, образуя угловой сектор.
Овал	object.circle (X, Y), R, C,,, K K – коэффициент сжатия овала. При 0 < K< 1 сжатие по горизонтали, при K>1 – по вертикали.
Отрезок линии	object.Line (Xl,Y1) –(X2,Y2), C XI, Yl – координаты точки начала отрезка, X2, Y2 – его конца, C – цвет.
Прямоугольник	object.Line(Xl,Y1) –(X2,Y2), C, B XI, Yl — координаты левой верхней вершины прямоугольника, X2, Y2 — координаты правой нижней вершины, C – цвет.
Прямоугольник закрашенный	object.Line (Xl,Y1) –(X2,Y2), C, BF XI, Yl — координаты левой верхней вершины прямоугольника, X2, Y2 — координаты правой нижней вершины, C – цвет.
Очистка нарисованного	object.Cls
Возвращение цвета точки с указанными координатами	object.Point (X,Y)
Вывод строки символов	object.Print [output] В качестве output может быть строковое или числовое выражение. Вывод осуществляется от последней построенной точки изображения. Для указания точки вывода можно использовать метод Pset(X,Y).
Масштабирование окна вывода	object.Scale (XI, Yl) – (X2,Y2) (XI, Yl) и (X2,Y2) – «мировые» координаты выводимого изображения, левой верхней и правой нижней вершины окна соответственно

При построении изображения в графическом окне или на самой форме важным является выбор масштаба по вертикальной и горизонтальной осям. Масштаб устанавливается с помощью метода **Scale**.

При выводе текста в графическое окно можно задавать тип и размер шрифта, а также цвет выводимых символов и линий. Для этого используются свойства объекта *FontName* , *FontSize* и *ForeColor*. Кроме того, свойство *FontTransparent* позволяет сделать текст «прозрачным», то есть не закрывающим линии графика.

Программно цвет можно задать тремя способами.

1) Используя константы цветов. В этом случае цвет указывается непосредственно, например:

```
Form1.ForeColor = vbRed
```

2) С помощью функции RGB (Red-Green-Blue). Значение каждого из цветов меняется от 0 до 255. Например,

```
Form1.ForeColor = RGB( 255, 0, 0)
```

означает, что цвет символов будет ярко-красный.

Порядок выполнения

Необходимо построить в графическом окне график функции y=f(x) на заданном интервале $a \le x \le b$. В графическом окне должны быть также построены координатные оси с делениями и числовыми значениями около них в соответствии с выбранным масштабом.

Работа выполняется в два этапа.

На первом этапе заданная функция табулируется, т.е. для заданной функции y=f(x)выводятся на экран пары чисел x_i, $y_i=f(x_i)$ (i=1,…,N). По этим значениям оценивается интервал изменений значений функции на заданном интервале.

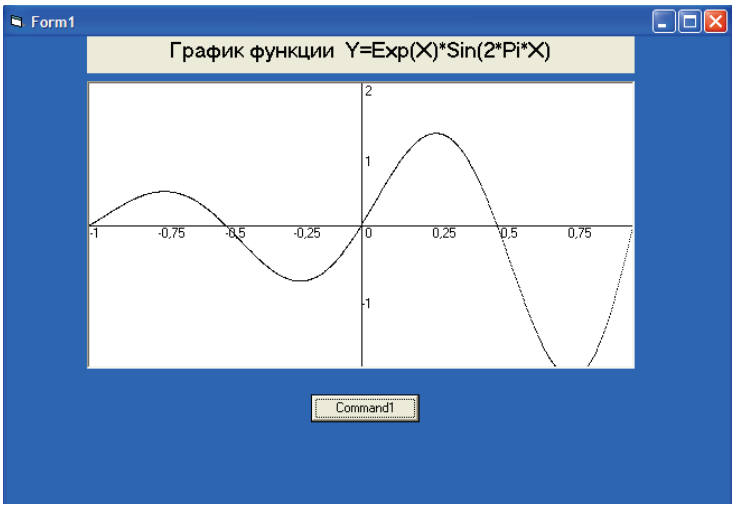
На втором этапе осуществляется построение координатных осей и вывод графика функции в графическое окно.

Пример. Построить график функции $y = e^x \cdot \sin(2\pi x)$ на интервале [–1, 1].

