

Министерство образования и науки Российской Федерации  
Волгоградский государственный архитектурно-строительный университет

# ПРОГРАММИРОВАНИЕ В СРЕДЕ MS VISUAL BASIC

Методические указания к лабораторным работам  
по дисциплине «Информатика»

*Составила Н. Н. Потапова*

Волгоград. ВолгГАСУ. 2016



© Федеральное государственное бюджетное  
образовательное учреждение  
высшего профессионального образования  
«Волгоградский государственный  
архитектурно-строительный университет», 2016

УДК 004.43(075.8)  
ББК 22.183.492я73  
П784

**Программирование** в среде MS VISUAL BASIC [Электронный ресурс]: методические указания к лабораторным работам по дисциплине «Информатика» / М-во образования и науки Рос. Федерации, Волгогр. гос. архит.-строит. ун-т; сост. Н. Н. Потапова. — Электронные текстовые и графические данные (1,0 Мбайт). — Волгоград : ВолгГАСУ, 2016. — Учебное электронное издание сетевого распространения. — Систем. требования: PC 486 DX-33; Microsoft Windows XP; Internet Explorer 6.0; Adobe Reader 6.0. — Официальный сайт Волгоградского государственного архитектурно-строительного университета. Режим доступа: <http://www.vgasu.ru/publishing/on-line/> — Загл. с титул. экрана.

Предназначены для формирования у студентов навыков создания Windows-приложений в системе программирования MS Visual Basic, которая является мощным программным средством, позволяющим решать большой спектр практических задач. Издание содержит теоретическую и практическую части. Теоретическая часть охватывает основные аспекты программирования на языке Visual Basic. Практическая часть содержит четыре лабораторные работы. В процессе выполнения лабораторных работ студенты осваивают средства создания пользовательского интерфейса приложений, а также средства ввода, редактирования и отладки программного кода. Каждая лабораторная работа содержит индивидуальные задания, пример выполнения, пояснения работы программных кодов и контрольные вопросы.

Для студентов направления «Строительство» очной формы обучения.

**УДК 004.43(075.8)**  
**ББК 22.183.492я73**

План выпуска учебн.-метод. документ. 2016 г., поз. 18

Минимальные систем. требования:  
PC 486 DX-33; Microsoft Windows XP; Internet Explorer 6.0; Adobe Reader 6.0

Подписано в свет 29.08.2016.  
Гарнитура «Таймс». Уч.-изд. л. 2,5. Объем данных 1,0 Мбайт

Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования  
«Волгоградский государственный архитектурно-строительный университет»  
400074, Волгоград, ул. Академическая, 1  
<http://www.vgasu.ru>, [info@vgasu.ru](mailto:info@vgasu.ru)

# I. Теоретическая часть

## Введение

Объектно-ориентированное программирование является самой популярной технологией программирования. Visual Basic – это современная высокоэффективная интегрированная (IDE) среда быстрой разработки приложений, и ее можно поставить в один ряд с такими популярными средствами, как C++ или Delphi.

Система программирования Visual Basic включает в себя текстовый редактор, конструктор форм, компилятор и отладчик.

Основные понятия:

- **объект** – форма, а также элементы управления, находящиеся на форме, который можно использовать для создания приложения, каждый объект обладает определенными свойствами, методами и событиями;

- **свойство** (Property) – характеристика формы или элемента управления, которая определяет изображение объектов на экране, их поведение и название;

- **метод** (Method) – действие, которое может выполнить элемент управления или форма, например, метод Hide формы удаляет ее с экрана, а метод Show вновь выводит ее на экран;

- **событие** (Event) – действие, которое может выполнить пользователь над объектом, например, щелчок мышью, или нажатие клавиши, выбор пункта меню;

- **инкапсуляция** - это объединение данных и черт поведения объекта в одном пакете;

- **наследование** - это процесс, посредством которого один объект может наследовать свойства другого и добавлять к ним черты, характерные только для него, смысл и универсальность наследования в том, что не надо каждый раз описывать новый объект, достаточно указать "родителя" (базовый класс) и описать отличительные особенности;

- **полиморфизм** (от греч. "многоликость") означает, что метод с одним именем может применяться к родственным объектам, т. е. "потомки" знают какие методы они должны использовать.

Просмотреть перечень событий, свойств и методов можно в окне Object Browser.

## 1.1. Запуск среды MS Visual Basic

Запуск среды Visual Basic осуществляется командой:  
Пуск/Программы/MS Visual Basic.

## 1.2. Создание нового проекта

При первом запуске Visual Basic, запускается мастер Project Wizard, на экране появляется диалоговое окно **New Project** (Новый проект). Здесь выбирается один из нескольких типов шаблонов проектов.

Окно состоит из 3-х вкладок (рис.1):

1. **New** (новый) – будет создаваться новый проект;
2. **Existing** (существующий) – содержит список всех проектов, которые хранятся на компьютере;
3. **Recent** (последние) – содержит список проектов, с которыми производились работы в последнее время.



Рис. 1. Окно New Project (вкладка New)

Во вкладке **New**, присутствуют много шаблонов проектов, которые можно создать в среде Visual Basic, для работы мы будем использовать шаблон **Standart.EXE** (стандартный EXE – файл), после чего можно нажать кнопку «Открыть». Появится интегрированная среда разработки Visual Basic.

### 1.3. Сохранение проекта

Для сохранения проекта следует выполнить команду:

*File / Save Project As*

Появится диалоговое окно **Save File As**, в котором необходимо указать папку для хранения проекта и имя файла формы.

Каждая форма сохраняется как отдельный файл. Visual Basic запрашивает имя для каждой формы. После ввода имени файла необходимо щелкнуть левой клавишей мыши по кнопке «Сохранить».

При создании файла формы Visual Basic добавляет к имени формы расширение .frm.

После сохранения всех форм проекта отображается диалоговое окно **Save Project As**, в котором надо задать имя проекта (Visual Basic автоматически добавит к нему расширение .vbp) и щелкнуть левой клавишей мыши по кнопке «Сохранить».

Для сохранения проекта со старым именем следует просто щелкнуть по пиктограмме с изображением дискеты.

*Последующие сохранения.* По завершении сеанса работы следует ввести команду меню *File / Exit*. Появится окно Microsoft Visual Basic с вопросом *Save Changes to the following Files?* (Сохранить изменения следующих файлов?) Следует щелкнуть по кнопке Yes.

Для сохранения проекта в виде загрузочного файла следует из меню File выполнить команду: *Make Project.exe*.

## 1.4. Открытие проекта

Открытие проекта можно осуществить несколькими способами.

1-й способ. Открыть рабочую папку из окна программы «Проводник», где был сохранен проект.

Найти файл с именем проекта, например, *Project1.vbp*. Значок этого файла имеет вид:



Сделать по нему двойной щелчок.

2-й способ. Запустить систему программирования *Visual Basic*. Ввести команду меню *File / ...Project1.vbp*.

3-й способ. Нужный существующий проект можно выбрать во вкладке *Existing* или *Recent* в зависимости от даты последнего сохранения проекта.

После выполнения приведенных выше действий в соответствии с одним из способов открывается существующий проект в главном окне Visual Basic, где и выполняются все работы с ним.

## 1.5. Визуальная интегрированная среда Visual Basic

Интегрированная среда Visual Basic представлена окнами, которые позволяют управлять составными частями проекта (рис. 2).

Если какие-либо окна не видны, то их можно вызвать на экран с помощью меню View (Вид):

View/Project Explorer (Проводник проекта)

View/Properties Windows (Окно свойств)

View/Tool Box (Панель инструментов)

View/Immediate Window (Окно отладки)

View/Code (Окно редактора кода для текущей формы или модуля)

Для нормальной работы в Visual Basic необходимо, чтобы в рабочем окне были активны (включены): панель инструментов, окно проводника, окно свойств.

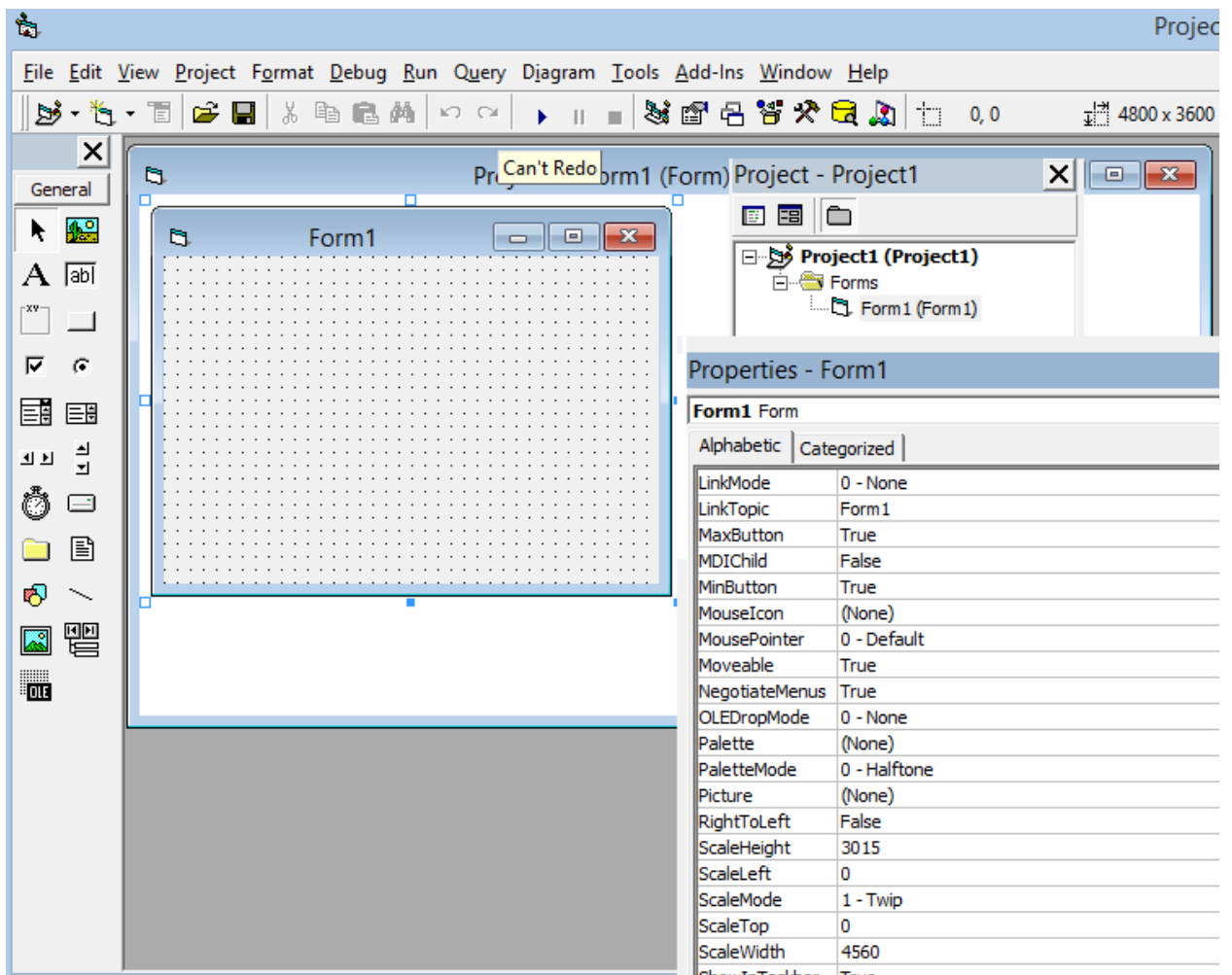


Рис.2. Интегрированная среда разработки

1. **Основное** окно, которое находится в верхней части экрана, появляется как в режиме проектирования, так и в режиме выполнения. В этом окне располагается меню Visual Basic и панель инструментов (рис.3), выполняющих наиболее важные функции. Кроме того, имеются кнопки «Свернуть», «Развернуть», «Заккрыть».

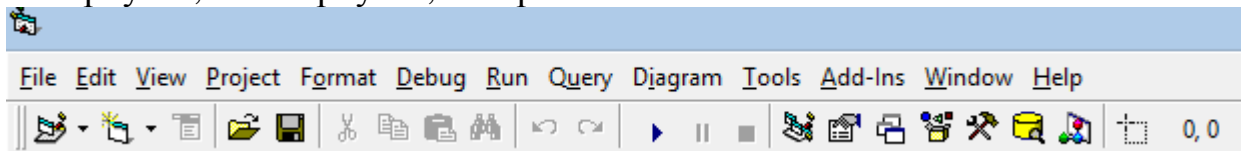


Рис. 3. Меню и панель инструментов основного окна

Меню представлено следующими командами:

**File** содержит команды для работы с файлом проекта (создание нового, открытие существующего проекта, его печать, сохранение проекта и формы, создание exe-файла и т.д.);

**Edit** содержит команды редактирования (вырезать, скопировать, вставить и т.д.);

**View** содержит команды настройки главного окна проекта, в частности определяет окна, которые располагаются в рабочей области;

**Project** содержит команды добавления и удаления форм, программных модулей, пользовательских управляющих элементов;

**Format** определяет размеры и расположение элементов управления и формы;

**Debug** используется для отладки приложений (можно остановить приложение, расставить точки прерывания и др.);

**Run** содержит команды запуска программы, завершения выполнения программы, продолжение после останова, включение режима прерывания, запуск из режима прерывания;

**Tools** содержит команды, которые позволяют добавить процедуры и задать значения их атрибутов, разработать собственное меню;

**Add-Ins** содержит дополнительные утилиты – надстройки. В него включаются подкоманды для проектирования и заполнения баз данных;

**Window** используется для работы с окнами IDE (расположение, упорядочивание, быстрый переход);

**Help** содержит справочную информацию по работе с Visual Basic и др.

2. **Окно формы (Form)** содержит форму, которая является основой интерфейса пользователя (рис.4 ). При загрузке Visual Basic для создания проекта она первоначально пуста, содержит только полосу заголовка и рабочую область, на которой имеется сетка из точек для выравнивания элементов управления. При работе приложения сетка не видна.

Процесс конструирования формы начинается с задания ее свойств, например: название (текст в заголовке формы), имя этой формы, высота и ширина формы, шрифт и др. Размеры формы можно установить и с помощью маркеров на ее границе.

Далее форма заполняется элементами управления.

В состав проекта может входить несколько форм, с каждой из которых может быть связан свой программный модуль.

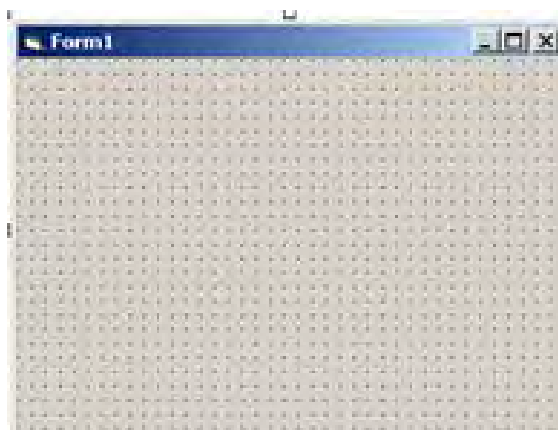


Рис.4. Окно формы



3. **Окно свойств** отображает свойства текущего объекта.

Рабочая часть окна Properties состоит из двух вкладок Alphabetic и Categorized, в которых свойства объекта представлены соответственно в алфавитном порядке либо группируются по категориям (рис.5).

Набор свойств в списках зависит от типа элемента управления. Каждый из списков состоит из двух столбцов: в первом указаны названия свойств, а во втором – их значения. Первоначально каждое свойство имеет значение, заданное по умолчанию. Значение любого свойства можно изменить путем редактирования.

Например, для формы можно установить следующие свойства:

**Name** – имя объекта;

**Caption** – надпись в заголовке объекта;

**Font** – шрифт;

**Height** – высота;

**Width** – ширина;

**Left** – левая граница;

**Top** – верхняя граница;

**BackColor** – цвет фона;

**Icon** – рисунок в строке заголовка формы и др.

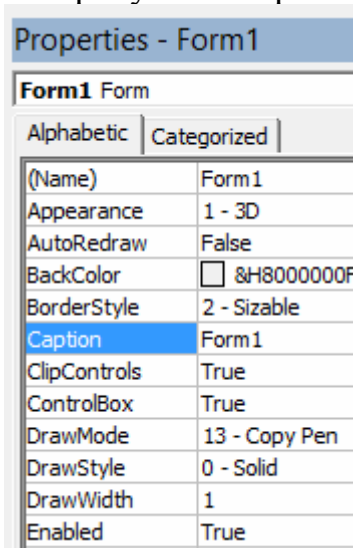


Рис. 5. Окно свойств

4. **Окно элементов управления** (или **Панель инструментов**) (рис.6) служит для организации интерфейса между пользователем и компьютером. Элементы управления размещаются в окне *Панель инструментов*, откуда их можно перетащить с помощью мыши на форму.

Основные элементы управления следующие:

**A** Надпись (Label) используется для отображения текста, который пользователь не может изменить с клавиатуры;




текстовое поле (TextBox) применяется для ввода данных;







Рис.6. Окно элементов управления


 командная кнопка (CommandButton) предназначена для того, чтобы начать, прервать или закончить какой-либо процесс;


 флажок (CheckBox) используется для отметки (включения) какого-либо параметра. Щелчок мыши приводит к появлению “галочки”, а значение (Value) становится равным 1, повторный щелчок мышью убирает “галочку” (приводит в первоначальное состояние) и ее значение становится равным 0. Программно может быть установлено и третье состояние – “отмечено, но недоступно”, при этом его значение равно 2;

 переключатель (OptionButton) может иметь два состояния: включено (на центре кружка имеется точка, а значение равно True) и выключено (в центре кружка отсутствует точка, а значение равно False);

 рамка (Frame) является элементом-контейнером и предназначена для объединения в группу нескольких элементов (например, флажки или переключатели). В частности, переключатели без рамки использоваться не могут. Если в рамке находятся флажки, то выбранными могут быть сразу несколько (в том числе и ни одного или даже все), а если переключатели, то всегда только один.

Для создания группы элементов управления вначале создают рамку, а затем в ней управляющие элементы.

 простой список (ListBox) служит для выбора из него одного или нескольких элементов. Элементы можно добавлять в список и удалять из него;

 поле со списком (ComboBox) представляет собой комбинацию текстового поля и списка и используется, когда нужно не только выбрать

элементы, но и вводить значение непосредственно в текстовое поле, после чего это значение автоматически становится элементом списка;



изображение (Image) используется для отображения рисунка, например, для создания подложки (фона) формы;



графическое окно (PictureBox) позволяет отображать и редактировать графическое изображение (используется OLE-технология);



элемент OLE (OLE Container) предназначен для вставки в форму любого OLE-объекта (текста из Word, таблицы или диаграммы Excel и т.д.), которые при необходимости можно редактировать.

**5. Проводник проекта** (рис.7) отображает группы объектов (формы, модули и т.д.), позволяет увидеть иерархию этих компонентов и перейти к работе с нужным, щелкнув по значку, предварительно выделив его на иерархическом дереве.

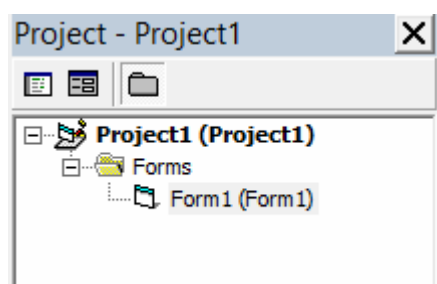


Рис. 7. Окно проводника проекта

### 6. Окно редактирования кода.

С каждым элементом управления могут быть связаны различные события (щелчок мышью, двойной щелчок мышью, получение фокуса, открытие или изменение размера формы и т.д.). Любое из этих событий может быть обработано процедурой, составленной на языке Visual Basic. Ее текст вводится в окне кода.

Для создания **процедуры обработки события**, связанного с элементом управления, достаточно дважды щелкнуть на нем или выполнить команду View/Code (Вид/Код). При этом появляется окно кода (рис.8), в котором подготовлен заголовок и окончание будущей процедуры.

Например,

```
Private Sub Command1_Click ()  
<Тело процедуры>  
End Sub
```

Пользователь должен ввести только тело процедуры.

В заголовке процедуры указывается имя соответствующего элемента управления (в нашем случае кнопка Command1) и связанного с ним события (в нашем случае щелчок мышью Click()).

Если нужно написать процедуру для другого элемента управления, то его имя выбирается из левого списка окна редактирования кода **General** а связанного с ним события из правого списка **Declaration**.

Каждая процедура отделяется тонкой линией от предыдущей.

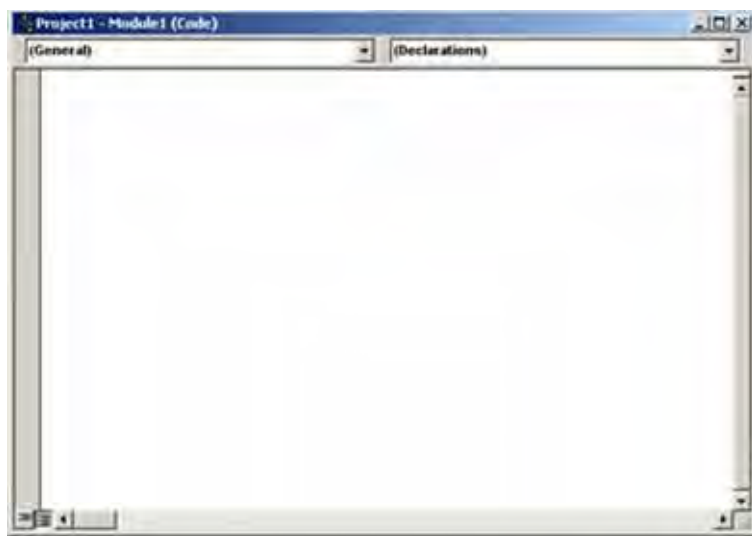


Рис. 8. Окно редактирования кода

Окно кода в раскрывающемся списке раздела **General** содержит имена формы, а также имена всех объектов управления, находящихся на поверхности формы. В разделе **Declaration** будут содержаться имена всех событий, которые доступны для выбранного объекта.

