

ФЕДЕРАЛЬНОЕ АГЕНСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего
профессионального образования
ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет автоматизации машиностроения

Кафедра «Информационных технологий в машиностроении»

**Лабораторный практикум
По дисциплине
«Информатика»**

Методическое пособие
для выполнения лабораторных работ

Дисц.: Информатика

Спец. 1201, 170400
для студентов
дневного отделения

Киров 2006

Содержание

1	Знакомство с Visual Basic.....	5
1.1	Первое знакомство.....	5
1.1.1	Запуск VB.....	5
1.1.2	Выход из VB	7
1.1.3	Практическое задание.....	7
1.2	Файлы проекта.....	7
1.3	Шаги создания проекта на VB	8
1.3.1	Планирование действий, выполняемых приложением.....	8
1.3.2	Планирование пользовательского интерфейса	8
1.3.3	Установка свойств	9
1.3.4	Написание текста программы	9
1.3.5	Отладка приложения	9
1.4	Первый проект	9
1.4.1	Основные сведения об исполнении приложений в операционной системе Windows.....	9
1.4.2	Разработка проекта	10
1.4.3	Условие задачи	12
1.4.4	Элементы управления.....	14
1.4.4.1	Вставка надписей.....	15
1.2.1	Разработка программы	17
1.3	Усовершенствование проекта	21
1.4	Компиляция проекта.....	25
1.5	Отображение окон в среде VB.....	26
1.6	Вопросы для контроля.....	27
2	Переменные и константы	29
2.1	Основные типы данных VB	30
2.2	Структура проекта	33
2.3	Объявление переменных и констант	34

2.4	Преобразование и совместимость типов	36
2.5	Разработка проекта	38
2.6	Вопросы для контроля.....	46
3	Ввод и вывод значений переменных	47
3.1	Ввод данных с помощью функции InputBox.....	47
3.2	Вывод данных с помощью инструкции Print	51
3.3	Вывод данных с помощью функции MsgBox	56
3.4	Вопросы для контроля.....	58
4	Выражения в VB.....	60
4.1	Арифметические операции	61
4.1.1	Сложение.....	62
4.1.2	Вычитание	62
4.1.3	Умножение	62
4.1.4	Деление.....	63
4.1.5	Возведение в степень.....	63
4.1.6	Целочисленное деление.....	64
4.1.7	Остаток от деления	64
4.1.8	Унарный минус	65
4.2	Логические операции	65
4.2.1	Логическое И	66
4.2.2	Логическое ИЛИ.....	66
4.2.3	Логическое НЕ	67
4.2.1	Операции отношения.....	69
4.3	Символьные операции.....	71
4.4	Синтаксис выражений	71
4.5	Явное преобразование типов данных	73
4.6	Задание для самостоятельной работы	74
5	Конструкции VB, реализующие базовые алгоритмические структуры.....	79
5.1	Базовые структуры алгоритмов	79

5.1.1	Следование (композиция)	79
5.1.2	Ветвление	80
5.1.3	Цикл	83
5.2	Конструкция If..Then	84
5.3	Конструкция Select ... Case	87
5.4	Конструкция Do...Loop	88
5.5	Цикл со счетчиком. Конструкция For...Next	91
5.6	Задание для самостоятельной работы	92
6	Массивы в VB	96
7	Алгоритмы сортировки	103
7.1	Алгоритм сортировки выборкой	103
7.2	Алгоритм сортировки вставкой	106
7.3	Алгоритм пузырьковой сортировки	108
7.4	Алгоритм быстрой сортировки	108
7.5	Алгоритм сортировки слиянием	108
7.6	Алгоритм сортировки подсчетом	108
8	Приложение: Коды ошибок VB	109
9	Описание событий VB	117
10	Приложение: Таблица кодов ASCII	133
11	Приложение: Виртуальные коды клавиш	139
12	Приложение: Функции VB	145
13	Приложение: Дополнительные материалы по вычислению некоторых функций	179
14	Приложение: Операторы VB	180
15	Приложение: Графики функций к самостоятельным заданием п. 4.6	210

1 Знакомство с Visual Basic

1.1 Первое знакомство

Visual Basic (далее будем применять сокращение VB) содержит удобную графическую среду для быстрого создания эффективных приложений для Windows.