Ниже приведена процедура построения осей координат и графика функции в графическом окне Picture1. Запускается процедура по нажатию кнопки Command1.

```
Private Sub Command1.Click ( )
Const PI = 3.141529
Picture1.Scale  (–1, 2 ) – (1 , –2)   ’ Задание масштаба
Picture1.Line   (–1,  0) – (1 , 0)     ‘ Построение координатных осей
Picture1.Line    (0 , –2) – (0 ,  2)
For  I = –1 To  1  Step  0.25          ‘ Вывод надписей на оси X
        Picture1.PSet ( I , 0)
        Picture1.Print I
```

```
Next  I
For  I = –2 To  2                          ‘  Вывод надписей на оси Y
        Picture1.PSet (0 , I)
        Picture1.Print  I
Next  I
For  X =  –1  To  1  Step  0.002          ‘  Вывод точек графика функции
        Y = exp (X) * sin (2 * PI * X ) )
        Picture1.PSet  ( X , Y )
Next  X
End Sub
```



На рисунке показан вид экранной формы с графическим окном для данного примера.

Варианты задания.

№	функция	интервал
1	$3 \cdot x^2 \cdot \cos(3 \cdot \pi \cdot x - 1)$	[0, 1]
2	$10 \cdot e^{2x} \cdot \sin(2 \cdot \pi \cdot (x + 0.5))$	[-1, 1]
3	$x \cdot (x^2 + x - 2)$	[-2, 2]
4	$5 \cdot e^{3x} \cdot \cos(3 \cdot \pi \cdot x + 5)$	[0, 2]
5	$4 \cdot \cos(x^2) \cdot \sin(2 \cdot \pi \cdot x)$	[-1, 2]
6	$3 \cdot x^3 \cdot \cos(4 \cdot \pi \cdot x - 2)$	[-1, 1]
7	$x \cdot (x^3 - 2 \cdot x^2 - x + 2)$	[-1, 2]
8	$2 \cdot e^{-3x} \cdot \cos(2 \cdot \pi \cdot x + 1)$	[0, 3]
9	$4 \cdot x^3 \cdot \sin(2 \cdot \pi \cdot x - 3)$	[-2, 2]
10	$x^4 - 4 \cdot x^3 + x^2 - 6x$	[-3,1]
11	$8 \cdot e^{2x-1} \cdot \sin(2 \cdot \pi \cdot (x + 2))$	[-2, 1]
12	$5 \cdot x^3 \cdot \cos(4 \cdot \pi \cdot x - 3)$	[-2, 2]
13	$3 \cdot (x^2 - 4) \cdot (x^2 - 1)$	[-3, 3]

№	функция	интервал
14	$4 \cdot e^{2x+1} \cdot \sin(3 \cdot \pi \cdot (x+1.5))$	[-2,3]
15	$3 \cdot x^2 \cdot \sin(3 \cdot (\pi \cdot x-1))$	[-1, 1]
16	$(x^2-3 \cdot x+2) \cdot (x^2-7 \cdot x+12)$	[0, 4]
17	$7 \cdot e^{3x} \cdot \sin(2 \cdot \pi \cdot (x-2.5))$	[-2, 2]
18	$2 \cdot x^2 \cdot \cos(3 \cdot \pi \cdot x+2)$	[-1,2]
19	$x \cdot (x^2-0,25) \cdot (x^2-3 \cdot x+2)$	[0, 2]
20	$3 \cdot e^{2x-1} \cdot \sin(\pi \cdot x+2)$	[-3, 3]
21	$5 \cdot x^3 \cdot \sin(\pi \cdot x-3)$	[-1, 1]
22	$4 \cdot (2 \cdot x^2-5x+3) \cdot (2 \cdot x^2-9 \cdot x+10)$	[0, 3]
23	$2 \cdot e^{2(x+1)}) \cdot \cos(3 \cdot \pi \cdot (x+1))$	[-2, 2]
24	$(x^3-8) \cdot \cos(2 \cdot \pi \cdot x-3)$	[0, 2]

Лабораторная работа № 6
Работа с файлами последовательного доступа

Последовательным доступом называется способ, при котором данные считываются последовательно от начала файла. Чаще всего он применяется для считывания данных из текстового файла формата TXT, в котором содержится последовательность ASCII-кодов символов. Каждая запись в таком файле представляет собой строку символов, заканчивающуюся кодами управляющих символов «13» и «10» (которые называются соответственно «возврат каретки» и «перевод строки»).

Для открытия текстового файла используется оператор

Open *имя_файла* **For** *тип_доступа* **As #** *номер*

Здесь *имя_файла* – имя открываемого файла с указанием пути доступа к нему, *тип_доступа* – тип доступа (**Input** – для чтения, **Output** – для вывода, **Append** – для добавления), *номер* – порядковый номер от 1 до 255. В дальнейшем при работе с открытым файлом указывается не имя, а его порядковый номер.