7. **Окно Immediate** предназначено для отладки приложения. В этом окне можно осуществлять печать значений переменных и массивов в процессе выполнения программы. Для этого в нужное место текста программного кода следует ввести временный дополнительный оператор **Debug.Print список вывода** и запустить программу на выполнение.

В этом окне можно распечатать значения переменных и свойств объектов, а также ввести выражение, в состав которого входят имена переменных программы и вычислить его значение.

Для этого в окне Immediate надо напечатать знак «?»», который заменяет слово Print и после набора, например, выражения, нажать клавишу Enter. Строкой ниже появится значение вычисленного выражения.

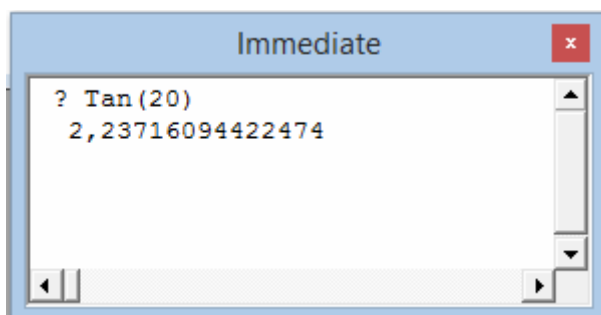


Рис. 9. Окно Immediate

**8. Окно расположения формы (Form Layout)**, которое используется для определения размеров формы и ее положения на экране, и **Окно просмотра объектов (Object Browser)**, которое служит для просмотра свойств, методов, констант, типов и переменных всех объектов в данном проекте, вызываемые соответствующими командам из меню View (Вид), на начальной стадии изучения Visual Basic как правило, используются редко. Поэтому не рекомендуется выводить их на экран, чтобы не загромождать рабочую область главного окна.

Таким образом, в программах на Visual Basic пользовательский интерфейс представляется в виде форм, на которых помещаются элементы управления, а процесс обработки данных описывается программным кодом. Этот код связывается или с формой, или с элементами управления, или может быть представлен отдельным модулем.

## **1.6. Программирование в среде MS Visual Basic. Основные понятия и конструкции.**

Любой язык, в том числе и язык программирования, подчиняется определенным правилам. Их принято разделять на правила, определяющие синтаксис языка и правила, определяющие его семантику.

**Синтаксис** языка – это совокупность правил, определяющих допустимые конструкции языка (слова, предложения) языка, его форму.

**Семантика** языка – совокупность правил, определяющих смысл синтаксически корректных конструкций языка, его содержание.

Описание синтаксиса языка начинается с определения алфавита языка и его простейших конструкций.

**Алфавит языка** программирования Visual Basic включает:

- строчные и прописные буквы латинского алфавита; буквы кириллицы (могут использоваться только в качестве имен переменных и констант);
- цифры от 0 до 9;
- специальные символы: . , + - \* / = : > < ( ) ^ % & ! # \$ @ >= <= и т.д.
- служебные слова.

### **1. Переменные**

**Переменная** - это именованная область памяти, используемая для временного хранения данных.

Переменную в программе необходимо сначала описать т. е. указать имя, тип, размер.

В отношении имен переменных в Visual Basic действуют следующие правила:

- длина имени должна быть не более 255 символов;
- имя не может содержать таких символов, как точка, пробел, % & ! # \$ @;

- имя должно начинаться с буквы и может содержать любую последовательность букв, цифр и символов, кроме специальных;
- имя должно быть уникальным внутри данного проекта (модуля);
- в качестве имени не могут быть использованы служебные слова.

В Visual Basic имеется много различных типов данных. В таблице ниже представлены типы, которые наиболее часто используются.

### Некоторые типы данных

Тип	Описание	Размер области памяти
Boolean	Логическое значение	2 байта
Byte	Однобайтное целое число	1 байт
Integer	Целое число	2 байта
Long	Длинное целое число	4 байта
Single	Число с плавающей точкой одинарной точности	4 байта
Double	Число с плавающей точкой двойной точности	8 байт
Currency	Число с фиксированной точкой (денежный тип)	8 байт
String	Текстовая строка (последовательность символов от 0 до 255)	Зависит от числа символов (1 байт на каждый символ)
Date	Дата и время	
Variant	Любое значение из перечисленных выше	8 байт 16 байт

По умолчанию все данные имеют тип *Variant*.

### Описание переменных

При программировании, необходимо все переменные описать. Для этого используют оператор Dim. Синтаксис этого оператора:

Dim ИмяПеременной [As <ТипПеременной>]

Квадратные скобки говорят о том, что этот параметр является необязательным и может использоваться по необходимости.

Например:

Dim P As String - описание строковой переменной

Dim k As Integer - описание числовой переменной целого типа.

Оператор объявления типа записывается в начале программы или процедуры в разделе объявлений.

С помощью одного оператора можно объявить сразу несколько переменных, например:

```
Dim n As Integer, A As Single
```

Рекомендуется все переменные объявлять явно. Для этого нужно в коде перед процедурами задать оператор **Option Explicit**. В этом случае Visual Basic отслеживает появление новых имен. И если переменная не объявлена в операторе Dim, то будет выдано сообщение об ошибке.

При использовании процедур и функций, а также параметров в функциях или процедурах следует следить за областью их видимости (или доступности) из разных разделов программы. Неправильное обращение к области видимости часто является причиной ошибок.

Существует несколько способов, уточняющих область видимости объявляемой переменной, процедуры или функции:

```
Private Имя_1 [As тип1], Имя_2 [As тип2],...
```

```
Static Имя_1 [As тип1], Имя_2 [As тип2],...
```

```
Public Имя_1 [As тип1], Имя_2 [As тип2],...
```

В том случае, когда переменная объявляется с ключевым словом **Dim**, областью ее использования будет только та процедура, в которой она была описана. Такие переменные называются локальными (закрытыми). После выполнения процедуры их значения будут потеряны. В разных процедурах для локальных переменных можно использовать одинаковые имена (например, в счетчиках циклов). Если используется ключевое слово **Static**, то переменная тоже будет локальной, однако ее последнее значение сохраняется. Если требуется, чтобы переменные, функции и процедуры были доступны в нескольких разделах одного модуля, следует использовать оператор **Private** и объявление переменных следует осуществлять в разделе **Declarations** текущего модуля. Когда нужно, чтобы переменная была доступна всем модулям, ее следует объявить в этом разделе оператором **Public**. Такие переменные называются глобальными (открытыми).

Переменные, значения которых не меняются в процессе выполнения программы, называются **константами**. Константа характеризуется именем и значением. В отношении имен констант действуют такие же правила, как и в отношении имен переменных. Константы также, как и переменные, могут быть разных типов, в зависимости от типов данных, которые они представляют.

Синтаксис объявления констант следующий:

```
Const ИмяКонстанты [As <Тип>]= ЗначениеКонстанты
```

Перед ключевым словом Const может стоять слово Public или Private. Ключевое слово Public означает, что данная константа может быть использована во всех процедурах программы (общедоступная константа), а

ключевое слово `Private` – только в той процедуре, где записан оператор `Const` (локальная константа). Если не указано ни одно из них, то подразумевается `Private`.

Например,  
`Const R As Integer = 9`

### Оператор присваивания

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

ИмяПеременной=Выражение

Например,  
`X=4`  
`Y=Y+2`

*Примечание.* С помощью оператора присваивания можно установить или изменить значение свойства объекта, используя синтаксис:

Объект.Свойство=Значение

**Пример.** Свойству `Height` (размер по вертикали) метки с именем `MyLable` присваивается значение 1000:

`MyLable.Height=1000`

Получить значение свойства можно, используя синтаксис:

Переменная=Объект.Свойство

## 2. Массивы

Последовательность данных одного типа, расположенных в памяти последовательно, может образовывать массив.

Массивы бывают одномерные, которые можно представить в форме одномерной таблицы, и двумерные, которые можно представить в форме двумерной таблицы. Массив состоит из пронумерованной последовательности элементов. Номера в этой последовательности называются **индексами**. По умолчанию индекс массива начинается с нуля.

Объявление массива проводится аналогично объявлению переменных, необходимо только дополнительно указать диапазон изменения индекса. В *Visual Basic* имеются два типа массивов: **статические** (фиксированного размера) и **динамические** (когда количество элементов массива неизвестно заранее и будет определяться по ходу выполнения программы).

Синтаксис объявления массива фиксированного размера:

`Dim ИмяМассива(размерность1, размерность2) [As <тип >].`

Для явного указания границ массива применяется служебное слово *To*.

**Пример 1.** Объявление числового массива, содержащего 20 элементов,:

`Dim P(1 To 20) As Single`

В этом случае нумерация элементов массива будет начинаться с 1. Массив состоит из 20 элементов.



Если не указывать явно номер первого элемента массива, объявив его следующим образом:

```
Dim P(20) As Single
```

то, так как по умолчанию Visual Basic нумерует массивы начиная с 0, этот одномерный массив будет состоять из 21 элемента.

**Пример 2.** Объявление двумерных массивов:

```
Dim B(1,9) As Integer
```

```
Dim B(1 To 2,1 To 10) As Integer.
```

Значения верхней и нижней границ области памяти для статического массива не могут быть изменены в программе.

Для объявления **динамического** массива применяются операторы Dim и ReDim.

Так как динамический массив в начале своего жизненного цикла не содержит ни одного элемента, то он объявляется так же, как обычный массив, используя ключевое слово Dim и имя массива с пустыми скобками.

```
Dim Y() As Integer
```

Затем в массив необходимо добавить элементы, в которых будут храниться значения. Для этого используется оператор ReDim, который может встречаться только в процедурах. Затем добавлять в него элементы по мере надобности. Это методика особенно полезна при работе с большим количеством элементов.

**Пример.**

```
Dim n As Integer
```

```
Dim A() As Integer
```

```
n=300
```

```
ReDim A(n) ' для A выделяется память из 2*300 байт*
```

**Примечание.** Для того, чтобы нумерация массивов начиналась с единицы, необходимо перед программным кодом ввести оператор **Option Base 1**.

### 3. Старшинство операций

Операции в Visual Basic образуют четыре общих категорий: арифметические, логические, сравнения и конкатенации.

**Выражение** состоит из данных, встроенных функций и скобок, соединенных знаками операций. Данные и встроенные функции, взятые отдельно, тоже являются выражениями. В **арифметические** выражения могут входить переменные числового типа и числа; над переменными и числами могут производиться различные арифметические операции, а также математические операции с функциями.

Вычисление выражения производится слева направо с учетом круглых скобок, а при их отсутствии определяется следующим старшинством групп операций: первыми выполняются арифметические операции, следом за ними – операции сравнения, а последними – логические операции.

**Арифметические** операции имеют следующий приоритет, который можно изменить с помощью скобок:

- 1) возведение в степень (^);
- 2) умножение и деление (\*,/);
- 3) целое деление (\);
- 4) деление по модулю (Mod);
- 5) сложение и вычитание;
- 6) слияние строк.

**Логические** операции имеют следующий приоритет:

- 1) отрицание (Not);
- 2) логическое умножение (And);
- 3) логическое сложение (Or).

**Операции сравнения:**

- 1) меньше (<);
- 2) меньше или равно (<=);
- 3) больше (>);
- 4) больше или равно (>=);
- 5) равно (=);
- 6) не равно (<>).

Все операции сравнения имеют одинаковый приоритет, т.е. выполняются в порядке их записи слева направо.

Операция **конкатенации** ( & ) - это строковая операция, применяемая для объединения нескольких строк в одну. Следует обратить внимание на то, что символ конкатенации & должен слева и справа обрамляться пробелами.

#### 4. Встроенные функции

Понятие функции в языках программирования близко к понятию функции в математике. Функция может иметь один или несколько аргументов. В программировании говорят, что функция возвращает свое значение, если задано значение ее аргументов. Синтаксис записи функции следующий:

ИмяФункции (СписокАргументов)

Основные категории встроенных функций:

- математические;
- функции обработки строк;
- функции даты и времени;
- функции преобразования форматов;
- функции цвета.

*Примечание.* Ниже приводятся наиболее часто употребляемые функции в инженерных расчетах.

#### Математические функции

Функция	Возвращаемое значение
Sin (x)	синус числа x
Cos (x)	косинус числа x

Tan (x)	тангенс числа x
Atn(x)	арктангенс числа x
Sqr(x)	квадратный корень из числа x
Log (x)	натуральный логарифм числа x
Exp(x)	показательная функция числа x
Int (x)	наибольшее целое число, не превышающее число x
CInt (x)	целое число, ближайшее к числу x
Fix (x)	целое число, равное числу x без дробной части
Abs (x)	абсолютное значение числа A
Rnd(x)	случайное число

Некоторые математические функции в языке Visual Basic отсутствуют. Наиболее часто используемые **функции и константы, не представленные в Visual Basic** следующие.

число пи:	$Pi=4*Atn(1);$
число e:	$e=exp(1);$
десятичный логарифм числа x:	$Y=Log(x)/Log(10) ;$
котангенс числа x:	$Y=1/Tan(x);$
арксинус числа x :	$Y=Atn(x/Sqr(1-x^2));$
арккосинус числа x:	$Y=2*Atn(1)- Atn(x/Sqr(1-x^2));$
перевод угла x из радиан в градусы:	$Y=180*x/(4*Atn(1)).$

### Функции обработки строк

Len(Строка) – возвращает число, равное количеству символов в строке;

Left(Строка, Число) - возвращает подстроку, состоящую из указанного **Числа** первых символов в строке;

Right(Строка, Число) - возвращает подстроку, состоящую из указанного **Числа** последних символов в строке;

Mid(Строка, Позиция, Длина) - возвращает подстроку, состоящую из указанного числа в **Длине** и с началом, указанным в числе **Позиция**;

Space(Число) – возвращает заданное число пробелов;

InStr(Старт, Строка1, Строка2) – возвращает номер позиции первого вхождения Строки2 в Строку1 или 0, если не найдено. Старт – числовое выражение, задающее позицию, с которой начинается каждый поиск. Если этот параметр опущен, то поиск начинается с первого символа строки..

### Функции даты и времени

Date - возвращает значение текущей системной даты, которое можно присвоить переменным типа Date. Значение даты представляется в виде чисел #Месяц/Число/Год#, разделенных знаком «/». Разностью значений переменных типа Date является число дней между датами.

Time – возвращает текущее время по часам компьютера.

Now – возвращает значение, содержащее текущую системную дату и текущее время по часам компьютера.

Day(Date) – возвращает день, соответствующий заданной в аргументе дате.

Month(Date) – возвращает месяц, соответствующий заданной в аргументе дате

Year(Date) – возвращает ujl, соответствующий заданной в аргументе дате/

### **Функции преобразования типов данных**

Эти функции реализуют преобразование данных из одного типа в другой. Наиболее часто используются следующие.

Val (Строка) - преобразует строковое значение в числовое, аргументом которой является строка, а значением — число.

Format(Число\_или\_Переменная,[Format) – возвращает выражение в соответствии с инструкциями, включенными в выражение формата.

Параметр Format, вводимый пользователем для числовых типов данных, состоит из символов, описанных ниже.

Символ	Описание
0	В этой позиции печатается цифра разряда числа. Замыкающий или ведущий ноль печатаются.
#	В этой позиции печатается цифра разряда числа. Замыкающий или ведущий ноль не печатаются.
.	Десятичная точка.
,	Разделитель тысяч.
- + \$ ()	Эти символы печатаются также, как они показаны в формате.

**Примеры.** Преобразование чисел функцией Format с использованием перечисленных выше символов:

Вид функции Format	Возвращаемый результат
Format(8315.4, "00000.00")	08315.40
Format(8315.4, "#####.##")	8315.4
Format(8315.4, "##,##0.00")	8,315.40
Format(8315.4, "\$0000.00")	\$8315.40

## **5. Процедуры и функции**

В случае многократного выполнения одной и той же последовательности операций целесообразно представить эту последовательность в виде **процедуры** или **функции**.

При вызове процедуры или функции в качестве параметров указываются имена тех переменных, значения которых необходимо передать в нее в данный момент и которые обрабатываются операторами вызываемой процедуры или функции.

**Общая процедура** – это подпрограмма, которая обычно используется, когда нужно в разных частях программы повторить один и тот же блок операторов. Она начинается оператором Sub и заканчивается оператором End, между ними помещается код процедуры:

```
Sub ИмяПроцедуры([параметры])  
...(операторы)  
End Sub
```

Вызов процедуры в программе осуществляется посредством указания имени и параметров:

```
Call ИмяПроцедуры[(параметры)]
```

Выполнение общих процедур не связывается с какими-либо событиями, они вызываются на выполнение с помощью оператора Call.

Общая процедура представляет собой подпрограмму, которая начинает выполняться после ее вызова из другой процедуры.

Список входных параметров - это набор переменных, значения которых должно быть установлено до начала выполнения процедуры.

Список выходных параметров - это набор переменных, значения которых должно быть установлено после окончания выполнения процедуры.

**Пример.** Процедура SUM, суммирующая элементы заданного массива, и обращение к ней при вычислении среднего арифметического.

```
Sub SUM(x() As Single, k, S)      'Начало описания процедуры SUM  
For i = 1 To k  
S = S + x(i)  
Next i  
End Sub                          'Конец описания процедуры SUM  
Private Sub Command1_Click()  
Dim A(3) As Single  
A(1) = 3  
A(2) = 5  
A(3) = 7  
Call SUM(A(), 3, S)              'Обращение к процедуре SUM  
S = S / 3  
Print "SR="; S  
End Sub
```

В данной программе использовано обращение в процедуре к фактическому одномерному массиву A() с пустыми скобками.

**Локальные и глобальные процедуры.** Кроме событийных и общих процедур в программных модулях могут присутствовать процедуры, которые нельзя вызвать из других модулей и процедуры, которые предполагают

подобную возможность. Первые процедуры называются локальными, вторые – глобальными.

**Локальная процедура** доступна только внутри данного программного модуля и не может быть вызвана из другого модуля. В процедурах используется ключевое слово **Private**, которое указывает, что данные процедуры являются локальными.

Синтаксис локальной событийной процедуры следующий:

```
Private Sub Объект_Событие()
```

```
.....
```

```
End Sub
```

**Глобальные процедуры** доступны из всех программных модулей. Они предваряются ключевым словом **Public**. Однако его наличие не является обязательным. По умолчанию, если перед ключевым словом Sub ключевые слова отсутствуют, процедура является глобальной.

**Функция** в отличие от процедуры возвращает в программу значение, полученное путем выполнения оператора присвоения вида:

```
Имя_функции=значение.
```

Функция, как и любая переменная, имеет определенный тип. При этом значение может быть указано внутри функции явно или получено в результате преобразований (вычислений или вызова другой функции).

```
Function ИмяФункции(параметры) [As <тип>]
```

```
...(операторы)
```

```
Имя_функции=значение(выражение)
```

```
End Function
```

Функция вызывается указанием ее имени и параметров в правой части какого-либо оператора или в сложном выражении (т.е. там, где нужно получить ее значение):

```
x = Имя_Функции(параметры)
```

**Пример.** Функция, определяющая факториал числа.

'Начало описания функции Fact:

```
Function Fact(n As Integer) As Integer 'Начало описания функции Fact
```

```
Dim i As Integer
```

```
Dim F As Integer
```

```
For i=1 To n
```

```
F=F*i
```

```
Next i
```

```
Fact=F
```

```
End Function 'Конец описания функции Fact
```

## 6. Реализация основных алгоритмических структур

**Линейные алгоритмы** (рис.10) реализуются на языке *Visual Basic* несколькими командами (операторами), которые должны быть выполнены последовательно одна за другой.

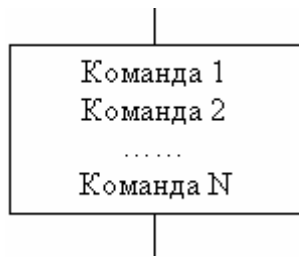


Рис.10. Линейный алгоритм

### Ветвящиеся алгоритмы

Алгоритмическая структура ветвления (рис.11) на языке *Visual Basic* программируется с помощью оператора условного перехода.

Ключевыми словами оператора являются *If* (если), *Then*(то) и *Else*(иначе). После слова *If* должно быть размещено условие, после слова *Then* – последовательность команд (Действие 1), которая должна выполняться, если условие истинно, после слова *Else* – последовательность команд (Действие 2), которая должна выполняться, если условие ложно.

Оператор условного перехода может быть записан в многострочной форме или в однострочной форме.