1.1.1 Запуск VB

Запустите VB и начните краткое знакомство. Для этого выполните следующие действия:

- щелкните на кнопке **Пуск** (Start) панели задач, выполните команду **Программы** (Programs) и укажите на папку **Microsoft VB**. Появятся значки содержимого папки.
- щелкните на значке программы **Microsoft VB**.

Если появится диалоговое окно нового проекта (**New Project**) (Рисунок 1), в котором можно выбрать один из нескольких типов шаблонов проектов, выберите **Standard.exe** (стандартный шаблон) и щелкните на кнопке **Open** (Открыть) – на экране появится окно среды Microsoft VB (Рисунок 2).



Рисунок 1 - Окно нового проекта

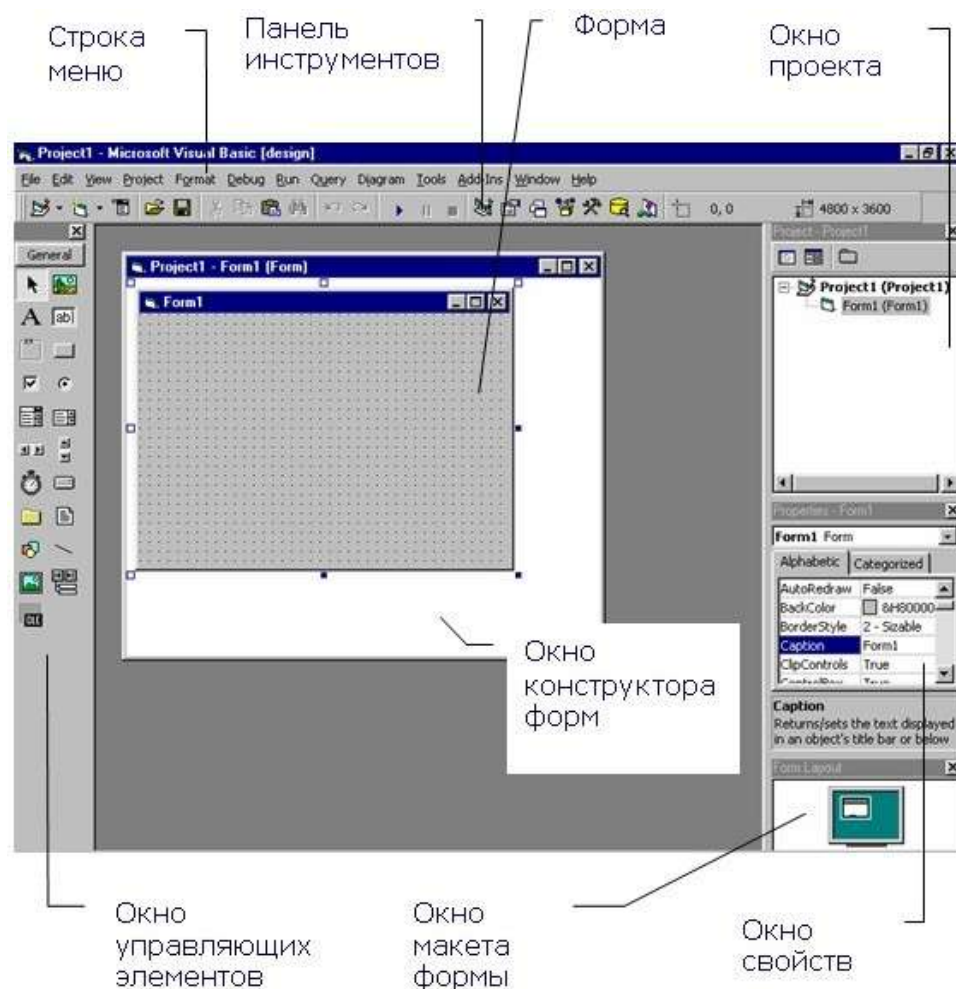


Рисунок 2 - Основные компоненты окна Visual Basic

Рассмотрим основные элементы интерфейса среды разработки VB:

- **Строка меню** VB содержит команды для работы с файлами, просмотра и добавления программных модулей, для выполнения компиляции, а также для конфигурирования среды VB.
- **Панель инструментов** имеет кнопки с пиктограммами, что позволяет быстро обращаться к наиболее часто используемым командам.
- **Окно управляющих элементов** дает возможность очень просто вставлять в свои программы такие элементы управления, как кнопки, текстовые окна, диалоговые окна и прочие элементы.
- **Форма** – это изображение на экране, с которым Вы будете взаимодействовать при проектировании, а также при выполнении Вашей программы.
- **Окно конструктора форм** – позволяет работать с формами в

процессе планирования интерфейса проекта.