Чтобы закрыть файл используется оператор **Close** (*номер).*

Для считывания одной строки из открытого для чтения текстового файла используется оператор **Input**:

Input # *номер*, *имя_переменной*

где *имя_переменной* – имя переменной типа **String**.

При открытии файла в его начало автоматически устанавливается так называемый «файловый указатель». После команды **Input** он перемещается к началу следующей строки. После считывания всех строк он устанавливается в конец файла. При попытке выполнить еще одну команду **Input** в этом случае будет выдано сообщение об ошибке (код 62).

Для того, чтобы избежать аварийного завершения программы можно использовать логическую функцию **EOF**(*номер*), где *номер* – номер открытого файла. Например, для считывания данных из текстового файла A.TXT в строковый массив MAS(i) можно использовать цикл типа **Do Until … Loop**:

```
i=0
Do Until EOF (1)
        i=i+1
        Input # 1, MAS(i)
Loop
```

Полезной может оказаться также функция **LOF**(*номер*), возвращающая количество байтов памяти, занимаемых открытым файлом.

Запись данных в файл, открытый для записи, осуществляется командой

Print *#номер, список_вывода*

Здесь *список_вывода* – одна или несколько переменных (или строки символов, взятые в кавычки), разделенных либо запятыми (,), либо точками с запятой (;). В первом случае выводимые значения будут записаны в виде одной строки и разделены пробелами, во втором – выведены без разделителя, слитно. Если поставить точку с запятой в конец списка вывода, то следующая порция данных, выводимая оператором будет дописана в ту же строку.

Например, в результате выполнения двух операторов

```
Print #1, “Иванов”
Print #1, “Петров”
```

в открытым для вывода под номером 1 файле окажутся две строки по одной фамилии в каждой.

Если в конец первого оператора поставить точку с запятой:

```
Print #1, “Иванов” ;
Print #1, “Петров”
```

то будет записана одна строка в виде «ИвановПетров».

Следует отметить, что при попытке открыть для чтения несуществующий файл, будет выдано сообщение об ошибке (код ошибки 53). Если же дана команда открыть несуществующий файл для вывода или добавления, сообщения об ошибке не будет, файл с указанным именем будет создан.

Порядок выполнения.

1. Исходные данные – текстовые файлы, содержащие данные в соответствии с указанным в варианте заданием, – студент должен подготовить самостоятельно, используя текстовый редактор «Блокнот».

2. В каждом из вариантов требуется разработать программу, с помощью которой исходный файл (или файлы) открывается для чтения, данные считываются, обрабатываются и результаты записываются в новый файл.

3. При разработке проекта необходимо предусмотреть возможность вывода на форму в текстовые окна содержимое как исходных файлов, так и создаваемых для того, чтобы можно было сразу оценить правильность работы программы.

Варианты задания.

1. Текстовый файл содержит несколько строк с фамилиями студентов. Создать новый файл, в котором будут те же фамилии, но расположенные в алфавитном порядке.

2. Текстовый файл содержит одну строку, в которой записаны фамилии, разделенные пробелами. Создать новый файл, в котором будут те же фамилии, но расположенные в разных строках.

3. Текстовый файл содержит несколько строк, в каждой из которых записано целое число. Разделить этот файл на два, записав в один только четные числа, а в другой – нечетные.

4. Текстовый файл A содержит две строки с текстом, а файл B – одну строку. Создать файл C, в котором будут эти же три строки, но строка из файла B должна оказаться между строками из файла A.

5. Текстовый файл A содержит текст из заглавных (прописных) русских букв. Создать файл B, в котором будет тот же текст, но из строчных букв.

6. Текстовый файл A содержит текст с русскими и латинскими символами. Создать новый файл B, котором из этого текста будут удалены латинские символы.

7. Текстовый файл содержит три строки символов. Создать новый файл, в котором эти строки будут записаны в обратном порядке.

8. В текстовых файлах A и B записаны два массива по N чисел в каждом, каждое число в отдельной строке. Создать файл из N строк, в каждой из которых записана сумма соответствующих чисел из файлов A и B.

9. В файле A построчно записаны фамилии студентов. В файле B – две фамилии из тех, которые приведены в A. Создать файл, в который записать все фамилии из файл A кроме тех, что указаны в файле B.

10. Текстовый файл A содержит несколько строк с фамилиями студентов, а B – строки с их именами. Создать новый файл, в котором строки будут содержать фамилии тех же студентов вместе с их именами.