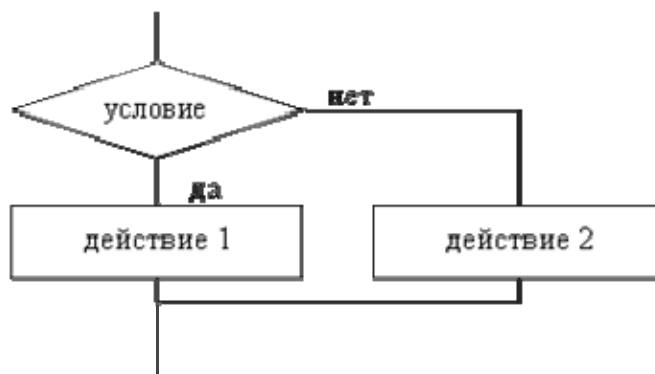


Рис. 11. Ветвящийся алгоритм

Синтаксис многострочной формы условного перехода:

```
If Условие1 Then
[блок операторов1]
[Else If Условие2 Then
[блок операторов2]]
...
[Else
[блок операторов n]]
End If
```

Оператор *If* «закрывается» ключевыми словами *End If*.

В однострочной форме условный оператор записывается в одну строчку, при этом ключевые слова *End If* не употребляются.

Синтаксис однострочного оператора *If...Then* и *If...Then...Else*:



If Условие Then Оператор  
If Условие Then Оператор1 [Else Оператор2]

Действие этого оператора состоит в следующем. Вначале проверяется условие. Если оно удовлетворяется, то выполняется Оператор1, стоящий после слова **Then**, в противном случае – Оператор2, стоящий после слова **Else**. Условие задается выражением булевского типа, т.е. результат его вычисления принимает значение True или False.

Блочный синтаксис оператора If...Then:  
If Условие Then  
Операторы  
End If

Оператор **множественного выбора** *Select... Case* удобно применять, если требуется проверка нескольких условий, при этом одно и то же выражение сравнивается с несколькими значениями:

```
Select Case Переменная
  Case Значение1
    Последовательность операторов1
  ...
  Case Значение(N-1)
    Последовательность операторов(N-1)
  [Case Else
    Последовательность операторовN]
End Select
```

**Пример.** Выбор действий в зависимости от значений переменной *x*.

```
Select Case x
  Case s =5
    'Выполнить некоторые действия
  Case s >=20
    'Выполнить другие действия
  Case Else
    'Действия, которые следует выполнить, когда
    ни одно из предыдущих условий не является верным
End Select
```

### **Циклические алгоритмы**

Алгоритмические структуры циклов применяются в случае, если какие-либо операции требуется применять несколько раз, пока не выполнится некоторое условие. При этом многократная последовательность операций называется телом цикла.

Циклические алгоритмические структуры делятся на:

- **циклы со счетчиком** (или арифметические), в которых тело цикла выполняется заранее известное определенное количество раз (рис. 12);

• **итерационные** (или циклы по условию), в которых тело цикла выполняется до тех пор, пока выполняется (или не выполняется) некоторое условие, то есть, когда заранее неизвестно количество повторений.

На языке Visual Basic цикл реализуется с помощью специальных инструкций.

**Цикл со счетчиком.** Когда заранее известно, какое число повторений тела цикла необходимо выполнить, можно воспользоваться циклической инструкцией (оператором цикла со счетчиком) For ... Next.

Синтаксис данного оператора следующий: строка, начинающаяся с ключевого слова For, является заголовком цикла, а строка с ключевым словом Next — концом цикла, между ними располагаются операторы, являющиеся телом цикла:

```
For Счетчик = НачЗнач To КонЗнач [Step шаг]
Тело цикла
Next [Счетчик]
```

В начале выполнения цикла значение переменной **Счетчик** устанавливается равным **НачЗнач**. При каждом выполнении цикла переменная **Счетчик** увеличивается на величину шага. Если она достигает величины **КонЗнач**, то цикл завершается и выполняются следующие за ним операторы.

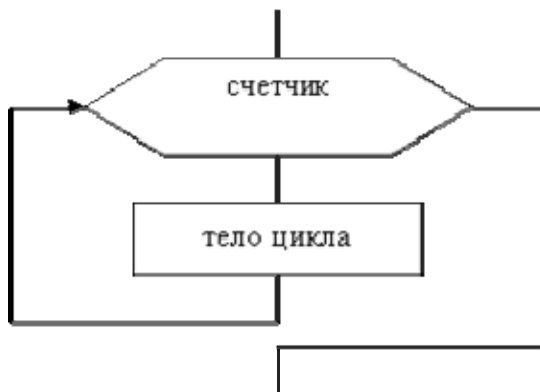


Рис. 12. Цикл со счетчиком

### **Итерационный цикл.**

Такой цикл реализуется с помощью инструкции Do ...Loop или While...Wend.

Существуют циклы с предусловием (рис. 13) и циклы с постусловием (рис. 14). Условие выхода из цикла можно поставить в начале и в конце цикла.

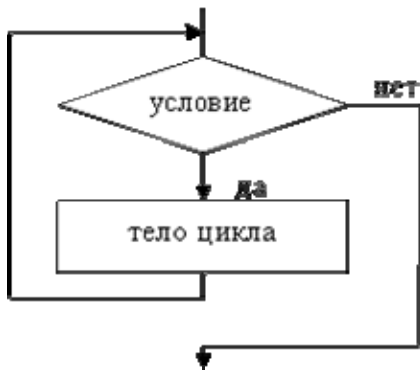


Рис. 13. Цикл с предусловием

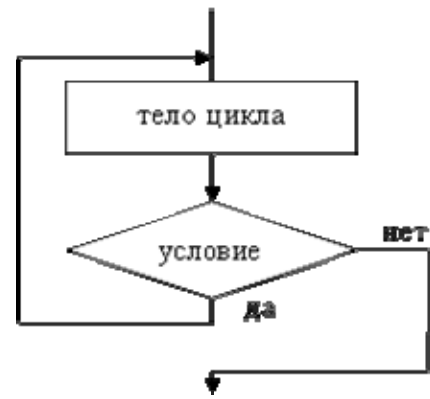


Рис.14. Цикл с постусловием

Проверка условия выхода из цикла проводится с помощью ключевых слов *While* или *Until*. Эти слова придают одному и тому же условию противоположный смысл. Ключевое слово *While* обеспечивает выполнение цикла до тех пор, пока выполняется условие, т.е. пока условие имеет значение *True* (истина). Как только условие примет значение *False* (ложь), выполнение цикла закончится.

Ключевое слово *Until* обеспечивает выполнение цикла до тех пор, пока не выполняется условие, т.е. пока условие имеет значение *False* (ложь). Как только условие примет значение *True* (истина) выполнение цикла закончится.

#### **Разновидности циклических конструкций.**

1. Do While Условие  
тело цикла  
Loop
2. Do Until Условие  
тело цикла  
Loop
3. Do  
тело цикла  
Loop While Условие
4. Do  
тело цикла  
Loop Until Условие
5. While Условие  
тело цикла  
Wend

## II. Практическая часть

Цель лабораторных работ — получение навыков создания Windows-приложений в среде Visual Basic, предназначенных для проведения инженерных расчетов. Эта цель достигается решением задач создания пользовательского интерфейса средствами конструктора форм, а также разработки и отладки текстов программ средствами редактора программного кода.

### Порядок выполнения лабораторных работ

1. Изучить теоретический материал по каждой теме.
2. Составить алгоритм решения задачи.
3. Разработать интерфейс проекта с установкой значений свойств формы.
4. Сохранить проект.
5. Ввести элементы управления на форме, задать их свойства.
6. Ввести программные коды, модули, реализующие решение задач.
7. Оформить отчет и подготовить ответы на контрольные вопросы.

### Содержание отчета

Отчет должен содержать:

- название лабораторной работы;
- цель лабораторной работы;
- формулировку предлагаемого индивидуального задания;
- копию созданного в среде Visual Basic проекта.

### Лабораторная работа 1. ВВОД ВЫРАЖЕНИЙ

*Цель лабораторной работы:* ознакомиться с основными компонентами интегрированной среды разработки проектов *Visual Basic*, научиться составлять проекты в этой среде на примере вычисления выражений.

#### З а д а н и е

Вычислите значение выражения, приведенного в табл. 1. Номер варианта равен Вашему номеру в журнале учебной группы.

Таблица 1

№ варианта	Выражение	Значение
1	2	3
1	$1,2 \cdot \operatorname{tg}(3 \cdot x) + \ln(2 \cdot x + 5,5) - 4,7 + \frac{\sqrt{x} + 1}{x \cdot \sqrt{x} + x + \sqrt{x}}$	2,21

2	$3,7 \cdot \cos^3(2 \cdot x - 7,1) + e^x - 14,3 \cdot  x  + \frac{\sqrt[3]{x}}{3 \cdot \sqrt[3]{x^5} - \sqrt[3]{x^2}}$	3,82
3	$6,5 \cdot \sin^2\left(\frac{4}{\ln x  + 1,1}\right) - 8 \cdot x^2 + \frac{\sqrt{x^2 + x} \cdot \sqrt{x^2 - 1}}{x}$	-2,36
4	$2 \cdot e^x + 3,5 \cdot \sin^2\left(x + \frac{x}{5-x}\right) +  x  + \frac{1 + \sqrt{x}}{\sqrt{x+1}}$	3,14
5	$3 \cdot x^2 - 6 \cdot \ln(x^2 + 2 \cdot x + 1) + 7 \cdot e^x + 4\sqrt[4]{\frac{2 \cdot x}{1 + \sqrt{x}}}$	5,73
6	$8,4 \cdot x^3 + \operatorname{arctg}\left(\frac{x}{x^2 + 1}\right) + \ln(3 \cdot x) - \frac{\sqrt[4]{x^2 - x}}{x^3 + x}$	2,92
7	$7,51 \cdot x^2 + 1,2 \cdot \operatorname{tg}^4\left(x + \frac{x}{x^3 + 3}\right) - 3,5 + \frac{1}{\sqrt{x^3} - \sqrt{x+1}}$	5,37
8	$8,5 \cdot \ln(x^4 + 1) + \frac{\cos^2(x + 2)^2}{2,5} + 2^x + \frac{\sqrt{2 - x^2}}{x^3 + 2 \cdot x}$	1,21
9	$6,75 \cdot e^{(x-1)} - 2 \cdot \cos^3\left( x  + 2 \cdot x^2 + \frac{x}{7}\right) + \frac{x - \sqrt{x+1}}{\sqrt[3]{3 \cdot \sqrt{2 \cdot x + 1} - x}}$	2,27
0	$3,7 \cdot x^2 + 7,1 \cdot \sin^2(x^4 + x^2 - 1) - \ln( x ) + \frac{x - \sqrt{x}}{\sqrt[3]{x + \frac{8}{x}}}$	2,51
11	$7,1 \cdot x^3 + 13,4 \cdot \operatorname{tg}^2(x - 1) - \frac{ x }{5} + \frac{\sqrt{x \cdot (1 + \sqrt[3]{x})}}{1 - x \cdot \sqrt{2 - x}}$	1,85
12	$18,5 \cdot x^2 + 2 \cdot \ln\left(\frac{1+x}{\sqrt{x}}\right) - 4,7 \cdot \cos^2(x) + \frac{\sqrt[4]{x + 2 \cdot \sqrt{x}}}{1 - x^3}$	2,65
13	$2,65 \cdot x^2 - 3,7 \cdot e^{(x-1)} + 2,3 \cdot \cos^2(2 \cdot x - 3) + \sqrt{\frac{5 + 2 \cdot \sqrt{x}}{\sqrt{x} - x^3}}$	0,74
14	$3,7 \cdot \sin^3(2 \cdot x - 1,5) + 5 \cdot \operatorname{tg}(2 \cdot x) - 2 \cdot \operatorname{arctg}(x - 1) + \frac{\sqrt[5]{x^2 + 2}}{1 + \sqrt{x+1}}$	1,64
15	$8,1 \cdot \operatorname{arctg}^2(3 \cdot x - 1) + 14 \cdot \ln(x + 1) + 3 \cdot e^x + \frac{\sqrt[3]{x^3 - x^2}}{1 + \sqrt{x+1}}$	2,32
16	$12,4 \cdot \ln\left(\frac{x+3}{4,5}\right) + 7 \cdot \operatorname{tg}^2(e^x - 3) + \frac{x \cdot \sqrt{x-2}}{x^2 + 2 \cdot x}$	2,85

17	$6.85 \cdot \sin^4(x - 1.5) + 2 \cdot \ln\left(3 \cdot x + \frac{4}{x}\right) + \frac{\sqrt[5]{x^4 - 3}}{3 - \sqrt{x + 3}}$	9,12
18	$8,7 \cdot \operatorname{tg}^2(3,5 \cdot x + 4,5) + \frac{ 1 - x^3 }{\cos(2 \cdot x - 1)} + e^x + \sqrt[3]{\frac{x^2 + 3}{1 + x\sqrt{x}}}$	3,17
19	$9,8 \cdot x^2 - e^{(3 \cdot x + 4)} + \operatorname{arctg}(x^3) + \sin(2 \cdot x) + \frac{\sqrt{x + 0,62} \cdot \sqrt{x}}{2 + \sqrt[3]{x}}$	0,24
20	$8,46 \cdot x^2 - 3 \cdot \operatorname{tg}(2 \cdot x - 5) + 7,2 \cdot e^{(x-1)} + \ln(x) + \sqrt[3]{\frac{x + \sqrt{x}}{1 - x^3}}$	0,95
21	$2,1 \cdot \operatorname{ctg}^2(2 \cdot x + 21) + \ln\left(\frac{1 + x^3}{1 + x^2}\right) - 7,4 + \frac{1 + \sqrt{x}}{1 + \sqrt[3]{x}}$	5,21
22	$7.2 \cdot \cos^2(2 \cdot x - 5.7) + e^{(x+3)} - 7 \cdot \sqrt{x} + \frac{\sqrt[3]{2 - \sqrt{x+1}}}{\sqrt{x} + 3.3}$	2,15
23	$5,2 \cdot \sin^2(4 + \ln(x)) - 8 \cdot \operatorname{tg}(x + x^2) + \frac{\sqrt[3]{x^3 - x - 4}}{2 \cdot x + 3 \cdot \sqrt{x}}$	2,13
24	$4,8 \cdot x^3 + \operatorname{arctg}\left(\frac{x+1}{x}\right) + \ln(2 \cdot x) + \frac{\sqrt[3]{x+2} \cdot x^3}{2 + \sqrt{x+2}}$	3,52
25	$5,2 \cdot x^3 + 2,1 \cdot \operatorname{tg}^3\left(x - \frac{x}{x^2 + 2}\right) - 6,8 + \frac{1 + \sqrt[3]{1 + x^4}}{1 - \sqrt{x+3}}$	2,81
26	$\ln\left(\frac{1 + x \cdot \sqrt{2} + x}{1 + x\sqrt{2} + x^2}\right) + 2 \cdot \operatorname{arctg}\left(\frac{x \cdot \sqrt{2}}{1 - x}\right)$	1,54
27	$\frac{1}{2} \cdot \operatorname{tg}(x) + \ln(\cos(x)) + \sqrt[3]{\frac{x \cdot (x+1)}{x-1}}$	1,35
28	$\operatorname{arctg}\left(\frac{1 - \cos(x)}{1 + \cos(x)}\right) - \frac{x \cdot (x+1)}{1 - x}$	0,87
29	$\frac{1}{3} \cdot \ln\left(\frac{1 + x}{\sqrt{x + x^2} + 1}\right) + \frac{1}{\sqrt{3}} \cdot \operatorname{arctg}\left(\frac{2 \cdot x - 1}{\sqrt{3}}\right)$	2,38
30	$\arcsin(\sqrt{\sin(x)}) + \operatorname{arctg}\left(\frac{4 \cdot \sin(x)}{5 \cdot \cos(x)}\right)$	2,75

## Пример выполнения задания

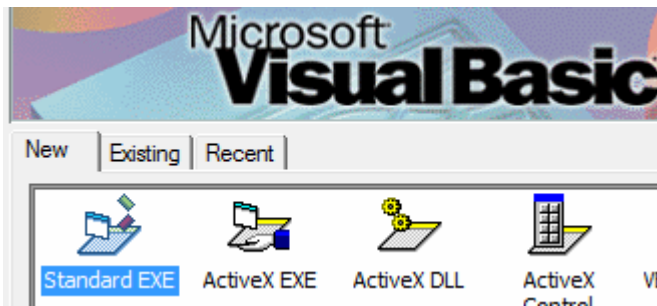
### 1. Постановка задачи.

Вычислить значение выражения  $Y = \frac{x + \sqrt{x^2 + 2}}{\sin(x^2) + \ln^2 \left| x + \sqrt{1 + x^3} \right|}$  при  $x = 5,5$ .

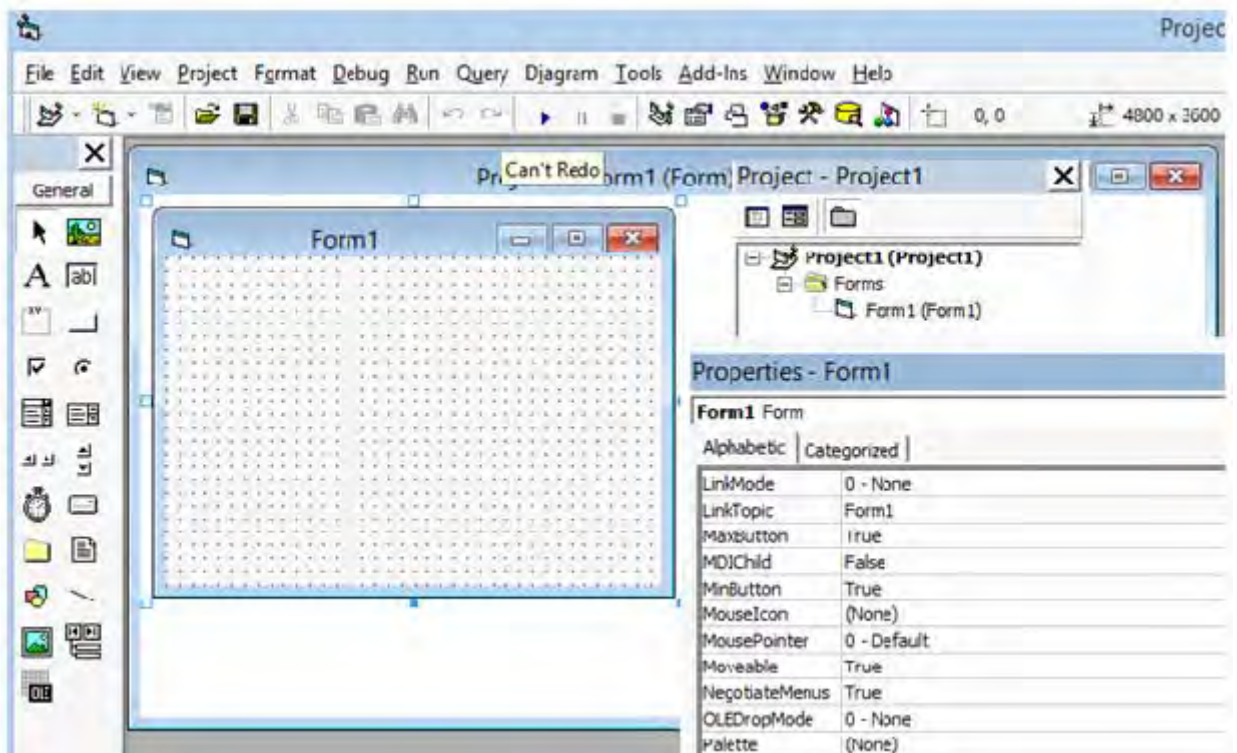
**2. Разработка интерфейса проекта** (визуальное проектирование) состоит из следующих шагов.

2.1. Запустим систему программирования *Visual Basic* командой:  
*Пуск/Программы/MS Visual Basic*

2.2. . Возникнет окно *New Project*, в котором во вкладке *New* выберем шаблон *Standart.EXE*.



2.3. Появится окно интегрированной среды *Visual Basic*.





Познакомимся с содержимым появившегося окна и проследим, чтобы в нем присутствовали окно панели управления элементами, окно проводника, окно свойств. При отсутствии какого-нибудь из этих окон обратимся к соответствующей команде главного меню *View* (Вид) и вызовем его.

2.4. В этом окне единственный объект – форма *Form 1* (пока пустая).

Присвоим свойствам нашей формы значения. Для этого в окне свойств во вкладке *Alphabetic* в столбце слева найдем имя свойства, а в столбце справа зададим его значение в соответствии с таблицей 2.

Таблица 2

Имя свойства	Значение свойства	Пояснения
Name	Expr	Имя формы используется для обращения к ней из программы
Caption	Вычисления	Заголовок формы. Размещается в строке заголовка формы
Left	2000	Расстояние левого верхнего угла формы от левой границы окна компьютера
Top	2000	Расстояние левого верхнего угла формы от верхней границы окна компьютера
Width	5000	Ширина формы
Height	3700	Высота формы
MaxButton	False	Запрет максимизации формы
MinButton	False	Запрет минимизации формы
Font	Ms San Serif	Размер шрифта 14

Остальные свойства оставим заданными по умолчанию системой *Visual Basic*.