- **Окно проекта** содержит имена форм и модулей, входящих в проект.
- **Окно свойств** позволяет задавать по Вашему желанию значения свойств вставленных в форму элементов (кнопок и прочих).
- **Окно макета формы** предназначено для изменения положения на экране окна проектируемого приложения.

1.1.2 Выход из VB

Ознакомимся с самым простым способом выхода из VB:

- Выберите команды меню **File/Exit**.
- Если появиться вопрос, хотите ли Вы сохранить изменения проекта, то выбрать **No**, если Вам не требуется сохранить выполненные изменения или **Yes**, если выполненные изменения требуется сохранить.

1.1.3 Практическое задание

Выполните запуск VB.

Ознакомьтесь с основными элементами интерфейса среды разработки VB.

Выйдите из VB.

1.2 Файлы проекта

В соответствии с принятой в VB терминологией, прикладная программа, которую мы разрабатываем, называется проектом.

В проект могут входить несколько файлов различных видов, в том числе:

- Один файл проекта (имеет расширение .vbp). Включает имена всех файлов проекта;
- Один файл формы для каждой формы (расширение .frm). В этом файле содержится программа, определяющая вид окна, с которым Вы будете

взаимодействовать при выполнении Вашей программы;

- Один файл для каждого стандартного модуля (расширение .bas). В него входит программный текст, который может использоваться различными модулями проекта;

- Файл ресурсов (расширение .res). Содержит данные для проекта (растровые рисунки, текстовые строки), которые можно изменять без необходимости редактирования программы;

- Исполняемый файл (расширение .exe). Является результатом компиляции программы (т.е. перевода программы с языка VB на язык команд компьютера).

1.3 Шаги создания проекта на VB

Разработка прикладной программы для Windows на VB включает шаги, которые могут быть выполнены повторно при необходимости:

- планирование действий, выполняемых приложением;
- планирование пользовательского интерфейса;
- установка значений свойств объектов;
- написание текста программы;
- отладка приложения.

1.3.1 Планирование действий, выполняемых приложением

Это первый шаг в создании прикладной программы. На данном этапе необходимо произвести анализ предметной области и планирование действий выполняемых программой.

1.3.2 Планирование пользовательского интерфейса

На данном этапе разработчик решает, какой набор форм и элементов управления (окна, кнопки, меню, линейки прокрутки и т.д.) сделает удобным взаимодействие пользователя с программой.

1.3.3 Установка свойств

Каждый из элементов управления, применяемых в программе, имеет набор свойств, которые определяют их внешний вид и поведение. На этом шаге задаются значения свойств каждого применяемого элемента управления.

1.3.4 Написание текста программы

На данном этапе разработчик осуществляет ввод исходного кода (текста) приложения.

1.3.5 Отладка приложения

При выполнении отладки осуществляется поиск и исправление ошибок, а также проводится дополнительный анализ и совершенствование приложения.

1.4 Первый проект

В данном пункте вы произведете создание первого приложения. Получите основные сведения об объектно-ориентированном программировании.

1.4.1 Основные сведения об исполнении приложений в операционной системе Windows

При разработке Windows приложений имеющих графический интерфейс широко применяется объектно-ориентированный подход, который позволяет значительно сократить время разработки новых приложений и облегчить повторное применение ранее разработанных компонентов приложений.

Форма приложения, и все элементы управления, используемые при разработке приложений, являются объектами – компонентами приложений. Такие объекты объединяют под одним именем данные (свойства) и


процедуры их обработки (методы), а между собой объекты взаимодействуют с помощью сообщений. Данные компоненты разработаны ранее и поставляются в составе дистрибутива либо виде дополнительных библиотек. Это позволяет использовать компоненты для быстрого создания новых приложений путем компоновки и некоторой доработки имеющихся в распоряжении разработчика описаний объектов.