11. Текстовый файл содержит несколько строк с фамилиями. Создать файл, в котором эти фамилии будут записаны в одну строку и разделены запятыми.

12. В текстовом файле A записаны целые четные числа, в файле B – нечетные. Количество четных и нечетных чисел может быть разным. Создать файл, в котором будет одинаковое количество четных и нечетных чисел, причем они чередуются.

13. Текстовые файлы A и B содержат по две строки с текстом. Создать файл C, в котором будут эти же строки, в следующем порядке: 1-я из A, 1-я из B, 2-я из A, 2-я из B.

14. Текстовый файл A содержит текст из строчных русских букв. Создать файл B, в котором будет тот же текст, но из заглавных (прописных) букв.

15. Текстовый файл A содержит текст с русскими и латинскими буквами. Создать новый файл B, котором из этого текста будут удалены русские буквы.

16. Текстовый файл A содержит три строки символов. Создать новый файл, в котором будет только первая и третья строки из A.

17. В текстовых файлах A и B записаны два массива по N чисел в каждом, каждое число в отдельной строке. Создать файл из N строк, в каждой из которых записано меньшее из соответствующих чисел из файлов A и B.

18. В файле A построчно записаны фамилии студентов. В файле B – одна фамилия из тех, которые приведены в A. Создать файл, в который записать все фамилии из файл A, которые записаны после указанной в файле B фамилии.

19. Текстовый файл содержит несколько строк с фамилиями. Создать файл, в котором эти фамилии будут записаны в одну строку и разделены пробелами.

20. Текстовый файл A содержит текст с русскими и латинскими буквами. Создать новый файл B, заменив в тексте все латинские буквы звездочками (*).

21. Текстовый файл содержит одну строку, в которой записаны фамилии, разделенные запятыми. Создать новый файл, в котором будут те же фамилии, но расположенные в разных строках.

22. В файле A построчно записаны фамилии студентов. В файле B – одна фамилия из тех, которые приведены в A. Создать файл, в который записать все фамилии из файл A, которые записаны перед указанной в файле B фамилией.

23. В текстовых файлах A и B записаны два массива по N чисел в каждом, каждое число в отдельной строке. Создать файл из N строк, в каждой из которых записано большее из соответствующих чисел из файлов A и B.

24. В файле A построчно записаны фамилии студентов. В файле B – две фамилии, одна из них есть в A, другой нет. Создать файл, в который записать все фамилии из файла A и добавить из B новую.

Лабораторная работа № 7
Работа с дисками и папками

На панели инструментов окна Visual Basic есть три объекта, с помощью которых можно осуществлять просмотр существующих на диске папок и выбрать нужные файлы для последующей работы с ними.

Это объекты **DriveListBox** (список дисков), **DirListBox** (список папок) и **FileListBox** (список файлов). При размещении экземпляров этих объектов на форму им будут присвоены имена соответственно **Drive1**, **Dir1** и **File1**.

Основным свойством объекта **Drive1** является свойство **Drive** – имя выбранного диска из числа доступных в данный момент. Присвоение значения этому свойству происходит после выбора нужного имени из списка дисков (событие **Change**).

У объекта **Dir1** основным свойством является **Path** – после выбора из списка папок (событие **Change**) там сохраняется имя выбранной папки или список вложенных папок, разделенных слэшом (обратной косой чертой).

Объект **File1** на форме представлен в виде текстового окна со списком файлов текущей папки. При выборе из этого списка нужного имени с помощью мыши (событие **Click**) свойству **FileName** присваивается имя выбранного файла с указанием пути к текущей папки. Для того, чтобы связать список файлов с выбранной папкой, используется свойство **Path**.

В качестве примера ниже приведен текст трех процедур, которые запускаются при наступлении указанных событий. В результате работы программы выбранный текстовой файл открывается, считывается и текст выводится в текстовое окно.