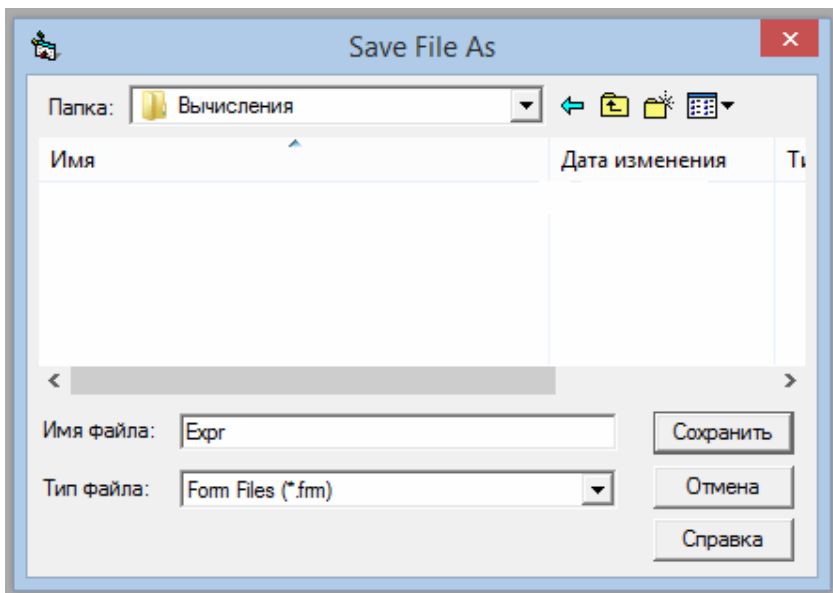
2.5. Сохраним наш проект. Для этого выполним следующие действия.

1. Свернем все окна на панель задач.
2. Создадим папку под именем *Вычисления* для хранения проекта любым способом, например, используя проводник.

Эту папку можно было создать перед началом работы над проектом.

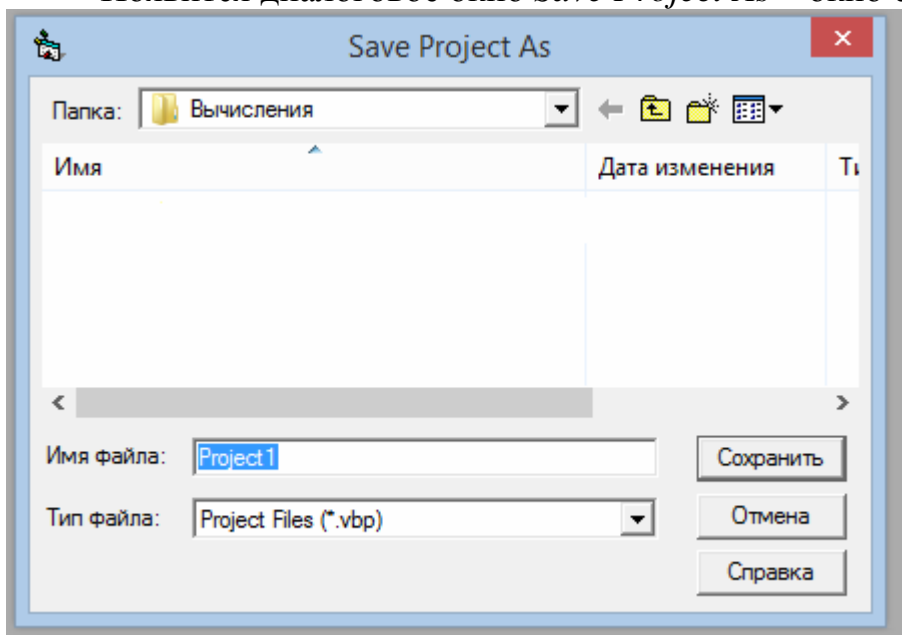
3. Развернем все окна.
4. Введем команду меню *File / Save Project As*.

Появится диалоговое окно *Save File As* – окно сохранения формы:



В строке *Папка* укажем папку *Вычисления*. В окне *Имя файла* по умолчанию введено имя файла формы *Expr*. Согласимся с таким именем и нажмем кнопку *Сохранить*.

Появится диалоговое окно *Save Project As* – окно сохранения проекта:



В выпадающем списке *Папка* введено имя папки для проекта *Вычисления*. В поле *Имя файла* установлено по умолчанию имя файла проекта *Project1*. Согласимся с этим именем и нажмем кнопку *Сохранить*.

5. Убедимся, что в окне *Project Explorer* отобразились изменения имен (рис.1).

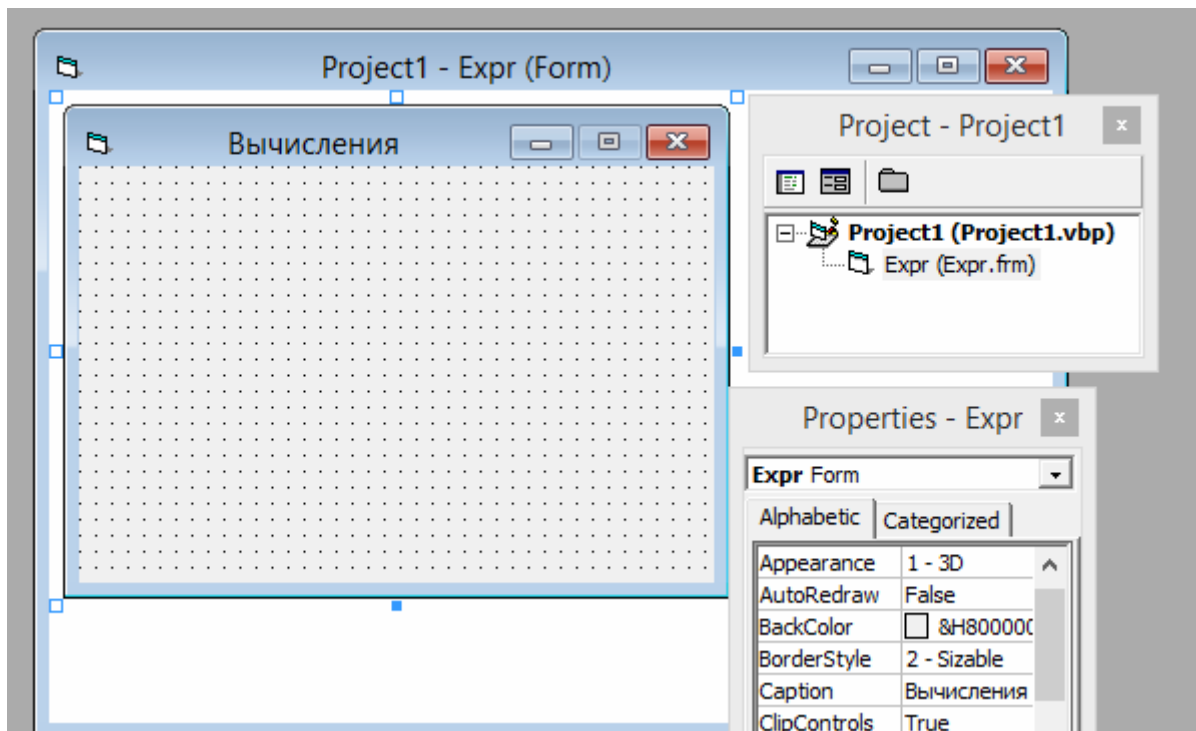


Рис.1. Вид основного окна после сохранения проекта

2.6. Выйдем из системы *Visual Basic*, нажав на кнопку «Закреть» либо выполнив команду *File / Exit*. Тогда появится окно с вопросом: *Save Changes to the following Files?* (Сохранить изменения следующих файлов?). Щелкните по кнопке *Yes*.

2.7. Убедимся, что в папке *Вычисления* находятся два файла: файл формы *Expr.frm* и файл проекта *Project1.vbp*. Для этого воспользуемся например, проводником и войдем в папку *Вычисления* (рис. 2).

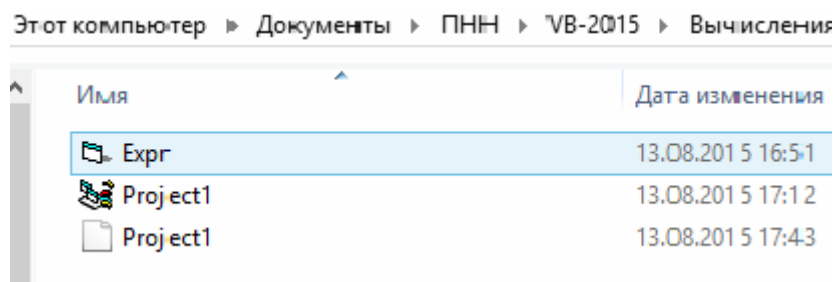


Рис. 2. Содержимое папки *Вычисления*

2.8. Откроем свой проект. Для этого в папке *Вычисления* дважды щелкнем по значку с именем проекта:



2.9. Введем элементы управления на поверхность формы.

Окончательный вид формы в результате проектирования показан на рис.3.

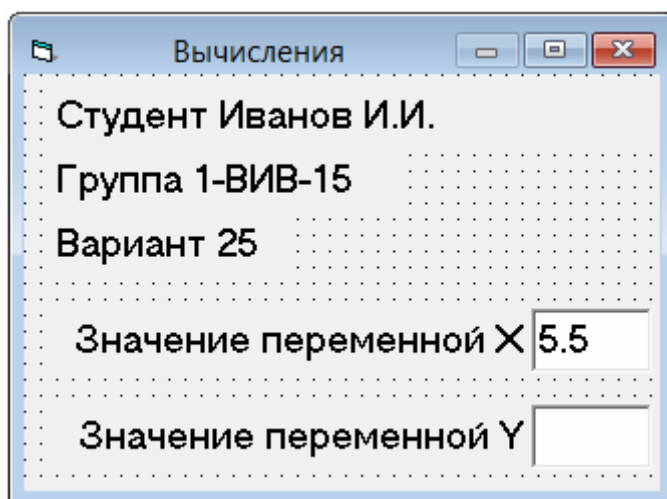




Рис 3. Вид формы в результате проектирования

Следовательно, на поверхности формы, перетаскиванием с помощью мыши, необходимо расположить следующие элементы управления (сверху вниз) из *Панели управления*: пять надписей (*Label1, Label2, Label3, Label4, Label5*- значок ), содержащих фамилию студента, номер группы, номер варианта, значение переменной *X*, значение переменной *Y*,

и правее последних двух надписей два текстовых поля (*TextBox1, TextBox2*, – значок ) для ввода числового значения *X* и вывода результата *Y*.

2.10 Зададим следующие свойства элементам управления в соответствии с таблицей 3.

Таблица 3

Имя свойств	Значения свойств						
Name	Label1	Label2	Label3	Label4	Label5	TextBox1	TextBox2
Caption	Смотри рис. 2					Такого свойства нет	
Left	240					3800	
Top	120	600	1080	1800	2520	1760	2480
Height	375					480	
Width	4500	2535	1700	3500		900	
Text	Такого свойства у надписи нет					Пусто	Пусто
Font	Имя шрифта MS Sans Serif, размер шрифта 14						

Для надписей *Label4* и *Label5* свойству *Alignment* (Выравнивание) выбрать значение равное 1 - *Right Justify* (По правому краю).

### 3. Программирование в коде

После того, как форма примет вид, показанный на рис. 3, создадим программный код для вычисления выражения. Для этого выполним следующие действия.

3.1. Введем команду меню *View / Code*. Появится окно редактора кода. Окно почти пустое. Наверху окна имеется два выпадающих списка. В левом списке имеется надпись *General*, в правом — *Declarations*. Раскроем левый выпадающий список редактора кода.

Список содержит все объекты формы, их 8. Первый объект — это сама форма (она обозначена словом *Form*) и еще 5 надписей (*Label*) и 2 текстовых поля (*Text*). Нам нужно ввести программу, которая запускалась бы в работу после щелчка левой кнопкой мыши по текстовому полю *Text1* (где вводится переменная *X*).

3.2. Выберем в списке *Text1*. Левый список закрывается, в правом списке появляется слово *Change* (изменить), а в поле редактора кода — оператор заголовка процедуры *Private Sub Text1\_Change()* и оператор конца процедуры *End Sub*. Эта заготовка нам не нужна и ее удаляем.

3.3. Раскроем правый список, содержащий события. Щелчком мыши выберем событие *Click*. В поле редактора кода появятся заголовок процедуры *Private Sub Text1\_Click()* и оператор конца процедуры *End Sub*. Между этими операторами вводим остальные операторы (тело) процедуры, приведенные ниже.

### Option Explicit

---

```
Private Sub Text1_Click()  
Dim X As Single  
Dim Y As Single  
X = Val(Text1.Text)  
Y = (X + Sqr(X ^ 2 + 2)) / (Sin(X ^ 2) _  
+ Log(Abs(X + Sqr(1 + X ^ 3))) ^ 2)  
Text2.Text = Y  
End Sub
```

*Комментарии к программному коду.*

1) В строке  $Y = \dots$  студенты вводят из табл. 1, выражение в соответствии со своим вариантом.

2) Структура заголовка процедуры обработки события:

```
Private Sub Text1_Click().
```

Он состоит из двух частей, соединенных нижним тире. Левая часть — это имя объекта *Text1* (текстовое поле для ввода переменной *X*), а правая часть — имя события *Click* (щелчок левой кнопкой мыши), произошедшего с этим объектом.

3) Рассмотрим операторы тела процедуры.

Первыми располагаются операторы *Dim*, предназначенные для объявления переменных *X* и *Y* с типом данных *Single*.

После печати служебного слова *As* и нажатия клавиши пробел появляется список. Далее следует набрать буквы *Si*. Список прокрутится так, что нужное слово *Single* окажется наверху списка подсвеченным. Далее продолжать набор не обязательно. Следует нажать клавишу [Tab], после чего будет введена остальная часть слова. Это свойство редактора кода целесообразно использовать как можно чаще во избежание опечаток.

Далее следует оператор присваивания

```
X = Val(Text1.Text).
```

Это означает, что мы из объекта *Text1* извлекаем значение свойства *Text*. Это значение имеет текстовый формат и является аргументом функции *Val*, которая переводит введенное пользователем значение из текстового формата в числовой. Разделителем целой части числа от дробной функция *Val* воспринимает только точку. Затем числовое значение, возвращаемое функцией *Val*, присваивается переменной *X*.

Далее следует оператор присваивания

```
Y = (X + Sqr(X ^ 2 + 2)) / (Sin(X ^ 2) _  
+ Log(Abs(X + Sqr(1 + X ^ 3))) ^ 2).
```

Здесь осуществляется вычисление значения выражения, и этот результат присваивается переменной *Y*.

Строка программы, в которой располагается вычисляемое выражение, получается очень длинной, с ней неудобно работать. Поэтому сделан перенос, то есть одна строка разбита на две так, чтобы Visual Basic воспринимал их как одно выражение. **Для переноса выражения надо в нужном месте ввести пробел, а за ним нижнюю черточку «\_» и нажать на клавишу «Enter».**

Вычисляемое выражение состоит из переменных (в нашем случае это *X*), символов математических операций +, -, \*, /, ^ и встроенных функций. Описание встроенных функций находится в теоретической части пособия в соответствующем разделе

Далее следует оператор присваивания

```
Text2.Text = Y.
```

Здесь свойству *Text* объекта *Text2* текстового типа присваивается числовое значение *Y*. Преобразование значения числового типа в значение текстового типа Basic производит самостоятельно.

#### **4. Запуск проекта на выполнение.**

Для запуска проекта (приложения) и получения результата выполним следующие действия.

4.1. Введем команду *Run / Start*.

4.2. В соответствии с условием нашего примера введем число  $X = 5.5$  в текстовое поле *Text1* и щелкнем по нему мышью.

Студенты вводят значение  $X$  из табл. 1, соответствующее своему варианту.

4.3. В поле *Text2* появится результат.

*Примечание.* Число  $X$  можно ввести в строке свойства *Text* объекта *Text1*. Тогда это число не нужно будет вводить при многочисленных запусках при отладке (См. рис.3, где введено число 5.5 по условию этого примера). Этот прием следует применять и в дальнейшем.

### Контрольные вопросы

1. Из каких компонентов состоит система программирования *Visual Basic*? Их назначение.
2. Опишите технологию визуального проектирования.
3. Что такое объект?
4. Назначение элемента управления *Label* (Метка).
5. Назначение элемента управления *TextBox* (текстовое поле).
6. Что такое метод? Как метод обозначается в программном коде?
7. Что такое событие объекта? Как оно обозначается в программном коде?
8. Что такое процедура обработки события для данного объекта? Какие действия может совершать процедура обработки события.
9. Опишите вид окна системы программирования *Visual Basic*, назовите компоненты окна, как их вызвать или убрать.
10. Назначение окна *Project Explorer*. Что в нем отображается? Какие действия можно совершать в этом окне? Его использование для открытия форм и модулей
11. Назначение окна *ToolBox* (палитра объектов). Что в нем отображается и каков порядок вставки объектов на поверхность формы?
12. Окно *Properties* (свойств), назначение, правила использования.
13. Как изменить значение свойства объекта в период разработки проекта?
14. Как изменить значение свойства объекта в программе?
15. Назначение окна конструктора форм. Что в нем отображается? Как управлять содержимым этого окна?
16. Окно редактирования кода. Что может содержаться в этом окне? Назначение списка *Object* и списка *Procedure* и кнопок переключения режимов просмотра кода.
17. Как переключиться с окна конструктора форм на окно редактирования кода и обратно,
18. Информационные возможности редактора кода. Как их включить?
19. Что такое объект-контейнер и как он обозначается в программном коде?
20. Математические операции *Visual Basic*.
21. Математические встроенные функции.
22. Какими способами осуществляется ввод следующих выражений  $x^2, \sqrt{x}, \sqrt[3]{x}, |x|, e^x, \ln(x), \log(x), \log_a(x), \sin^2(x), \operatorname{tg}(x), \operatorname{ctg}(x), \operatorname{arctg}(x), \arcsin(x), \arccos(x)$  ?

23. Как ввести числа  $e$ ,  $\pi$  без потери точности?
24. Как провести перенос длинной строки, таким образом, чтобы Visual Basic продолжал бы воспринимать ее как одну строку программного кода?
25. Можно ли разместить в одной строке программного кода несколько коротких операторов? Если можно, то как их разделять друг от друга?
26. Как ввести строку заголовка и строку завершения следующих процедур обработки событий:
  - щелчка левой кнопкой мыши по форме?
  - щелчка правой кнопкой мыши по форме?
  - двойного щелчка левой клавиши мыши по форме?
  - щелчка левой кнопкой мыши по текстовому полю *TextBox*?
  - щелчка левой кнопкой мыши по командной кнопке *CommandButton*?
- Запишите на бумаге строку заголовка одной из процедур обработки событий, перечисленных в предыдущем пункте.
27. Какие способы ввода свойств в период разработки приложения вы знаете?
28. Как в период разработки приложения присвоить значение какому-либо свойству формы или элемента управления?
29. Как установить размер и название шрифта в текстовом окне *TextBox*?
30. Как в программном коде присвоить или получить значение свойства формы или элемента управления?
31. Что такое Твип и чему он равен?
32. Что означают следующие свойства *Name*, *Caption*, *Left*, *Top*, *Width*, *MaxButton*, *MinButton*, *Font*, *Alignment*.
33. Как сохранить проект?
34. Как открыть проект?
35. Как разместить (ввести) на поверхности формы элемент управления?
36. Назначение элементов управления, прототипы которых расположены на панели элементов управления при открытии нового проекта.
37. Что такое модуль? Какие виды модулей вы знаете? Как их ввести в проект?
38. Требования к идентификаторам.
39. Как объявить переменную?
40. Типы данных.
41. Напишите операторы объявления переменных числового типа, строк переменной и фиксированной длины, пользовательского типа, массива.
42. Область видимости переменной. Объясните значения ключевых слов *Private*, *Public* и *Static*, используемых при объявлении переменных.
43. Назначение функции *Val(...)*. Какой символ функция *Val(...)* воспринимает как разделитель дробной и целой частей числа?
44. Как ввести в программу информацию, введенную в текстовое окно *TextBox*?
45. Как вывести в текстовый блок *TextBox* информацию из программы?



## Лабораторная работа 2. УСЛОВНЫЕ ОПЕРАТОРЫ

*Цель работы:* научиться в среде *Visual Basic* разрабатывать проекты, реализующие алгоритмы ветвления.

### З а д а н и е

1. Составить проект расчета выражения  $Y$  в зависимости от величины  $x$  в соответствии с одним из приведенных ниже вариантов.

2. Значения  $Y$  вычислить для таких шести значений  $z$ , чтобы три значения аргумента  $x$  соответствовали расчету  $Y$  по верхней формуле, а три — по нижней.

Результаты оформить в виде таблицы табуляции (табл. 6).

Эта таблица должна состоять из трех столбцов и шести строк. В каждой строке в левом столбце должна располагаться возрастающая сверху вниз последовательность из шести значений  $z$  с постоянным шагом табуляции равным 0,2. Шаг табуляции — это постоянная разность между смежными значениями  $z$ . Значения  $z$  студент вычисляет «вручную», используя в качестве калькулятора окно *Immediate*. В среднем и правом столбцах — соответствующие значения  $X$  и  $Y$ .

Эта таблица должна быть приведена в отчете по лабораторной работе.