Разработанные приложения при исполнении взаимодействуют с операционной системой, которая обслуживает работу устройств и ввод пользователя. При возникновении определенных ситуаций (событий) операционная система предупреждает о них окна приложений с помощью специальных сообщений. Получая сообщения о событиях, приложения выполняют предопределенные разработчиком последовательности действий, соответствующие данным сообщениям и называемые обработчиками событий. Таким образом, создавая новое приложение, разработчик компоует интерфейс приложения из элементов управления, а затем описывает обработчики событий, на которые должно реагировать приложение.

1.4.2 Разработка проекта

Приступим к разработке и реализации проекта:

1. Запустите VB. Если появилось окно *New Project*, то выберите *Standart EXE* (стандартный) вкладки *New* (новый) и щелкните на кнопке *Открыть*. Ознакомьтесь с окном проекта.

2. Щелкните на кнопке  *Project Explorer* (Обозреватель проекта) на панели инструментов. Окно проекта (*Project*) теперь выделено. Если это окно было закрыто, сейчас оно появилось.

3. Если в окне проекта слева от пиктограммы *Project1* отображается знак (+), то щелкните на этом значке. Появится пиктограмма *Form1*. Если окно формы не появилось, то дважды щелкните на пиктограмме *Form1* в окне проекта.

4. Если окно свойств (Рисунок 3) на экране не отображается, выберите в строке меню **View** (вид), **Properties** (свойства), чтобы вывести окно свойств **Form1**.

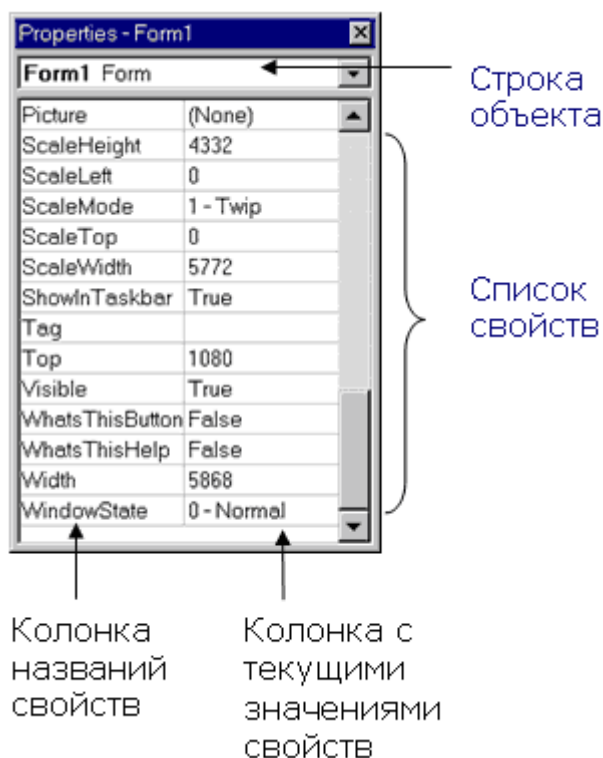


Рисунок 3 - Окно свойств

5. Периодически (после работы в течение десяти – пятнадцати минут) выполняйте сохранение проекта. Если Вы работаете в компьютерном классе, то сохраните проект в каталоге *c:\students\<название группы>\<Фамилия студента>\<лабораторная работа №>\<задание №>* (например, *c:\students\ТМ-13\Иванов\лабораторная работа 1\задание 1*).¹ В дальнейшем мы всегда будем сохранять проекты в подкаталогах каталога *c:\students* с соответствующими именами.

5.1. Для сохранения проекта выполните команды строки меню File (файл) и Save Project (Сохранить проект).

¹ **Внимание:** все каталоги и файлы, расположенные вне описанных каталогов или имеющие некорректные имена, будут удалены администратором аудитории.

5.2. В открывшемся диалоговом окне *Save File As* (Сохранить файл как) выберите каталог проекта. При этом потребуется создать папку, в которой будут храниться все файлы создаваемого проекта.²

5.3. Наберите *MyClock* в поле ввода *File Name* (Имя файла) и нажмите клавишу *Enter*. Форма *Form1* будет сохранена в файле с именем *MyClock.frm*.