Примечание. Для того, чтобы в текстовое поле **Text2** можно было вывести больше одной строки, свойству **Multiline** необходимо присвоить значение **True**.

```
Private Sub Drive1_Change()      ' после выбора нужного диска из списка
Dir1.Path = Drive1.Drive         ' его имя передается свойству Dir1.Path
ChDir Dir1.Path                  ‘ и выполняется команда “сменить директорию”
End Sub

Private Sub Dir1_Change()        ' после выбора нужной папки из списка
File1.Path = Dir1.Path           ' путь к этой папке передается свойству File1.Path
End Sub

Private Sub File1_Click()                        '  после выбора нужного файла
File_name = Dir1.Path + "\" + File1.FileName     ‘ переменной File_name присваивается
                                                 ' имя файла и добавляется путь к нему
Open File_name For Input As #1                   ’ файл открывается для чтения
ST = ""                                          ' в переменную ST построчно
Do Until EOF(1)                                  ' считываются записи из файла
    Input #1, S                                  ' и добавляются управляющие коды
     ST = ST + S + Chr(13) + Chr(10)             ' <13>, <10>
Loop                                             ‘ файл закрывается
Close #1                                         ' значение переменной ST выводится
Text2.Text = STT                                 ‘ в окно Text2
End Sub
```

Еще одну возможность выбора папки и файла из списка имеющихся на диске дает использование объекта **CommonDialog**. Этот объект отсутствует изначально на панели инструментов. Для того, чтобы его туда поместить, необходимо в основном меню выбрать пункт **Project** в развернувшемся меню выбрать опцию **Components…**, в появившемся окне (рис. 5.11) отметить пункт **Microsoft Common Dialog Control**.

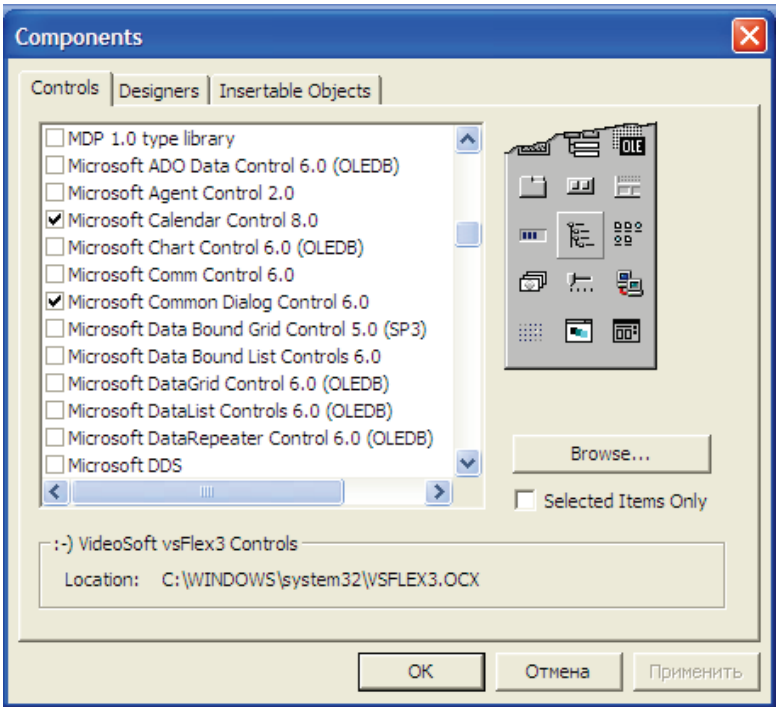


Рис.5. Окно выбора компонентов

В результате в панель инструментов будет добавлена пиктограмма объекта **CommonDialog**. У этого объекта есть ряд методов, используя которые можно открывать диалоговые окна для выбора файла (метод **ShowOpen**); для сохранении файла на диске (метод **ShowSave**); для выбора цвета из палитры (метод **ShowColor**); для выбора имени шрифта, его размер и стиля (метод **ShowFont**).

Если поместить объект **CommonDialog** на форму, ему будет присвоено имя **CommonDialog1**. Для того чтобы в чтобы организовать режим выбора нужного файла из диалогового окна **Open** в соответствующей процедуре достаточно написать строку

```
CommonDialog1.ShowOpen
```

В режиме выполнения эта команда вызовет появление диалогового окна выбора файла. После выбора нужной папки и файла и нажатия кнопки «Открыть» имя выбранного файла передается свойству **FileName** объекта **CommonDialog**. После этого он может быть открыт, например для чтения, командой

```
Open CommonDialog.FileName For Input As # 1
```

Аналогично выполняется процедура сохранения файла на диск. Предположим, что надо сохранить в текстовом файле строку S. Последовательность команд может быть такой

```
CommonDialog1.ShowSave
Open CommonDialog.FileName For Output As # 1
Print #1, S
Close #1
```

Порядок выполнения:

1) модифицировать проект, созданный при выполнении предыдущей работы, добавив на экранную форму элементы **Drive1**, **Dir1** и **File1**.

2) добавить процедуры, позволяющие выбрать с помощью указанных элементов текстовый файл, подлежащий обработке.

СОДЕРЖАНИЕ

Для заметок