#### Вариант 1

$$Y = \begin{cases} 0.064 \cdot \sqrt{x^3 + 1.5} & \text{если } x > 1,4 \\ \sin(2 \cdot x) & \text{если } x \leq 1,4 \end{cases}$$

где  $x = \operatorname{tg}(z)$

#### Вариант 3

$$Y = \begin{cases} x^2 + e^{-2 \cdot x} & \text{если } x > 1,1 \\ \sin^2(x) + \operatorname{tg}(x) & \text{если } x \leq 1,1 \end{cases}$$

где  $x = 3z$

#### Вариант 5

$$Y = \begin{cases} \frac{5 \cdot x^2 \cdot e^{-x}}{\ln(x^2)} & \text{если } x < 1 \\ \frac{x \cdot (1 + x^2)}{x \cdot (1 + x^2)} & \text{если } x \geq 1 \end{cases}$$

где  $x = 2z$

#### Вариант 7

$$Y = \begin{cases} 2 \cdot \operatorname{arctg}(\sin(x)) & \text{если } x \leq 1,1 \\ \frac{x^3 \cdot 3^{-x}}{(x+1)^2} & \text{если } x > 1,1 \end{cases}$$

где  $x = \sqrt[3]{z}$

#### Вариант 2

$$Y = \begin{cases} 100 \cdot \ln(x^2 + 1) & \text{если } x > 0 \\ 50 \cdot \sin(x) & \text{если } x \leq 0 \end{cases}$$

где  $x = 0.2z$

#### Вариант 4

$$Y = \begin{cases} \frac{5}{x^2 + 4} & \text{если } x \leq 3 \\ \sin(x) & \text{если } x > 3 \end{cases}$$

где  $x = 5 \cos(z)$

#### Вариант 6

$$Y = \begin{cases} \frac{2^x - 1}{\ln(x) + 2} & \text{если } x \geq 2 \\ 2 \cdot \sin(2 \cdot x) + \ln^2(x) & \text{если } x < 2 \end{cases}$$

где  $x = 4 \ln(z)$

#### Вариант 8

$$Y = \begin{cases} \ln(1 + e^{2 \cdot x}) & \text{если } x > 1 \\ \operatorname{arctg} \frac{x}{\sqrt{1 + x^2}} & \text{если } x \leq 1 \end{cases}$$

где  $x = z^2$

**Вариант 9**

$$Y = \begin{cases} 10 \cdot \sin(x) & \text{если } x \leq 0 \\ 3 \cdot \operatorname{tg}(x) & \text{если } x > 0 \end{cases}$$

где  $x = \operatorname{arctg}(z)$

**Вариант 10**

$$Y = \begin{cases} \frac{2 \cdot \sin(2 \cdot x)}{\sqrt[3]{2 + \cos^4(x)}} & \text{если } x < 1 \\ 10 \cdot \ln(x^2 + 0.2x) & \text{если } x \geq 1 \end{cases}$$

где  $x = \sqrt[3]{z^2}$

Следующие варианты с 11 по 30 можно получить, используя выражение для  $Y$  от одного варианта, а выражение для  $x$  от другого в соответствии с табл. 4.

Таблица 4

№ варианта	11	12	13	14	15	16	17	18	19	20
№ выражения для $Y$	1	2	3	4	5	6	7	8	9	10
№ выражения для $x$	10	1	2	3	7	5	2	7	5	9
№ варианта	21	22	23	24	25	26	27	28	29	30
№ выражения для $Y$	1	2	3	4	5	6	7	8	9	10
№ выражения для $x$	9	5	1	2	3	7	5	3	2	7

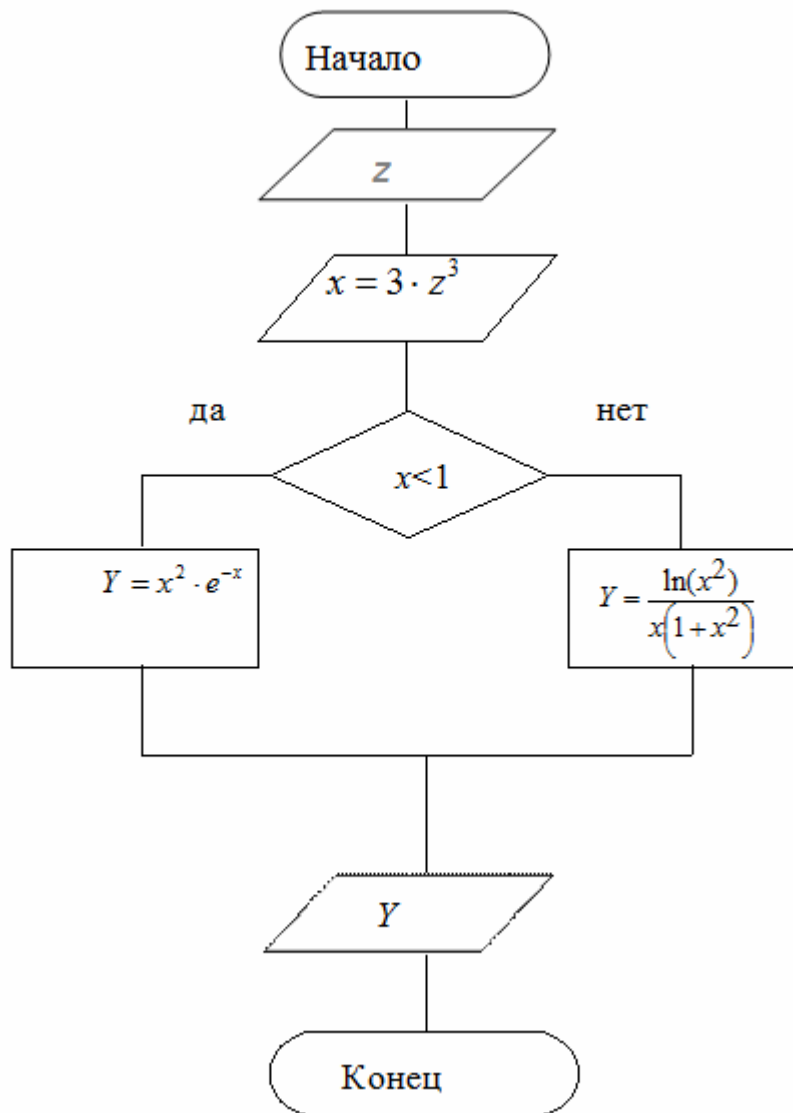
**Пример выполнения задания**

Пусть требуется вычислить значение

$$Y = \begin{cases} x^2 \cdot e^{-x} & \text{если } x < 1 \\ \frac{\ln(x^2)}{x(1+x^2)} & \text{если } x \geq 1, \end{cases}$$

где  $x = 3 \cdot z^3$

Блок-схема алгоритма ветвления имеет вид:



Для реализации первой части задания выполним следующие действия.

### 1. Создание новой папки.

Каждая лабораторная работа — это отдельный проект. Поэтому файлы каждой лабораторной работы должны храниться обязательно в отдельных папках. В рабочей папке (там, где папка с проектом *Вычисления*) создадим новую папку с именем *УслОпер*. Скопируем все файлы из папки *Вычисления* в папку *УслОпер*. Убедимся, что в новой папке проект работает точно так же, как и в старой.

### 2. Доработка интерфейса проекта, созданного в лабораторной работе №1.

1) Для формы (см. рис. 3) в окне свойств зададим свойству *Height* значение, равное 5720. Высота формы увеличится (опустится ее нижняя граница).

2) Ниже надписи *Label5* с помощью мыши введем надпись *Label6*, текстовый блок *Text3*.

3) Введем командную кнопку *Command1*.

4) В окне свойств присвоим этим объектам значения, приведенные в табл.5.

Таблица 5

Свойства	Label6	Text3	Command1
Left	240	3800	3120
Top	3240	3200	3960
Height	375	480	615
Width	3500	900	2775
Размер шрифта	14	14	14

5) Для надписи *Label6* свойству *Alignment* (Выравнивание) установим значение 1 - *Right Justify* (Выравнивание вправо).

6) Свойствам *Caption* надписей *Label4*, *Label5*, *Label6* и кнопке *Command1* присвоим соответственно: *Значение переменной Z*, *Значение переменной X*, *Значение переменной Y* и *Вычислить*.

В результате должна получиться форма, показанная на рис. 4.

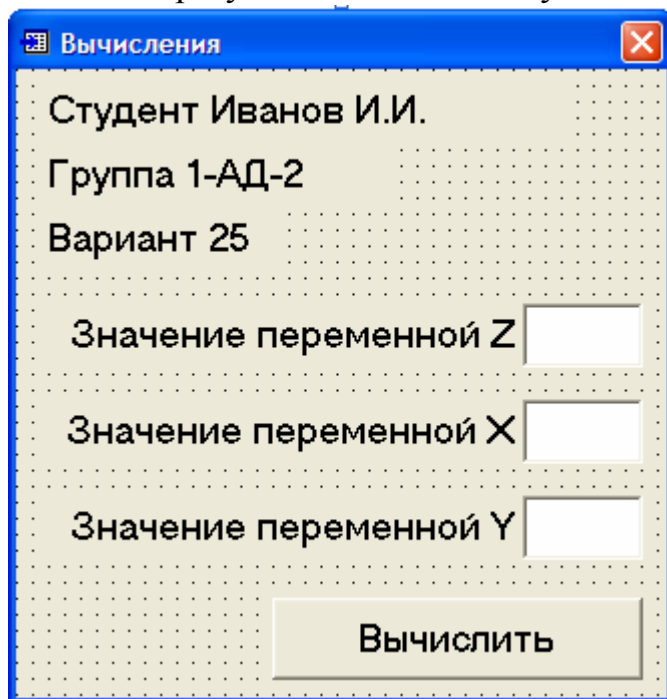
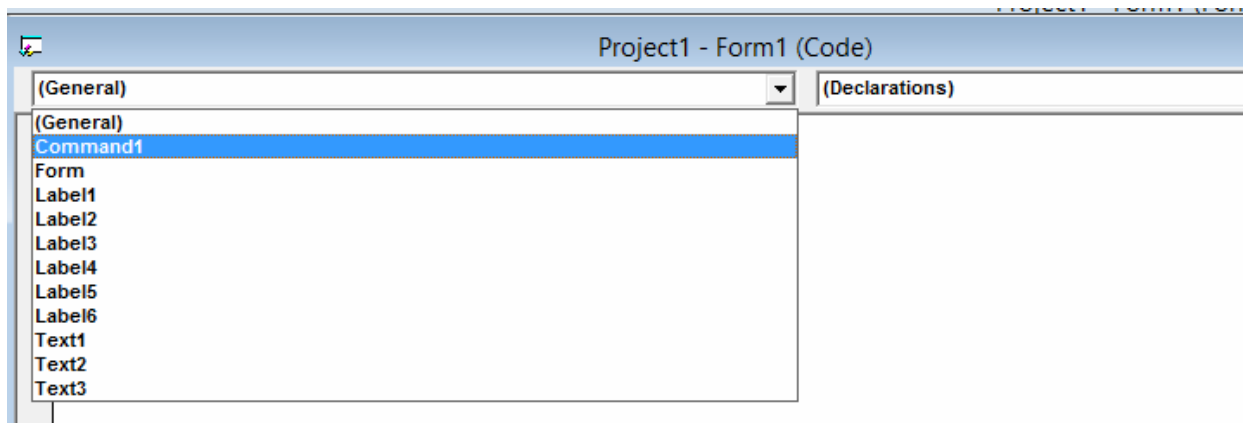


Рис.4

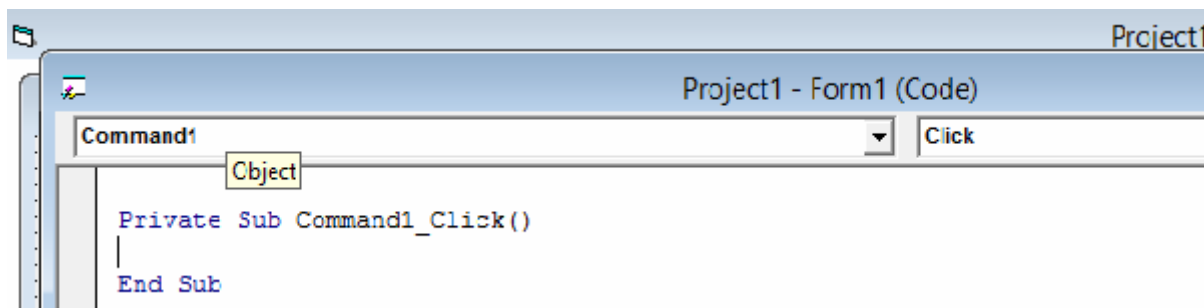
3.

### 3. Ввод программного кода.

- Вычисления должны проводиться с момента события нажатия кнопки. Выполним команду меню *View / Code*. Появится окно редактора кода.



- В этом окне откроем левый выпадающий список и выберем мышкой имя объекта управления *Command1*.
- Появятся операторы заголовка и конца процедуры обработки события щелчка левой кнопкой мыши по кнопке *Command1*.



- Все операторы первой процедуры скопируем (вставим) в тело новой процедуры.
- Старую процедуру удалим.
- Убедимся, что программа работает так же, но результат появляется после нажатия кнопки *Вычислить*.
- Доработаем программный код:
  - 1) Введем оператор Dim объявления переменной Z.
  - 2) Удалим строку, в которой осуществлялось вычисление выражения.
  - 3) Введем новые строки, в которых будет решаться новая задача.
  - 4) Вид полученной процедуры обработки события показан ниже.

```
Private Sub Command1_Click()
Dim X As Single
Dim Y As Single
Dim Z As Single
Z = Val(Text1.Text)
X = 3 * Z ^ 3
Text2.Text = Format(X, "0.000")
If X < 1 Then
```

```

Y = X ^ 2 * Exp(-X)
'Stop
Else
Y = Log(X ^ 2) / (X * (1 + X))
'Stop
End If
Text3.Text = Format(Y, "0.000")
End Sub

```

*Комментарии к программному коду*

В строке  $X = \dots$ , и двух строках  $Y = \dots$  студент вводит выражения, соответствующие своему варианту.

Строки со словом *Stop* закомментированы (они нам потребуются в дальнейшем при анализе полученных результатов).

#### 4. Запуск проекта на выполнение.

- 1) Для запуска проекта (приложения) введем команду *Run / Start*.
- 2) Появится окно формы (рис.4).
- 3) В нем введем любое число  $Z$  в текстовое поле *TextBox1*.
- 4) Щелкнем мышью по кнопке *Вычислить*.
- 5) Убедимся, что наш проект работает.

Для выполнения второй части задания выполним следующие действия.

**1. Поиск граничного значения  $Z_0$ , при котором  $x = 1$ .** Этот поиск в каждом варианте разный, так как значения аргумента  $x$  и связь между  $x$  и  $Z$  в заданиях индивидуальны.

В нашем случае, чтобы вычислить значения  $Y$  для шести значений  $Z$ , найдем сначала граничное значение  $Z_0$ , при котором  $x = 1$ . Тогда при  $Z < Z_0$  значения  $Y$  будут вычисляться по первой формуле задания  $Y = x^2 \cdot e^{-x}$ , в

противном случае – по второй  $Y = \frac{\ln(x^2)}{x(1+x^2)}$ .

Так как  $X$  и  $Z$  связаны по условию соотношением  $x = 3 \cdot z^3$  (в каждом варианте оно свое), то после подстановки в это равенство  $x = 1$  получим уравнение относительно  $Z_0$ :

$$1 = 3 \cdot Z_0^3.$$

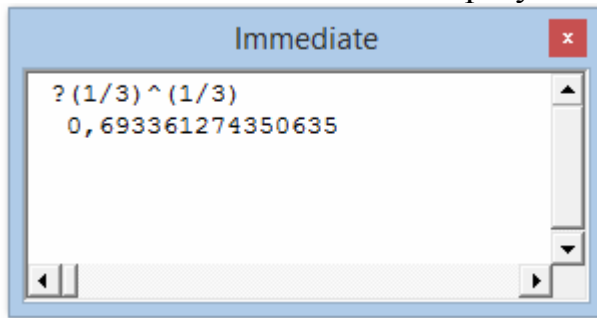
Решением этого уравнения будет следующее выражение для  $Z_0$ :

$$Z_0 = \left(\frac{1}{3}\right)^{\frac{1}{3}}.$$

Для вычисления этого выражения вызовем окно Immediate командой View/Immediate Window, щелкнем мышкой в этом окне, наберем на клавиатуре

$$(1/3)^{(1/3)}$$

и нажмем клавишу **[Enter]**. В следующей строке получим число 0.693361274350635. Ниже показан результат этих действий.



Таким образом, мы получили решение (корень) уравнения  $1 = 3 \cdot Z_0^3$  :  
 $Z_0 = 0.693361274350635$ .

## 2. Поиск шести значений $Z$ для решения задачи.

Нам необходимо выбрать шесть значений  $Z$  (три – левее и три – правее найденного  $Z_0$ ) с шагом равным 0.2.

Для этого сначала в числе 0.693361274350635 оставим только десятые, а остальные отбросим. Получим  $Z = 0.6 < 0.693361274350635$ .

Будем считать это число первым числом (началом) убывающей последовательности  $Z$  для соответствующего  $X < 1$ .

Остальные два значения  $Z$  для  $X < 1$  получим, дважды вычитая шаг 0.2 из полученного значения для  $Z$ . То есть  $Z = [0.6, 0.4, 0.2]$ .

Чтобы получить значения  $Z$  для  $X \geq 1$ , продолжим полученную последовательность, три раза прибавляя шаг 0.2 к значению  $Z = 0.6$ : 0.8, 1.0, 1.2.

Таким образом, получим единую последовательность

$$Z = [0.2, 0.4, 0.6, 0.8, 1.0, 1.2],$$

в которой значения 0.2, 0.4, 0.6 соответствуют  $X < 1$ , а значения 0.8, 1.0, 1.2 соответствуют  $X \geq 1$ .

## 3. Построение таблицы табуляции

Будем запускать нашу программу командой *Run / Start*, вводить поочередно эти значения  $Z$  в текстовое окно *TextBox1* и для каждого из них выписывать значения  $X$  и  $Y$ . Результаты этой «ручной» работы оформим на бумаге в виде таблицы табуляции 6.

Таблица 6

$Z$	$X$	$Y$
0.2	0.24	0.001
0.4	0.192	0.030
0.6	0.648	0.220
0.8	1.536	0.220
1.0	3.000	0.183
1.2	5.184	0.103

#### 4. Анализ полученных результатов.

Проведем анализ полученных результатов.

Как видим, при разных значениях  $Z = 0.6$  и  $Z = 0.8$  получено одинаковое значение  $Y = 0.220$ . Для выяснения причины этого явления вставим два оператора *Stop*. Первый — перед строкой со словом *Else*, а второй — перед словами *End If*. Тем самым будет проверена работа обеих ветвей условного оператора. Запустим программу и введем  $Z = 0.6$ . Программа остановится на первом операторе *Stop*, т.е. на нужной ветви программы.

Подведем указатель мыши к переменной  $Y$  программы перед оператором *Stop*. Рядом с указателем мыши появится значение 0.2196. Выйдем и снова запустим программу, введя  $Z = 0.8$ . Теперь остановка будет на втором операторе *Stop*, но значение  $Y = 0.2203$ . Совпадение результатов в программе при  $Z = 0.6$  и  $Z = 0.8$  объясняется округлением числа функцией *Format*, так как выведено только три знака после точки.

Удалим операторы *Stop* или отключим их путем ввода апострофа перед ними (см. программный код).

Студенты на бумаге чертят пустую таблицу аналогичную таблице 6, проводят описанные выше вычисления применительно к своему варианту, заполняют графу  $Z$  своими значениями. Затем шесть раз запускают программу на выполнение, из табл. 6 вводят в текстовое окно `TextBox1` **Значение переменной  $Z$** , щелкают по кнопке **Вычислить** и выписывают результаты вычислений в графы  $X$  и  $Y$ . Таким образом, осуществляется ручная табуляция выражений для  $X$  и  $Y$ .

#### Контрольные вопросы

1. Назначение элемента управления `CommandButton` (Командная кнопка).
2. Операции сравнения и логические операции.
3. Синтаксис оператора `If...Then` в однострочной и блочной форме. Всегда ли они взаимозаменяемы? Как работает оператор `If...Then`? Нарисуйте соответствующий фрагмент блок-схемы.
4. Синтаксис оператора `If...Then...Else`. Как он работает? Нарисуйте соответствующий фрагмент блок-схемы.
5. Синтаксис оператора `If...Then...ElseIf`. Как он работает?
6. Нарисуйте блок-схему алгоритма, содержащую математические выражения вашего варианта лабораторной работы.
7. Как вычислить значения исходной переменной  $Z$ , обеспечивающие тестирование программы вашего варианта лабораторной работы?
8. Как вывести результат вычислений в текстовый блок `TextBox` с использованием функции `Format`?
9. Пусть имеется следующий фрагмент программного кода:  
 $X = 455.765$



Text2.Text = Format(X, "0.000")

Что будет появляться в окне TextBox, если в этом фрагменте будем изменять символы форматирования (в кавычках) следующим образом: “0.0”, “0.00”, “0.000”, “0.0000”, “00.000”, “0000.0000”. “#####.#####”.

10. Что такое точка останова, как ее ввести и как пользоваться?

11. Назначение окна *Immediate*.

12. Как вывести в окно *Immediate* значение переменной программы в точке останова?

13. Как получить значение переменной в точке останова программы не пользуясь окном *Immediate*?

14. Как проводить вычисления в окне *Immediate*?

### **Лабораторная работа 3. ОПЕРАТОРЫ ЦИКЛА И МАССИВЫ**

*Цель работы:* научиться в среде *Visual Basic* разрабатывать проекты, реализующие циклические алгоритмы и использующие для хранения данных массивы.

#### **З а д а н и е**

Составить программу, которая вводила бы очередное значение  $Z$ , используя оператор цикла, и для каждого значения осуществляла бы расчеты  $X$  и  $Y$  по тем же формулам, что и в предыдущем задании (см. лабораторную работу 2). Причем, полученные в циклическом процессе значения  $Y$  сохранялись бы в одномерном массиве. После каждого расчета программа должна распечатать значения  $Z$ ,  $X$  и  $Y(i)$ , где  $i$  – индекс массива. Таким образом, должна быть получена таблица табуляции, содержащая те же значения  $Z$ ,  $X$ ,  $Y(i)$ , что и в табл.б .