5.4. Затем появится диалоговое окно *Save Project As* (Сохранение проекта).

5.5. Наберите *MyClock* и нажмите клавишу *Enter*. Файл проекта будет записан в рабочей папке под именем *MyClock.vbp*.

1.4.3 Условие задачи

Составим простой проект – цифровые часы для Windows.

Цифровые часы, внешний вид которых показан на следующем рисунке (Рисунок 4), должны показывать текущую дату и время. Эта информация должна обновляться при щелчке по кнопке с надписью "Показать время".

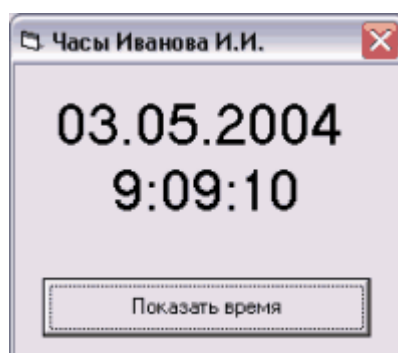



Рисунок 4 - Цифровые часы

² Для создания новой папки щелкните на кнопке  **Создание новой папки** панели инструментов окна *Save File As*. В поле имени появившейся папки введите имя папки, называемой рабочей. Закончив набор имени рабочей папки, нажмите на клавишу *Enter*.

Описав внешний вид цифровых часов, мы практически выполнили этапы разработки проекта, которые называются планированием действий выполняемых приложением и планированием пользовательского интерфейса. Мы уяснили, что должно быть предусмотрено место, где будет выводиться текущая дата и время, а также должна быть предусмотрена кнопка, щелчок на которой приводит к обновлению текущей даты и времени.

6. Прокрутите список свойств формы, пока не дойдете до свойства **Width** (ширина). Посмотрите на текущее значение свойства – это ширина окна **Form1**.

7. Для изменения ширины **Form1** щелкните на свойстве **Width**, чтобы выбрать его. Введите 4000 и нажмите **Enter**³, чтобы изменить значение этого параметра на 4000 твилов. Обратите внимание, что одновременно с этим изменится и ширина формы **Form1**.

8. Можно изменить значения свойства **Width** формы **Form1**, изменяя сам объект. Опробуйте этот второй способ изменения размеров формы:

8.1. Щелкните на окне **Form1**, чтобы сделать его активным.

8.2. Сделайте окно примерно на два сантиметра уже, перетаскив влево его правую границу.

8.3. Убедитесь, что значение свойства **Width** в окне свойств уменьшилось.

9. Измените свойство **Width** формы **Form1** на 4000.

10. Измените свойство **Height** (высота) на 4000.

11. Если Ваше окно **Form1** перекрывает окно элементов, перетащите его (за окно заголовка) с окна элементов вправо.

12. Сохраните проект. Для этого теперь просто достаточно щелкнуть на кнопке **Save Project** панели инструментов.

³ Линейные размеры в VB измеряются в твилах. В одном сантиметре 567 твилов.

1.4.4 Элементы управления

Панель Элементов управления содержит набор условных обозначений (пиктограмм) элементов, которые можно просто включить в проект. При разработке данного приложения потребуются три элемента: ***Label*** (Надпись), ***Command Button*** (Кнопка) и ***Timer*** (Таймер) (Рисунок 5).



Рисунок 5 - Панель элементов управления

Чтобы добавить на форму элемент управления надо:

- сделать активной форму, в которую Вы хотите добавить элемент управления;
- дважды щелкнуть на требуемом элементе на панели элементов управления (или щелкнуть на нем один раз, переместить указатель в ту точку на форме, в которой должен находиться левый верхний угол элемента, нажать левую кнопку мыши, переместить указатель мыши в точку, где должен находиться правый нижний угол элемента и отпустить кнопку мыши);
- изменить, если требуется, положение и размер элемента управления.

1.4.4.1 Вставка надписей

Элемент **Label** применяется для создания на форме окна с текстом.

На наших цифровых часах требуется отвести место на циферблате, куда должны выводиться текущая дата и время. Применим для этой цели элемент **Label**.