#### **Пример выполнения задания**

##### **1. Создание новой папки.**

В рабочей папке (там, где хранится папка с проектом Вычисления) создадим новую папку с именем *Циклы*. Скопируем все файлы из папки *УслОпер* в папку *Циклы*. Убедимся, что в новой папке проект работает точно так же, как и в старой.

##### **2. Доработка интерфейса проекта, созданного в лабораторной работе №2.**

1) Для формы в окне свойств зададим свойству Width значение, равное 10095. Ширина формы увеличится (переместится вправо ее правая граница).

2) Правее надписи Label2 мышкой разместим надпись Label7.

3) Ниже ее в одну строку введем три надписи Label8, Label9, Label10.

4) Ниже Label7 введем объект управления Line1 (линия).

5) Ниже надписей Label8, Label9, Label10 введем элемент управления Line2.

б) Введем значения свойств новых надписей, приведенные в табл. 7.

Таблица 7

Свойства	Label7	Label8	Label9	Label10
Left	6240	6350	7440	8530
Top	480	900	900	900
Height	375	375	375	375
Width	2900	375	375	375
Размер шрифта	14	14	14	14

7) Для линий установим следующие значения свойств (это координаты их концов):

Line1:  $X1 = 6240$ ,  $X2 = 9120$ ,  $Y1 = 840$ ,  $Y2 = 840$

Line2:  $X1 = 6240$ ,  $X2 = 9120$ ,  $Y1 = 1440$ ,  $Y2 = 1440$ .

8) Для ввода надписей свойствам Caption для Label4, Label5, Label6, Label7, Label8, Label9, Label10 в окне свойств введем соответственно следующие значения: *Начальное значение*, *Конечное значение*, *Шаг*, *Табличные значения*, Z, X, Y.

9) Свойству Caption кнопки Command1 введем значение *Табулировать*.

10) В текстовые поля Text1, Text2, Text3, расположенные правее надписей *Начальное значение*, *Конечное значение* и *Шаг* свойству Text в окне свойств введем числовые значения Z, полученные в лабораторной работе 2 *Условные операторы* (см. табл. 6).

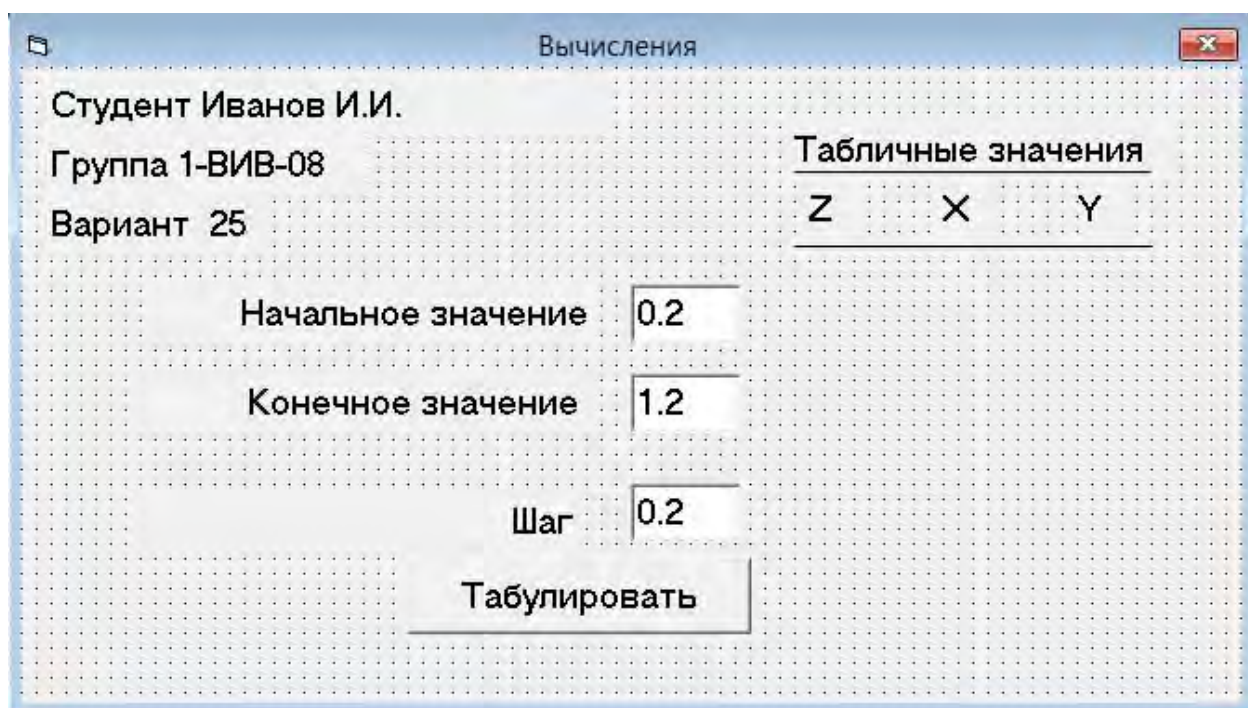


Рис. 5

В результате форма в режиме конструктора примет вид, показанный на рис. 5.

### 3. Ввод программного кода

Нам останется подправить существующий программный код, скопированный из лабораторной работы 2. Для этого необходимо выполнить следующее.

1) Вместо переменной  $Y$  оператором *Dim* объявим массив  $Y(6)$ . Здесь число 6 — это верхнее значение индекса. Массив  $Y(6)$  состоит из 7 элементов:  $Y(0)$ ,  $Y(1)$ ,  $Y(2)$ ,  $Y(3)$ ,  $Y(4)$ ,  $Y(5)$ ,  $Y(6)$ .

2) Объявим целочисленную переменную  $i$  для вычисления индекса массива.

3) Объявим оператором *Dim* переменные  $A$ ,  $B$ ,  $h$ ,  $i$ . В них соответственно будут содержаться: начальное значение параметра цикла  $Z$ , конечное значение параметра цикла  $Z$ , значение шага изменения параметра цикла  $Z$ , индекс элементов массива  $Y$ .

Значения переменных  $A$ ,  $B$ ,  $h$  введем как соответствующие значения свойства *Text* объектов  $Text1$ ,  $Text2$ ,  $Text3$ .

4) Введем положение начала *Списка вывода результатов* относительно левого верхнего угла объекта (в нашем случае формы), которое определяется операторами *CurrentX* (по горизонтали) и *CurrentY* (по вертикали).

В нашем примере положим оператор  $CurrentX=6240$ ,  $CurrentY=1400$ .

5) В конце тела цикла введем метод *Print* для вывода результатов на печать. Установим формат вывода результатов для  $Z$  с одним знаком после запятой, а для  $X$  и  $Y(i)$  с тремя знаками после запятой.

6) Таким образом, наш программный код будет иметь вид:

```
Private Sub Command1_Click()  
Dim Z As Single, X As Single, Y(6) As Single  
Dim A As Single, B As Single, h As Single, i As Integer  
A = Val(Text1.Text)  
B = Val(Text2.Text)  
h = Val(Text3.Text)  
CurrentY = 1400  
i = 0  
For Z = A To B Step h  
    X = 3 * Z ^ 3  
    i = i + 1  
    If X < 1 Then  
        Y(i) = X ^ 2 * Exp(-X)  
    Else  
        Y(i) = Log(X ^ 2) / (X * (1 + X))  
    End If  
CurrentX = 5000
```

```

Print Format(Z, "0.0"); Space(5); Format(X, "0.000"); _
Space(5); Format(Y(i), "0.000")
Next Z
'Stop
End Sub

```

### Комментарии к программному коду

1) На каждом шаге цикла индекс массива вычисляется с помощью рекуррентной формулы (счетчика):  $i = i + 1$ . В цикле он принимает 6 значений: 1, 2, 3, 4, 5, 6. До входа в цикл значение  $i$  приравнивается к нулю. Элемент массива  $Y(0) = 0$ , так как не вычисляется.

2) Главная особенность этого программного кода — использование оператора цикла *For ... Next*. В нашем примере *Параметр цикла* — это переменная  $Z$ , для которой установлено начальное значение *Начало* = 0.2, конечное значение *Конец* = 1.2, шаг *Приращение* = 0.2. Операторы тела цикла — это вычислительные операторы нашего примера.

3) Использование в программном коде одномерного массива  $Y(6)$  позволяет сохранить все вычисленные значения  $Y$ .

В этом легко убедиться, если выполнить следующие действия.

- После отладки программы перед оператором `End Sub` введем оператор `Stop`.
- Запустим программу на выполнение командой `Run/Start`, нажмем кнопку *Табулировать*.
- Программа остановится на операторе `Stop`.
- Убедимся, что в памяти ПК останутся все 7 элементов массива  $Y$ .

Для этого вызовем окно Immediate командой `View/Immediate Window`, щелкнем мышкой в этом окне, наберем на клавиатуре следующую строку:

```
?Y(0), Format(Y(0), "0,000")
```

Нажмем клавишу `[Enter]`. Ниже появится результат:

```
0 0.000 ,
```

где 0 — это значение элемента  $Y(0)$ , а 0.000 — это форматированное значение элемента  $Y(0)$ . Не запуская программы, аналогично можно получить значения остальных элементов массива  $Y$ .

4) Оператор `CurrentY=1400` располагается перед заголовком цикла. Он указывает положение по вертикали первого списка, а остальные списки (со второго по шестой) будут печататься каждый раз со следующей ниже строки. Эти списки вывода печатались бы от левого края формы, но оператор `CurrentX = 6240`, находящийся перед оператором `Print` в теле цикла, заставит выводить список по горизонтали каждый раз на одинаковом расстоянии 5000 твипов от левого края формы.

5) Так как метод `Print` находится в теле цикла, он будет печатать в каждом заходе (шесть раз) три форматированные переменные —  $Z$ ,  $X$  и  $Y(i)$  разделенные пятью пробелами (для разделения используется функция `Space(5)`).

## 5. Запуск проекта на выполнение.

- 1) Для запуска проекта (приложения) введем команду *Run / Start*.
- 2) Появится окно формы (рис.6).
- 3) В нем уже введены *Начальное значение*, *Конечное значение* и *Шаг* в соответствующих текстовых полях.
- 4) Щелкнем мышью по кнопке *Табулировать*.
- 5) Появится результат, в виде таблицы табуляции, совпадающей с таблицей 6:

		Табличные значения		
		Z	X	Y
Начальное значение	0.2	0,2	0,024	0,001
	0.4	0,4	0,192	0,030
	0.6	0,6	0,648	0,220
	0.8	0,8	1,536	0,220
	1.0	1,0	3,000	0,183
	1.2	1,2	5,184	0,103

Рис.6. Результат решения задачи

## Контрольные вопросы

1. Синтаксис оператора цикла `For...Next`. Объясните его работу.
2. Синтаксис операторов `CurrentX`, `CurrentY`. Объясните их работу.
5. Разделители элементов списка вывода оператора `Print`. Как эти разделители влияют на формат печати?
6. Чем завершается список вывода? Как это влияет на формат печати?
7. Синтаксис функции `Space(n)`, ее назначение.
8. Как называются и для чего предназначены составные части строки программного кода этой лабораторной работы:  
`Print Format(Z, "0.0"); Space(5); Format(X, "0.000"); _  
Space(5); Format(Y(i), "0.000")`
9. Как работает программа лабораторной работы в каждом заходе цикла.
10. Как, используя точку останова, определить значения переменных `A`, `B`, `h`.
11. Что такое массив?

12. Синтаксис оператора объявления массива фиксированного размера (вектора и матрицы). Что записывается в круглых скобках?
13. Синтаксис оператора объявления динамического массива.
14. Чему равен нижний предел индекса массива по умолчанию.
15. Как объявляется глобальный массив, массив уровня модуля, локальный массив?
16. Как указать значения нижнего и верхнего индекса массива?
17. Назначение оператора Option Base 1.
18. Как просмотреть все значения массива Y(6).
19. Как в окне Immediate вычислить сумму первого и пятого элементов массива Y(6), не изменяя программного кода, а только введя точку останова перед оператором End Sub?

#### **Лабораторная работа 4. ОПЕРАЦИИ НАД МАТРИЦАМИ И ВЕКТОРАМИ**

##### **З а д а н и е**

Разработайте приложение, реализующее:

- ввод пользователем числа  $p$  — порядкового номера студента по журналу учебной группы, а также числа  $q$  — номера учебной группы;
- расчет в программном коде размерности  $n$  матриц и векторов по формуле

$$n = \begin{cases} 3, & \text{если } 1 \leq p \leq 15 \\ 4, & \text{если } 16 \leq p \leq 30; \end{cases}$$

- циклические вычисления значений элементов исходных матриц  $A_{i,j}$ ,  $B_{i,j}$ ,  $C_{i,j}$  и векторов  $x_i$ ,  $y_i$  по следующим формулам:

$$x_i = \frac{q + p \cdot i}{\sqrt{i^2 + n}}, \quad y_i = 10 \frac{\sin(q + p \cdot i)}{\sqrt{i^2 + n}},$$

$$A_{i,j} = (p + q) \cdot \sin\left(\frac{i \cdot p \cdot q}{n^2 - j}\right), \quad B_{i,j} = \frac{p + (i + j) \cdot j}{i \cdot (j + 1)} \cdot n^{|i-j|},$$

$$C_{i,j} = \frac{5 \cdot \ln(p + q) - e^{i+j}}{n^{|i-j|} + 5},$$

где  $i$  и  $j$  — параметры цикла;

- решение четырех задач, номера которых выбираются из табл. 6.

Таблица 6

Номер варианта	Номера задач	Номер варианта	Номера задач
1	1, 11, 21, 31	16	6, 17, 28, 39
2	2, 12, 22, 32	17	7, 18, 29, 40
3	3, 13, 24, 34	18	8, 19, 30, 33
4	4, 14, 25, 33	19	9, 20, 24, 35
5	5, 15, 25, 33	20	10, 12, 25, 36
6	6, 16, 23, 36	21	1, 13, 26, 37
7	7, 17, 27, 37	22	2, 14, 23, 38
8	8, 18, 28, 38	23	3, 15, 27, 39
9	9, 19, 29, 39	24	4, 16, 28, 40
10	10, 20, 30, 40	25	5, 17, 29, 33
11	1, 12, 24, 33	26	6, 18, 30, 35
12	2, 13, 25, 35	27	7, 19, 24, 36
13	3, 14, 26, 36	28	8, 20, 25, 37
14	4, 15, 23, 37	29	9, 12, 26, 38
15	5, 16, 27, 38	30	10, 13, 23, 39

Номер варианта совпадает с числом  $p$ . Решение задач содержит составление блок-схем алгоритмов, ввод и отладку программного кода. Пользовательский интерфейс и структура приложения определены ниже в примере выполнения задания.

### Содержание задач, номера которых указаны в таблице 6

1. Вычислить координаты вектора  $W$ , разделив каждую координату вектора  $y$  на величину, равную среднему геометрическому координат вектора  $x$ .

2. Вычислить величину  $Q = S + V$ , где  $S$  – сумма координат вектора  $x$ , а  $V$  – произведение координат вектора  $y$ .

3. Вычислить длину вектора  $D = 2x - 3y$ .

4. Вычислить величину  $R = S + L$ , где  $S$  – среднее арифметическое координат вектора  $x$ , а  $L$  – длина вектора  $y$ .

5. Вычислить координаты вектора  $W = C y$ , где  $C$  – скалярное произведение векторов  $x$  и  $y$ .

6. Вычислить длину и произведение координат вектора  $V = \frac{x}{2} + 0.3 \cdot y$ .

7. Найти скалярное произведение вектора  $(2x - y)$  и вектора  $(x + y)$ .

8. Вычислить длину вектора  $R$ , координаты которого определяются по формуле  $R_i = x_i - y_i^2$ .

9. Найти сумму положительных элементов матрицы  $A$ .

10. Найти произведение отрицательных элементов матрицы  $C$ .

11. Получить матрицу  $F$  по формуле  $F_{i,j} = x_j^i$ , где  $x_j$  – координаты вектора  $x$ .

12. Найти след матрицы  $T$ , каждый элемент которой вычисляется по формуле  $T_{i,j} = x_i \cdot y_j$ , где  $x_i$  – координаты вектора  $x$ , а  $y_j$  – координаты вектора  $y$ .

13. Вычислить длину вектора  $U$ , координатами которого являются элементы второго столбца матрицы  $A$ .

14. Найти произведение матрицы  $A$  на вектор  $x$ .

15. Из элементов матрицы  $C$  сформировать два вектора  $U$  и  $V$ , координатами вектора  $U$  являются элементы главной диагонали матрицы  $C$ , а координатами вектора  $V$  являются элементы побочной диагонали матрицы  $C$ .

16. Определить вектор  $Z$ , координаты которого равны соответственно произведениям элементов столбцов матрицы  $A$ .

17. Определить вектор  $Z$ , координаты которого равны соответственно суммам элементов строк матрицы  $B$ .

18. Получите новую матрицу  $D$ , элементы первых  $n$  столбцов которой совпадают с элементами матрицы  $A$ , а элементы  $(n + 1)$ -го столбца равны координатам вектора  $x$ .

19. Получить новую квадратную матрицу  $F$  порядка  $n$  путем замены в матрице  $A$  элементов  $(n - 1)$ -го столбца координатами вектора  $x$ , а элементов  $n$ -го столбца координатами вектора  $y$ .

20. Вычислить матрицу  $D$ , каждый элемент которой определяется по формуле  $D_{i,j} = A_{i,j} + F_{i,j}$ . Причем матрица  $A$  задана, а элементы матрицы  $F$  заданы соотношениями:  $F_{i,j} = A_{i,j}$ , если  $A_{i,j} > 0$  и  $F_{i,j} = 1$ , если  $A_{i,j} \leq 0$  ( $i, j = 1, 2, \dots, n$ ).

21. Получить матрицу  $D$ , каждый элемент которой определяется формулой  $D_{i,j} = 2A_{i,j} - F_{i,j}$ . Матрица  $A$  задана, а элементы матрицы  $F$  вычисляются по формулам  $F_{i,j} = \sin(A_{i,j})$ , если  $A_{i,j} > 0$  и  $F_{i,j} = \cos(A_{i,j})$ , если  $A_{i,j} \leq 0$ . ( $i, j = 1, 2, \dots, n$ ).

22. Найти произведение наибольшего элемента вектора  $x$  и наименьшего элемента вектора  $y$ .

23. Вычислить элементы вектора  $V = \frac{1}{k} \cdot (x + y)$ , где  $k$  – сумма наибольших элементов векторов  $x$  и  $y$ . Получение наибольшего элемента вектора организовать в виде процедуры Function.

24. Дана матрица  $A$  порядка  $n$ . Получить новую матрицу  $D$ , разделив все элементы матрицы  $A$  на ее наибольший элемент.

25. Вычислить вектор  $V$  по формуле:  $V = kx + ly$ , где  $k$  – наибольший элемент матрицы  $A$ , а  $l$  – наименьший элемент матрицы  $B$ .

26. Вычислить угол между векторами  $x$  и  $y$ . Вычисление скалярного произведения и длин векторов организовать в виде одной процедуры Function.



27. Вычислить элементы матрицы  $D = A B - 4 C C$ . Вычисление произведения матриц организовать в виде процедуры Function.

28. Вычислить суммы элементов, расположенных на главных диагоналях матриц  $A$ ,  $B$  и  $C$ . Найти среднее геометрическое этих сумм. Вычисление суммы элементов, расположенных на главных диагоналях матрицы организовать в виде процедуры Function.

29. Найти наибольший элемент вектора  $z$ , координаты которого вычисляются по формулам:  $z_i = x_i y_i$ , если  $x_i + y_i > 0$  и  $z_i = x_i + y_i$ , если  $x_i + y_i \leq 0$  ( $i = 1, 2, \dots, n$ ).

30. Вычислить среднее арифметическое наименьших элементов матриц  $A$ ,  $B$  и  $C$ . Получение наименьшего элемента матрицы организовать в виде процедуры Function.

31. Вычислить среднее геометрическое модулей наибольших элементов матриц  $A$ ,  $B$  и  $C$ . Получение модуля наибольшего элемента матрицы организовать в виде процедуры Function.

32. Вычислить матрицу  $D = A^T - B C$  и заменить отрицательные элементы матрицы  $D$  нулями.

33. Найти матрицу  $D = 4 A - B C$  и заменить в ней положительные элементы единицами.

34. Найти разность наименьших элементов матриц  $A$  и  $B$ . Получение наименьшего элемента матрицы организовать в виде процедуры Function.

35. Вычислить матрицу  $D = \frac{A}{3} + 2 \cdot B - C^T$  и найти сумму элементов главной диагонали этой матрицы.

36. Вычислить среднее геометрическое модулей наибольшего и наименьшего элементов матрицы  $C$ .

37. Определить матрицу  $D = k (A + C)$ , где  $k$  – наименьший элемент матрицы  $B$ .

38. Переписать все элементы  $B$  по строкам с сохранением порядка следования элементов в вектор  $Z$ .

39. Вычислить сумму элементов матрицы  $D$ , определяемой формулой  $D = \frac{1}{3} \cdot A + B^T - C^2$ , где  $B^T$  – транспонированная матрица  $B$ , а  $C^2 = C C$ .

40. Вычислить матрицу  $D$  по формуле  $D = A (C - B)$ . Получить новую матрицу  $F$ , умножив положительные элементы матрицы  $D$  на 5.

### Пример выполнения задания

Пусть требуется решить следующие задачи.

**Задача 41.** Найти минимальный элемент каждой строки квадратной матрицы  $A$  и поместить их на главной диагонали, а диагональные элементы записать на место минимальных.

**Задача 42.** Вычислить сумму элементов матриц  $A$ ,  $B$ , и  $C$ .

**Задача 43.** Найти наибольший элемент главной диагонали квадратной матрицы  $A$ , и вывести на печать всю строку, в которой он находится.

**Задача 44.** Вычислить сумму элементов матрицы  $B$ , расположенных над главной диагональю.

**Замечание.** Отметим, что наш проект будет содержать: две формы – стартовую *Form1* и форму для вывода результатов *Output* с необходимыми элементами управления на них, три программных кода и один модуль.

Последовательность выполнения этого задания будет следующая.

### 1. Создание новой папки.

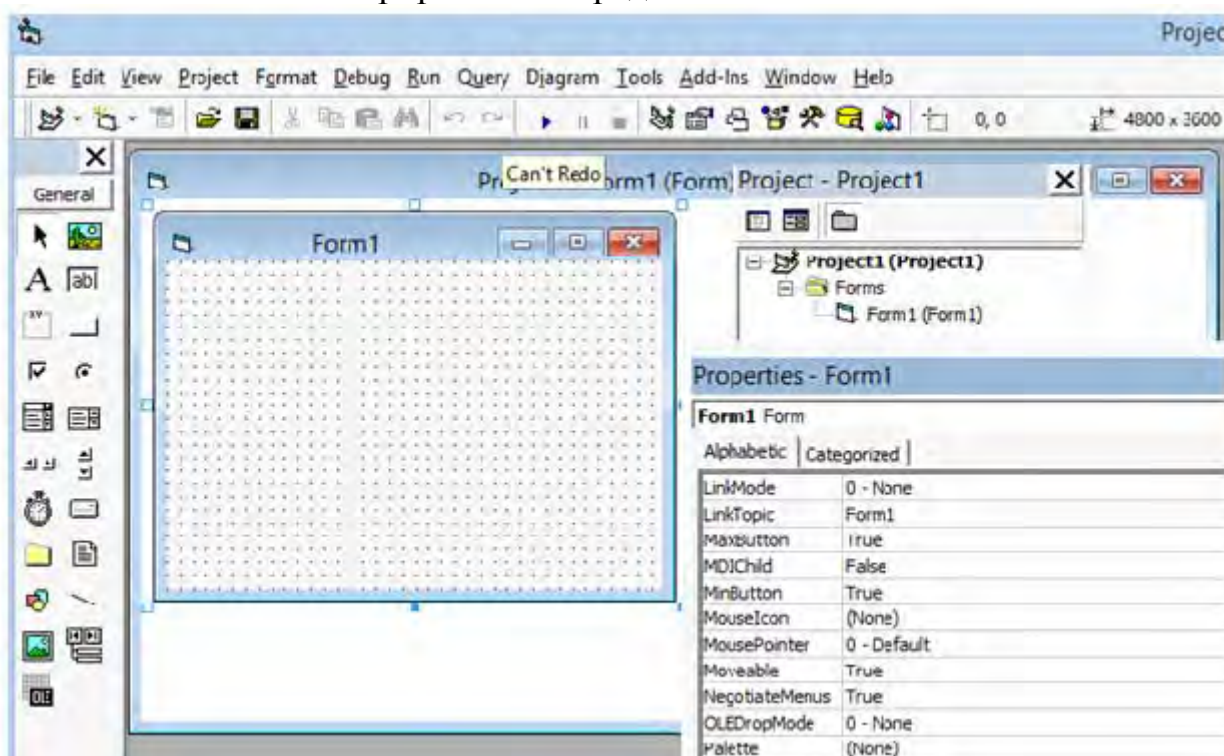
Создадим специальную папку для хранения файлов проекта, присвоим ей имя, например, *MatrVect*.

### 2. Запуск среды Visual Basic.

Запуск среды Visual Basic осуществляется командой:  
Пуск/Программы/MS Visual Basic.

3. Возникнет окно *New Project*, в котором во вкладке *New* выберем шаблон *Standart.EXE*.

4. Появится окно интегрированной среды *Visual Basic*:



5. Для появившейся при открытии стартовой формы установим следующие свойства:

Width – 4300,  
Height – 4400,

Caption – Операции над матрицами и векторами,  
MinButton – False,  
MaxButton – False.

6. В соответствии с рис. 5 разместим на поверхности формы следующие элементы управления:

- а) три надписи - Label1, Label2, Label3;
- б) два текстовых поля - TextBox1 (правее Label2), TextBox2 (правее Label3);
- в) рамку Frame1;
- г) внутри рамки (слева направо) - четыре переключателя Option1, Option2, Option3, Option4;
- д) две кнопки - Command1 и Command2 внизу формы слева направо.

7. Свойствам Caption надписей, рамки и кнопок присвоим текстовые значения в соответствии с рис. 7.

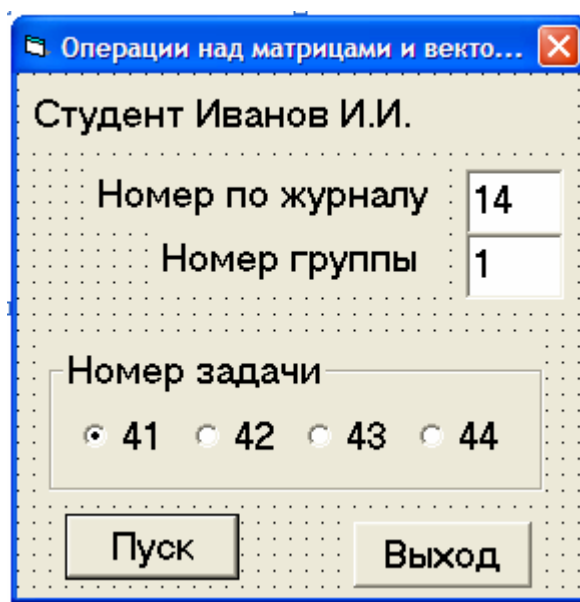


Рис. 7

8. Свойствам Caption переключателей Option1, Option2, Option3, Option4 в окне свойств вместо 41, 42, 43, 44, показанных на рис. 7, присвойте номера задач, соответствующих вашему варианту.

9. Для текстовых полей Text1 и Text2 свойствам Text присвойте значения, равные вашему номеру по групповому журналу и номеру вашей учебной группы.

10. Свойству Value переключателя Option1 присвоим значение True (на нем появится точка выбора по умолчанию, как на рис.7).

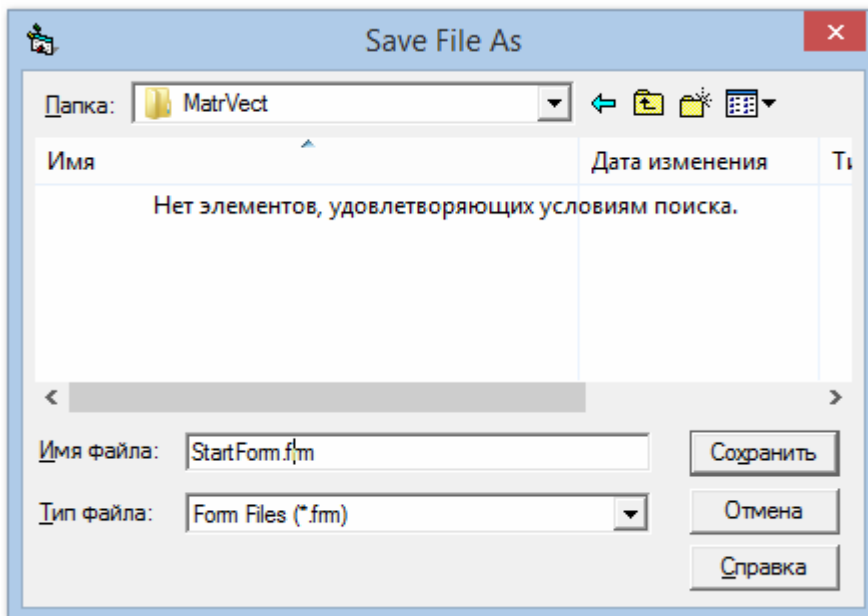
11. Для кнопки «Выход» свойству *Cancel* присвоим значение *True*, для кнопки «Пуск» свойству *Default* присвоим значение *True*. Тогда вместо щелчка мышью по кнопкам *Пуск* и *Выход* можно нажимать клавиши [Enter] и [Esc] соответственно.

12. Размер шрифта для всех элементов управления равен 14 пт.

### 13. Сохранение проекта.

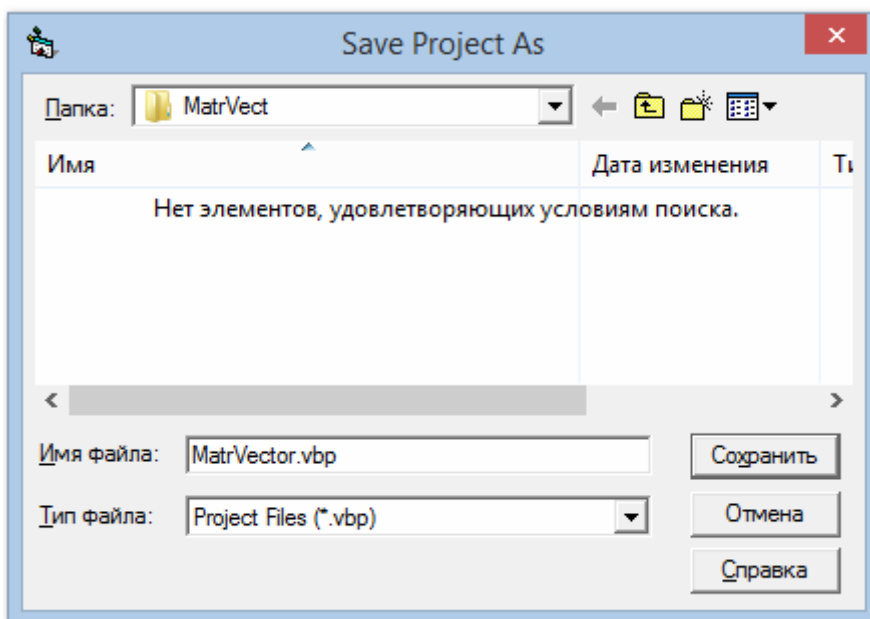
Выполняется командой *File/ Save Project As*

Появится диалоговое окно *Save File As* – окно сохранения формы:



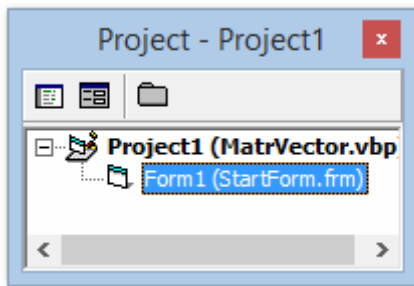
В строке *Папка* укажем папку *MatrVect*. В окне *Имя файла* введем имя файла формы *StartForm*. Нажмем на кнопку *Сохранить*.

Появится диалоговое окно *Save Project As* – окно сохранения проекта:

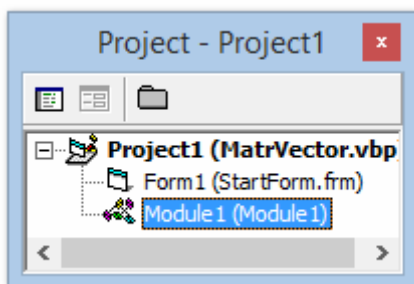


В выпадающем списке *Папка* введено имя папки для проекта. В поле *Имя файла* заменим *Project1* на имя файла проекта *MatrVector.vbp*.

14. Проконтролируем появление названий формы и проекта в окне *Project Explorer*:



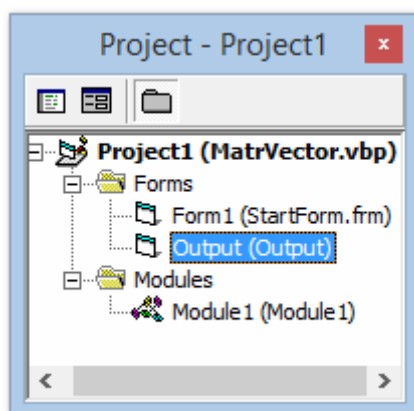
15. Для размещения программного кода, осуществляющего вычисления элементов исходных векторов и матриц и операции над ними, реализующие решения задач, **создадим стандартный модуль**. Для этого введем команду меню *Project / Add Module*. Проконтролируем его появление в окне *Project Explorer*:



Если сделать по его изображению двойной щелчок мышью, то появится окно программного кода стандартного модуля (пока пустое).

16. Программа, размещенная в модуле, должна будет выводить результаты вычислений как в окно Immediate, так и на поверхность специально созданной формы. В нашем проекте **создадим форму с именем Output**. Для этого введем команду *Project / Add Form*.

Сделаем по ней щелчок мышью и введем команду меню *View / Object*. Появится изображение формы. Присвоим ее свойству *Name* имя *Output*. Проконтролируем ее появление в окне *Project Explorer*:



17. Сделав активной форму *Output* щелчком мыши по ее названию, выполним следующие действия.

а) Присвоим некоторым свойствам следующие значения:

Caption (заголовок) — *Вывод исходных данных и результатов вычислений*;

ControlButton – False;

MinButton – False;

MaxButton – False.

б) Мышкой установим размеры формы (пока во весь экран, потом уточним).

в) У нижней границы формы разместим единственный элемент управления Command1 - кнопку с надписью *Заккрыть*.

г) Установим размер шрифта – 14 пт.

Вид формы Output показан на рис. 8.

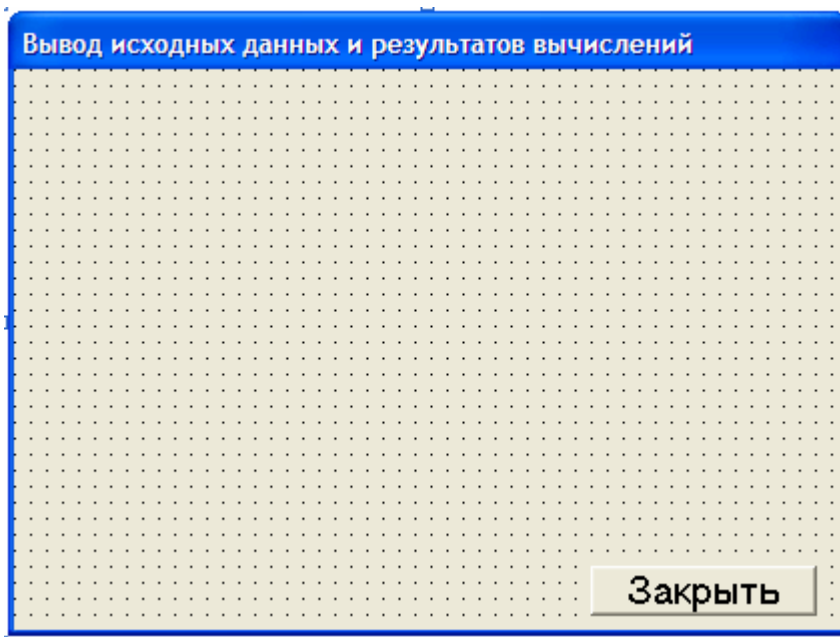


Рис.8. Вид формы Output

## 18. Ввод программных кодов.

Программные коды формы *Form1* состоят из двух процедур обработки события *Click* по кнопкам *Пуск* и *Выход*.

а) Переходим в форму *Form1* и вводим команду *View /Code* (или двойной щелчок по кнопке *Выход*). Откроется окно программного кода, в котором наберем текст программного кода. Процедура обработки события щелчка по кнопке ***Выход*** состоит из единственного оператора *End* (выход из программы):

```
Private Sub Command2_Click()  
End  
End Sub
```

б) Аналогично вводим процедуру обработки события щелчка по кнопке  
**Пуск** :

```
Private Sub Command1_Click()  
Dim p As Integer  
Dim q As Integer  
Dim n As Integer  
p = Val(Text1.Text)  
q = Val(Text2.Text)  
If p > 15 Then  
n = 4  
Else  
n = 3  
End If  
'Debug.Print p , q, n  
Call Massiv(p, q, n)  
End Sub
```

#### *Комментарии к программному коду*

- В этой процедуре оператором *Dim* объявляются имена *p*, *q*, *n* переменных целочисленного типа, обозначающие соответственно номер по журналу, номер группы и число строк и столбцов матрицы. Для векторов *n* — это число их элементов.

- Далее осуществляется считывание значений из текстовых полей *Text1* и *Text2*, преобразование этих значений в числовой тип данных функцией *Val* и присвоение их переменным *p* и *q*.

- Затем вычисляется значение переменной *n*. Если  $p > 15$ , то  $n = 4$ , иначе  $n = 3$ .

- В конце процедуры оператором *Call* вызывается процедура с именем *Massiv*. В круглых скобках после имени функции указываются имена переменных, значения которых передаются вызываемой процедуре (модулю) в качестве исходных данных для вычислений. В нашем примере процедуре *Massiv* передаются значения переменных *p*, *q*, *n*. Так как пока не существует процедуры *Massiv*, программа будет давать сообщение об ошибке. Временно отключите строку с оператором *Call* путем ввода предшествующего апострофа и подключите метод *Debug.Print* (удалите предшествующий апостроф). Тогда значения *p*, *q*, *n* будут печататься в окне *Immediate*, что позволяет контролировать работоспособность программы.

## 19. Ввод программного кода для формы *Output*.

Программный код для формы *Output* состоит из одной процедуры обработки события щелчка по кнопке *Заккрыть*. Переходим в форму *Output* и вводим команду View /Code (или двойной щелчок по кнопке *Заккрыть*). Откроется окно программного кода, в котором наберем следующий текст:

```
Private Sub Command1_Click()  
Form1.Enabled = True  
Output.Hide  
End Sub
```

*Комментарии к программному коду*

В этой процедуре сначала свойству *Enabled* формы *Form1* присваивается значение *True*. Тем самым форма активизируется (становится восприимчивой к воздействиям пользователя). Следующий оператор убирает форму *Output* с экрана.

**20.** Приступим к вводу основной (вычислительной) процедуры *Massiv*. Для этого в **окно стандартного модуля введем следующий программный код.**

```
Public Sub Massiv(p As Integer, q As Integer, n As Integer)  
Dim X() As Single, Y() As Single, A() As Single, B() As Single, C() As Single  
ReDim X(n) As Single  
ReDim Y(n) As Single  
ReDim A(n, n) As Single  
ReDim B(n, n) As Single  
ReDim C(n, n) As Single  
Dim i As Integer, j As Integer  
Dim OptButton As String, S As Single  
Dim Min As Single, Max As Single, k As Integer  
Output.Show 'Загрузка и вызов формы Output  
For i = 1 To n 'Ввод (расчет) элем. векторов X(i) и Y(i)  
X(i) = (q + p * i) / Sqr(i ^ 2 + n)  
Y(i) = 10 * (Sin(q + p * i) / Sqr(i ^ 2 + n))  
'Debug.Print X(i), Распечатка i-го элемента в окно Immediate  
Next i  
'Debug.Print 'Печать пустой строки в окно Immediate  
For i = 1 To n 'Вычисл. элементов матриц A(i,j), 'B(i,j), C(i,j) в цикле  
For j = 1 To n  
A(i, j) = (p + q) * Sin((i * p * q) / (n ^ 2 - j))  
B(i, j) = ((p + (i + j) * j) / (i * (j + 1))) * n ^ Abs(i - j)  
C(i, j) = (5 * Log(p + q) - Exp(i + j)) / (n ^ Abs(i - j) + 5)  
'Debug.Print A(i, j),  
Next j
```



```

'Debug.Print
Next i
'Debug.Print
'В след. четырех строках — присвоение знач. перем. OptButton
If Form1.Option1 = True Then OptButton = "Задача № 41"
If Form1.Option2 = True Then OptButton = "Задача № 42"
If Form1.Option3 = True Then OptButton = "Задача № 43"
If Form1.Option4 = True Then OptButton = "Задача № 44"

Select Case OptButton 'Оператор ветвления на 4 задачи
    Case "Задача № 41"
        Output.Cls
        Output.Print OptButton 'Печать заголовка
        Output.Print 'Печать пустой строки (пропуск строки)
        Output.Print "Исходная матрица"
        For i = 1 To n 'Распечатка матрицы A(i,j)
            For j = 1 To n
                Output.Print A(i, j),
            Next j
            Output.Print
        Next i
        Output.Print
        'Ниже реализован алгоритм решения задачи 41
        For i = 1 To n 'Заголовок цикла по строкам
            Min = A(i, 1)
            For j = 1 To n 'Заголовок цикла по столбцам
                If Min >= A(i, j) Then
                    Min = A(i, j)
                    k = j
                End If
            Next j 'Конец цикла по j (конец i-й строки)
            A(i, k) = A(i, i)
            A(i, i) = Min
        Next i 'Переход к следующей строке
        Output.Print "Преобразованная матрица"
        For i = 1 To n
            For j = 1 To n
                Output.Print A(i, j),
            Next j
            Output.Print
        Next i
        Case "Задача № 42"
            Output.Cls
            Output.CurrentX = 2000
            Output.CurrentY = 200 'Установка позиции по вертикали

```

```

Output.Print OptButton 'Печать заголовка задачи
Output.Print 'Пустая строка
'В следующей строке трижды вызывается процедура 'Function с
именем Sum с двумя передаваемыми ей
'параметрами в круглых скобках: первый параметр — матрица A,
второй параметр - переменная n
S = Sum(A(), n) + Sum(B(), n) + Sum(C(), n)
Output.CurrentX = 200
Output.CurrentY = 800
Output.Print "Сумма элем. матриц A, B, и C равна "; S
    Case "Задача № 43"
Output.Cls
Output.Print OptButton
Output.Print
Output.Print "Исходная матрица"
For i = 1 To n
For j = 1 To n
Output.Print A(i, j),
Next j
Output.Print
Next i
Output.Print
Output.Print "Строка с наиб. диагональным элементом:"
Max = A(1, 1)
For i = 1 To n 'Заголовок цикла
If A(i, i) >= Max Then
Max = A(i, i) 'переменной Max присвоить A(i, i)
k = i 'запомнить номер строки, где выполнилось условие
End If
Next i
For i = 1 To n
Output.Print A(k, i), 'Печать k-й строки
Next i
Output.Print
Output.Print "Номер строки k = "; k 'Печать значения k
    Case "Задача № 44"
Output.Cls
Output.Print OptButton
Output.Print
Output.Print "Исходная матрица"
For i = 1 To n
For j = 1 To n
Output.Print B(i, j),
Next j
Output.Print

```

```

Next i
S = 0
For i = 1 To n
For j = 1 To n
If i < j Then S = S + B(i, j)
Next j
Next i
Output.Print
Output.Print "Сумма элем. матрицы B, расположенных"
Output.Print "над главной диагональю, равна "; S
End Select
Form1.Enabled = False
End Sub

```

**21.** Из процедуры *Sub* с именем *Massiv* вызывается процедура *Function* с именем *Sum*. Эта процедура сразу следует за текстом процедуры *Massiv*. Ее текст приведен ниже:

```

Function Sum(M() As Single, g As Integer) As Single
'Выше располагается заголовок функции Sum. В круглых
'скобках. Указано, что эта функция принимает массив и
'переменные типа Single и Integer. Имена масс. и переем. в
'заголовке и в теле функции должны совпадать
Dim S As Single, i As Integer, j As Integer
S = 0
For i = 1 To g
For j = 1 To g
S = S + M(i, j)
Next j
Next i
Sum = S 'Здесь имени функц. Sum присв. возвр. знач. S
End Function

```

## **22. Запуск проекта на выполнение.**

- 6) Для запуска проекта введем команду *Run / Start*.
- 7) Появится окно формы (рис. 7).
- 8) В нем уже введены *Номер по журналу*, *Номер группы* в соответствующих текстовых полях.
- 9) Щелкнем мышью по кнопке *Пуск*.
- 10) Появится результат решения первой задачи, выведенный на форму *Output*.
- 11) Для запуска второй задачи свойству *Value* переключателя *Option2* (в стартовой форме), присвоим значение *True* (на нем появится точка выбора) и далее аналогично запустим проект на выполнение и т.д. для решения третьей и четвертой задач.