13. Добавьте надпись к форме **Form1** и измените некоторые ее параметры.

13.1. Поместите указатель мыши на элемент Label на панели элементов, но не щелкайте на нем. Появится подсказка, содержащая название элемента управления, - Label. С помощью такого приема можно определять элементы VB из набора, представленного на панели элементов управления.

13.2. Щелкните на элементе Label, чтобы выбрать его.

13.3. Переместите указатель мыши в окно Form1. Обратите внимание, что указатель при этом принял вид перекрестья.

13.4. Поместите перекрестье в левую верхнюю часть окна и нажмите кнопку мыши. Не отпуская ее, перетащите указатель вниз и направо, чтобы надпись стала похожа на ту, которая изображена на следующем рисунке (Рисунок 6).

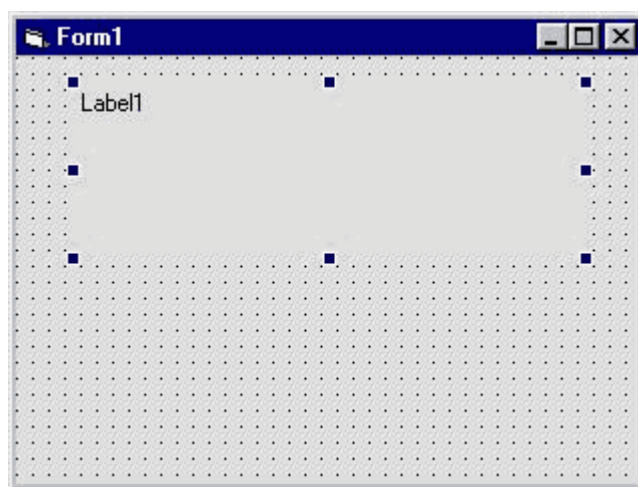



Рисунок 6 - Добавление надписи к форме

13.5.Посмотрите на Вашу надпись, которую VB автоматически назвал Label1. Если надо изменить ее положение или размеры, передвиньте ее (взяв за центр) или ее границы (взяв за один из черных квадратиков).

13.6.Сделайте активным окно свойств, щелкнув на его панели заголовка. Если окно свойств не отображено на экране, то следует на панели инструментов щелкнуть на кнопке  **Properties Window** (Окно свойств) либо использовать «горячую клавишу» - F4.

13.7.Поскольку выбрана надпись, ее имя (Label1) появится на строке объекта окна свойств, а ее свойства – под этой строкой.

13.8.Для установки шрифта, размера и эффектов надписи используется свойство **Font** (шрифт). Выберите это свойство, щелкнув на нем.

13.8.1. Обратите внимание, справа от параметра Font появился пропуск (кнопка с изображением "..."), который показывает, что для установки параметра используется диалоговое окно.

13.8.2. Щелкните на пропуске, чтобы открыть диалоговое окно Font. Текущие значения относятся к надписи Label1.

13.8.3. Посмотрите на текущий шрифт MS Sans Serif. Щелкните на MS Serif, чтобы изменить шрифт на MS Serif. В окне Sample (Пример) появится образец текста с текущими параметрами.

13.8.4. Посмотрите на текущее значение размера шрифта (8). Увеличьте его до 24 и посмотрите на изменение в окне Sample.

13.8.5. Щелкните на ОК, чтобы применить новые значения свойств к тексту Label1.

13.9.Установите значения свойств **Label1**:

13.9.1. **Alignment** (выравнивание): 2;

13.9.2. **Caption** (заголовок): Текущая дата и время;

13.9.3. **Name** (имя): lblTime.

13.10. Если надо, увеличьте с помощью мыши высоту и ширину

Вашей надписи, пока она не будет выглядеть, как показано на приведенном ниже рисунке (Рисунок 7).