## Контрольные вопросы

1. Что такое модуль? Какие два вида модулей вы знаете?
2. Что такое модуль формы?
3. Что такое стартовая форма? Какая форма будет стартовой?
4. Что такое стандартный модуль?
5. Какие виды процедур вы знаете? Где они хранятся?
6. Что такое процедура обработки события?
7. Главная (пользовательская) процедура Sub. Синтаксис заголовка и назначение процедуры.
8. Процедуры Function. Синтаксис заголовка, назначение процедуры. Как обеспечить возвращение значения процедуры?
9. Чем отличается процедура Sub от процедуры Function?
10. Как передаются параметры в процедуры?
11. Как объявляются динамические массивы?
12. Для чего служит оператор Cls?
13. Напишите синтаксис оператора выбора Select Case.
14. Как распечатать матрицу с одинаковой длиной строк и равными столбцами?
15. Как распечатать вектор в столбец (в строку)?
16. Назначение и основные свойства элемента управления Frame (Рамка).
17. Назначение и основные свойства элемента управления OptionButton (переключатель). Как обеспечить выбор нужного переключателя по умолчанию? (Чтобы на нем была точка сразу после запуска программы).
18. Как создать группу переключателей и организовать их выбор?

### З а д а н и е для самостоятельной работы:

#### Решение нелинейного уравнения методом половинного деления

Целью данной работы является приобретение навыков решения задач *итерационными* методами, в частности решение нелинейных уравнений методом половинного деления, используя конструкцию DO WHILE...LOOP.

#### Метод половинного деления (или дихотомии).

При решении нелинейного уравнения методом половинного деления задаются интервал  $[a, b]$ , на котором существует только одно решение, и желаемая точность  $\epsilon$ .

Постановка задачи. Пусть функция  $y=f(x)$  определена и непрерывна на отрезке  $[a; b]$ .

$$f(a) \cdot f(b) < 0$$

Требуется найти корень на отрезке  $[a; b]$  с точностью  $\epsilon$ .

#### Последовательность решения задачи.

Для реализации алгоритма дихотомии отрезок  $[a; b]$  делится пополам точкой  $c = (a + b)/2$  (рис. 1).

Если  $f(c)$  не равно 0, то возможны два случая:

а)  $f(x)$  меняет знак на отрезке  $[a; c]$ ;

б)  $f(x)$  меняет знак на отрезке  $[c; b]$ .

Выбираем тот отрезок, на котором функция меняет знак. Если  $f(x)$  меняет знак на отрезке  $[a; c]$ , то  $b=c$ ; если  $f(x)$  меняет знак на отрезке  $[c; b]$ , то  $a=c$ .

Деление отрезка пополам продолжается пока  $|b-a| > \epsilon$ .

Корень уравнения:  $x = (a + b)/2$

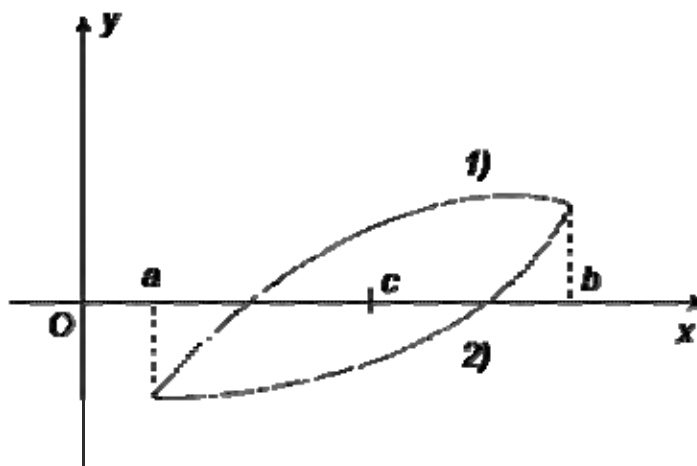
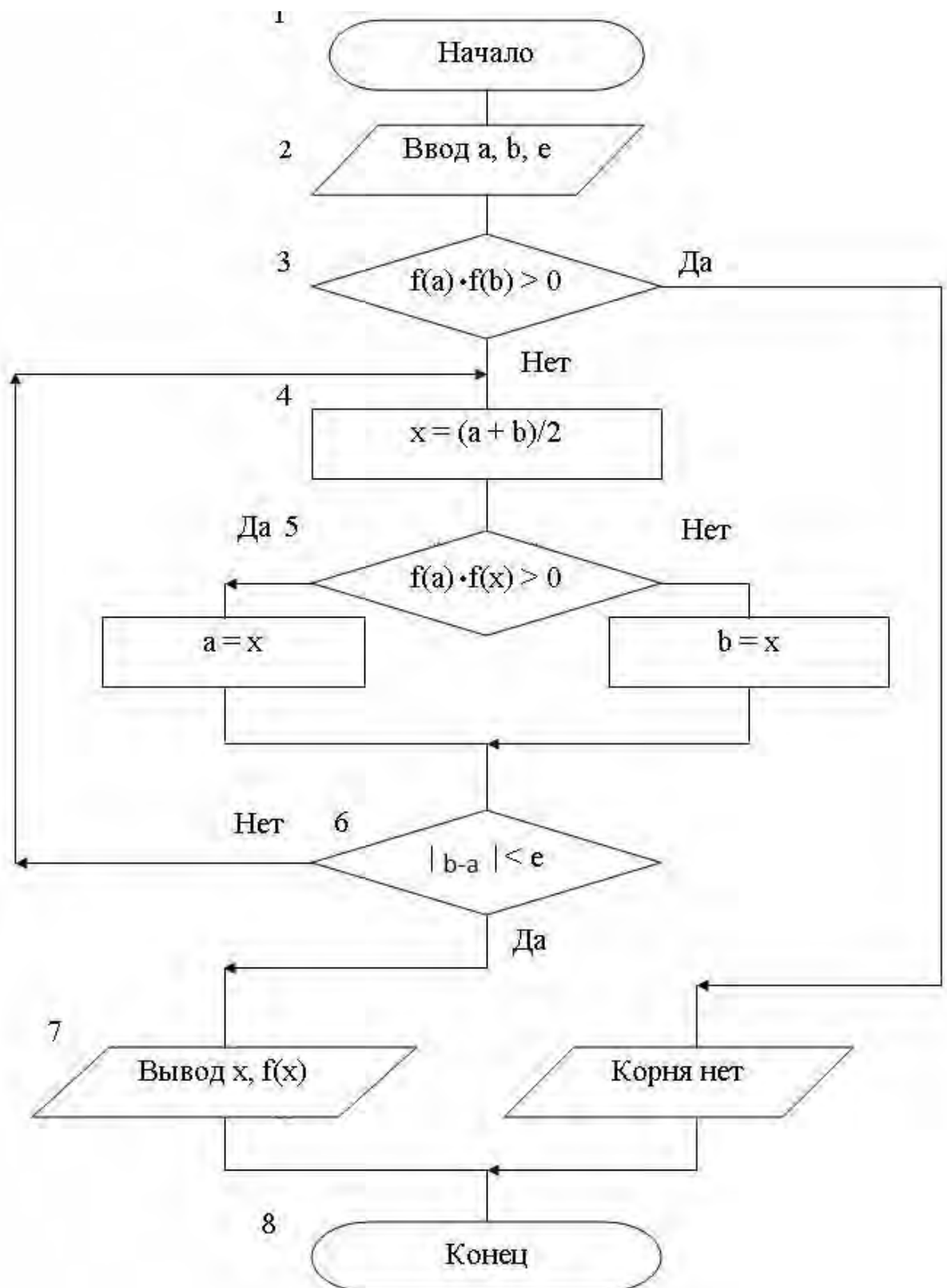


Рис.1. Графическая иллюстрация метода половинного деления.

### Алгоритм решения нелинейного уравнения методом половинного деления



### З а д а н и е.

Найти корень нелинейного уравнения методом деления пополам с точностью  $\epsilon=0.001$ .

### Варианты заданий для самостоятельной работы

№ варианта	Уравнение	Интервал
1	$x^4+x^2-2x-2=0$	$[-1;0]$
2	$\text{tg}(2x) - 1 - x=0$	$[0.3;0.7]$
3	$2^x-2x^2+1=0$	$[1;3]$
4	$e^x+x^2-2=0$	$[0;1]$
5	$\exp(-3x^2)-x=0$	$[0;1]$
6	$2x-3\ln(x)-3=0$	$[3;4]$
7	$\ln(x)-1/x=0$	$[1;2]$
8	$\cos(x)-\ln(x+1)=0$	$[0;2]$
9	$\ln(x)-\sin^2(x)=0$	$[1;3]$
10	$x+x^3-5=0$	$[1;2]$
11	$x-\exp(-3x^2)=0$	$[0.1;1]$
12	$x-\ln(x)-2=0$	$[3;4]$
13	$x^5+1-3x=0$	$[0;1]$
14	$2+\ln(x)-1/x=0$	$[0.1;1]$
15	$x^4+2x-6=0$	$[1;2]$
16	$\cos(x)-\text{tg}(x)=0$	$[0;1]$
17	$2+\ln x-1/x^2=0$	$[0.1;1]$
18	$x+2-x^3=0$	$[1;2]$
19	$\cos(x)-x=0$	$[0.1;1]$
20	$e^x-1/\sin x=0$	$[0.2;1]$
21	$x-0.5-x^8=0$	$[0.1;1]$
22	$x-\exp(-x^2)-2=0$	$[1;3]$
23	$\text{tg}x-1/x=0$	$[0.5;1]$
24	$\cos x-1/x=0$	$[4;6]$
25	$1/e^x-x^2=0$	$[0.1;1]$

### Пример выполнения задания.

Найти корень нелинейного уравнения  $\frac{10\sin(x+e)}{e^{|0.2x|}}+1=0$  методом деления пополам на интервале  $[2;4]$  с точностью  $\epsilon=0.001$ .

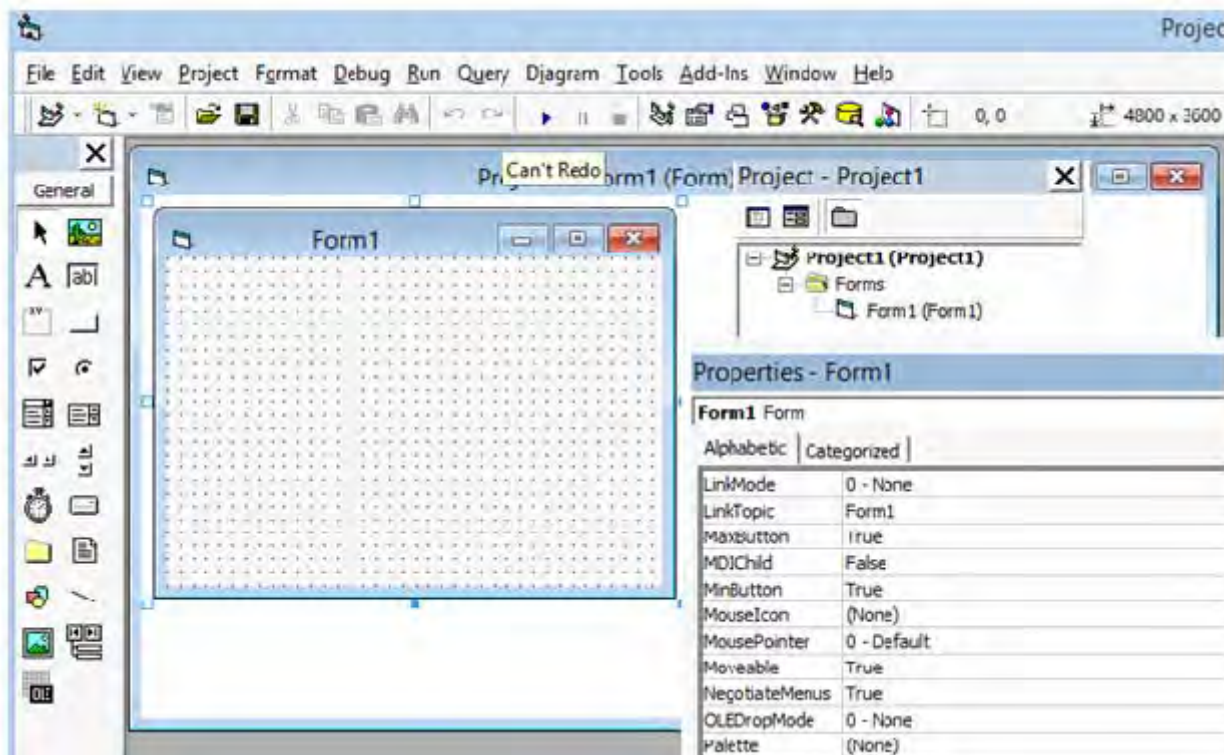
Последовательность выполнения этого задания предполагает следующие действия.

**1. Создание папки для хранения проекта.** Создадим новую папку для хранения файлов проекта, присвоим ей имя, например, *НелинУрав*.

**2. Запуск Visual Basic.** Запустим Visual Basic командой:  
Пуск/Программы/MS Visual Basic.

**3. Создание нового проекта.** Возникнет окно New Project, в котором во вкладке New выберем шаблон Standart.EXE.

**4. Появится окно интегрированной среды Visual Basic:**



**6. Установка свойств формы.** Для появившейся стартовой формы *Form1* установим следующие свойства:

Width – 4300,

Height – 4400,

Caption – Решение нелинейного уравнения

MinButton – False,

MaxButton – False.

**7. Размещение элементов управления на поверхности формы.** В соответствии с рис.2 разместим на поверхности формы следующие элементы управления:

а) семь надписей - Label1, Label2, Label3, Label4, Label5, Label6, Label7;

б) пять текстовых полей - TextBox1 (правее Label3), TextBox2 (правее Label4), TextBox3 (правее Label5), TextBox4 (правее Label6), TextBox5 (правее Label7);

в) одну кнопку - Command1 внизу формы.

Свойствам Caption надписей и кнопке присвоим текстовые значения в соответствии с рис. 2.



Рис. 2. Размещение элементов управления на стартовой форме

Для текстовых полей Text1, Text2, Text3 свойствам Text присвоим значения, соответствующие варианту, в нашем случае: 2, 4, 0.001.

Размер шрифта *Font* выберем равным 14 пт. Тип шрифта *MS Sun Serif*.

**8. Сохранение проекта.** Сохраним проект командой *File/ Save Project As*.

В появившемся диалоговом окне *Save File As* (окно сохранения формы) в строке *Папка* укажем папку *НелинУрав*. В окне *Имя файла* оставим имя файла формы *Form1.frm*. Нажмем на кнопку *Сохранить*.

В появившемся диалоговом окне *Save Project As* (окно сохранения проекта) в выпадающем списке *Папка* введено имя папки для проекта. В поле *Имя файла* оставим *Project1.vbp*.

**9. Создание программного кода.** Переходим в форму *Form1* и вводим команду *View /Code* (или двойной щелчок по кнопке *Вычислить*). Откроется окно программного кода, в котором наберем текст программного кода. Процедура обработки события щелчка по кнопке *Вычислить* состоит из следующих операторов:

## Option Explicit

---

```
Private Sub Command1_Click()
Dim a As Single, b As Single, c As Single, eps As Single
Dim result As Single
a = Val(Text1.Text)
b = Val(Text2.Text)
eps = Val(Text3.Text)
If y(a) * y(b) < 0 Then
c = (a + b) / 2
Do While Abs(b - a) > eps
    If y(a) * y(c) < 0 Then
        b = c
    ElseIf y(b) * y(c) < 0 Then
        a = c
    Else
        a=c
        b=c
    End If
    'Debug.Print a, b, c
    c = (a + b) / 2
Loop
result = y(c)
Text4.Text = Format(c, "0.000")
Text5.Text = result
Else
Text4.Print "Корня нет"
End If
End Sub
```

---

```
Private Function y(x As Single) As Single
y = 10 * Sin(x + Exp(1)) / Exp(Abs(0.2 * x)) + 1
End Function
```

### *Комментарии к программе*

В программном коде используется алгоритм метода половинного деления для вычисления корня нелинейного уравнения. Процесс является итерационным. Он реализуется с помощью конструкции цикла:

Do While Условие

тело цикла

Loop

Цикл выполняется до тех пор, пока *Условие* имеет значение *True*.

В программном коде используется обращение к процедуре-функции `y(x As Single) As Single`, которая вычисляет значение функции в точке.

Формула вычисления значения функции в точке вводится в соответствии с номером варианта.

### 10. Запуск проекта на выполнение.

- Для запуска проекта введем команду *Run / Start*.
- Появится окно формы (рис. 2).
- В нем уже введены *Значение начала интервала*, *Значение конца интервала*, *Заданная точность* в соответствующих текстовых полях.
- Щелкнем мышью по кнопке *Вычислить*.
- Появится результат решения задачи:

Решение нелинейного уравнения

Студент Иванов И.И.

Группа 1-ПГС-15

Значение начала интервала: a	2
Значение конца интервала: b	4
Заданная точность: eps	0.001
Значение корня уравнения	3,368
Значение функции в этой точке	8,376596E-04

Вычислить

### Контрольные вопросы.

1. В чем заключается метод половинного деления?
2. Всегда ли методом половинного деления можно решить нелинейного уравнения?
3. Какие циклические конструкции используются для реализации итерационных процессов?
4. Как работает конструкция DO WHILE...LOOP?

## Список литературы

1. Богомолов А.Н., Степанов М.М., Потапова Н.Н. Программирование на MS Visual Basic. Учебное пособие. Волгоград: ВолгГАСУ, 2005.
2. М.М. Степанов, Н.Н.Потапова. Программирование на MS Visual Basic. Лабораторный практикум по дисциплине «Информатика». Волгоград: ВолгГАСУ, 2009.
3. Глушков С.В., Сурядный А.С. Программирование на Visual Basic 6.0. Учебный курс. М.: изд-во «Фолио», 2003.

## Содержание

I. Теоретическая часть.	
Введение.....	3
1.1. Запуск среды MS Visual Basic.....	3
1.2. Создание нового проекта.....	3
1.3. Сохранение проекта.....	4
1.4. Открытие проекта.....	5
1.5. Визуальная интегрированная среда MS Visual Basic.....	5
1.6. Программирование в среде MS Visual Basic.	
Основные понятия и конструкции.....	12
II. Практическая часть.	
Лабораторная работа 1. Ввод выражений.....	26
Задание.....	26
Пример выполнения задания.....	29
Контрольные вопросы.....	36
Лабораторная работа 2. Условные операторы.....	38
Задание.....	38
Пример выполнения задания.....	39
Контрольные вопросы.....	45
Лабораторная работа 3. Операторы цикла и массивы.....	46
Задание.....	46
Пример выполнения задания.....	46
Контрольные вопросы.....	50
Лабораторная работа 4. Операции над матрицами и векторами....	51
Задание.....	51
Пример выполнения задания.....	54
Контрольные вопросы.....	65
Задание для самостоятельной работы. Решение нелинейного уравнения методом половинного деления (дихотомии).....	66
Задание.....	68
Пример выполнения задания.....	68
Контрольные вопросы.....	72
Список литературы.....	73