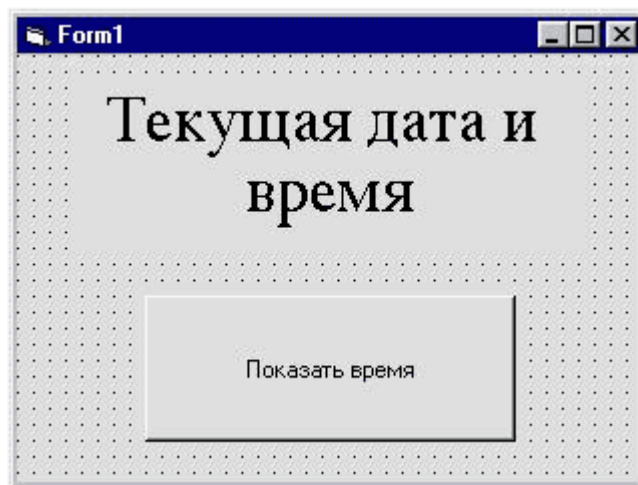


Рисунок 7 - Итоговый интерфейс

14. Добавьте к **Form1** кнопку. Когда Вы запустите Вашу программу, часы будут показывать текущее время и дату при нажатии на эту кнопку. Для этого щелкните на элементе **Command Button** (Кнопка) на панели элементов, чтобы выбрать его.

15. Поместите кнопку в окне **Form1**, чтобы получилась кнопка, подобная изображенной (Рисунок 7). Если требуется, передвиньте Вашу кнопку и поменяйте ее размеры.

16. Установите свойства для Вашей кнопки:


1.1 **Caption**: Показать время;

1.2 **Name**: cmdTime.

17. Сохраните проект.

Вы закончили выполнение шагов разработки проекта: планирование пользовательского интерфейса и установку значений свойств.


1.2.1 Разработка программы


18. Посмотрите на поведение созданных элементов управления до описания обработчиков событий для них. Для запуска проекта выберите в строке меню **Run** (Исполнить), **Start** (Начать) или щелкните на кнопке  **Start**

панели инструментов.

19. Посмотрите на экран. Панель элементов управления и окно свойств не отображаются. Обратите внимание, что в строке заголовка окна Microsoft Visual Basic в квадратных скобках появилось слово **[run]**, которое говорит о том, что сейчас установлен режим исполнения, а не режим разработки **[design]**, который был до этого.

20. Щелкните на кнопке "Показать время". Она нажимается и автоматически отжимается, но больше ничего не происходит, поскольку Вы не задали обработчик события для данной кнопки, т.е. не сообщили, что должно произойти при нажатии на эту кнопку.

21. Прекратите работу Вашей прикладной программы. Для этого щелкните на кнопке  **End** (Закончить) панели инструментов. Вы возвратились в режим разработки (о чем свидетельствует слово **design** на панели заголовка окна Microsoft Visual Basic).

22. Щелкните дважды на кнопке "Показать время", чтобы отобразилось окно кода (Рисунок 8). Окно кода можно отобразить другим способом. Для этого следует щелкнуть на кнопке  **View Code** (Показать код) на панели инструментов окна проекта.

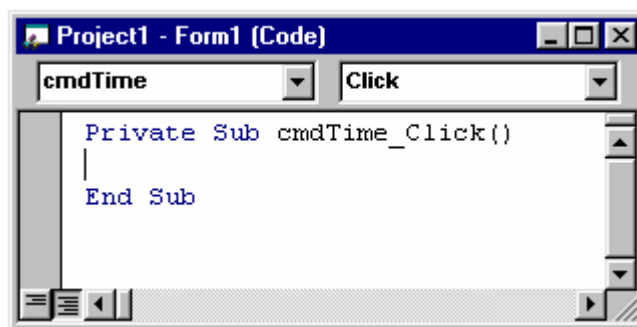


Рисунок 8 - Текст программы кнопки

23. Посмотрите на окно кода Form1. При выполнении двойного щелчка на кнопке среда разработки автоматически создала заготовку исходного кода обработчика события «щелчок кнопки мыши» для командной кнопки. Она начинается с заголовка, включающего слова **Private Sub**

(Локальная процедура), и заканчивается предложением **End Sub** (Конец процедуры). Между этими предложениями Вы можете вставить текст программы, описывающий последовательность действий, которую необходимо выполнять при получении сообщения о событии.

Для каждого события, которое может произойти с объектом (в данном случае с кнопкой), в системе VB предусмотрена соответствующая событийная процедура. В качестве примера перечислим некоторые события, которые могут произойти с кнопкой⁴:

- **Click** – пользователь щелкнул на кнопке управления;
- **MouseMove** – пользователь перемещает указатель мыши на кнопку;
- **MouseDown** – указатель мыши показывает на кнопку, и пользователь нажимает левую кнопку мыши;
- **MouseUp** – указатель мыши показывает на кнопку, и пользователь отпускает левую кнопку мыши.

События могут происходить практически с каждым объектом пользовательского интерфейса – формами, меню, текстовыми окнами, окнами рисунков и т.д. Вы должны снабдить все объекты процедурами для обработки каждого события, на которое они должны реагировать. Например, чтобы заставить командную кнопку что-либо делать, когда пользователь щелкает на ней, нужно вставить программный текст в процедуру с именем **Click**. Если же Вы хотите, чтобы происходило какое-либо действие, когда пользователь просто указывает на кнопку, следует вставить программный текст в ее процедуру **MouseMove**.

24. Запрограммируйте кнопку **cmdTime** так, чтобы при щелчке на этой кнопке на надписи выводилась текущая дата и время. Окно текста программы должно быть активно, а в нем – показана процедура **cmdTime_Click**. Если это не так, дважды щелкните по кнопке **cmdTime**. Между заголовком процедуры и конечной строкой введите инструкцию,

⁴ Более полный список событий элементов управления приведен в п. 9

предварительно установив для удобочитаемости отступ от начала строки в виде одного символа табуляции:

```
LblTime.Caption = Now
```

Вот и вся программа, которая состоит всего из одной инструкции. Эта инструкция делает результаты функции `Now` значением свойства ***Caption*** надписи ***lblTime***. Стандартная функция `Now` возвращает текущее время и дату. Свойство ***Caption*** определяет текст, который появляется на средстве управления (в данном случае – на надписи). Таким образом, когда эта строка программы выполнится, на надписи появятся текущие дата и время.

25. Закройте окно текста процедуры.

26. Сохраните проект (выберите ***File, Save Project*** или щелкните на кнопке ***Save Project*** панели инструментов).

27. Запустите проект. Несколько раз щелкните по кнопке "Показать время". Каждый раз, когда Вы щелкаете, текущее время и дата появляются на надписи ***lblTime***, как показано на рисунке (Рисунок 9).

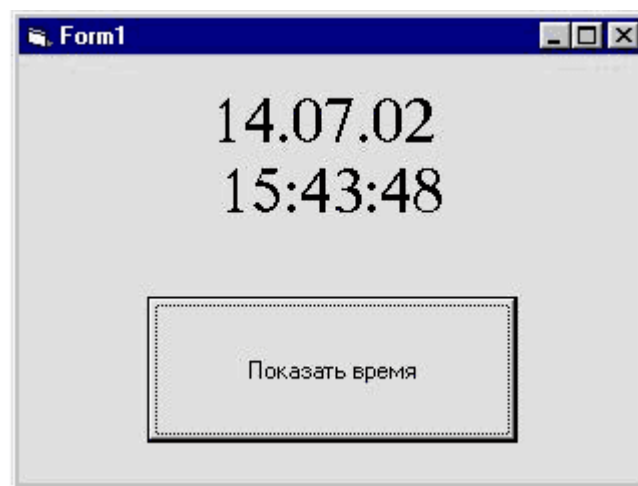


Рисунок 9 - Работающие цифровые часы

Попробуйте с помощью клавиатуры изменить показание времени. У Вас ничего не получится. Запомните, изменить значение свойства ***Caption*** надписи можно или в окне свойств в режиме разработки программы, или с помощью выполнения соответствующей инструкции (в нашем примере – это инструкция `LblTime.Caption = Now`) в режиме выполнения программы, но

нельзя изменить с помощью клавиатуры в режиме выполнения.

28. Завершите работу программы.

Начинающие разработчики часто допускают ошибку, забывая на этапе задания свойств переименовать объект, но в программном коде используют новое имя объекта. Например, такая ошибка будет сделана, если при выполнении пункта 13.9 данного задания, не переименовать надпись **Label1**, а в программном коде использовать имя LblTime. В этом случае окажется, что в программном коде будет применено имя несуществующего объекта, что приведет к ошибке № 424 времени выполнения (Рисунок 10).

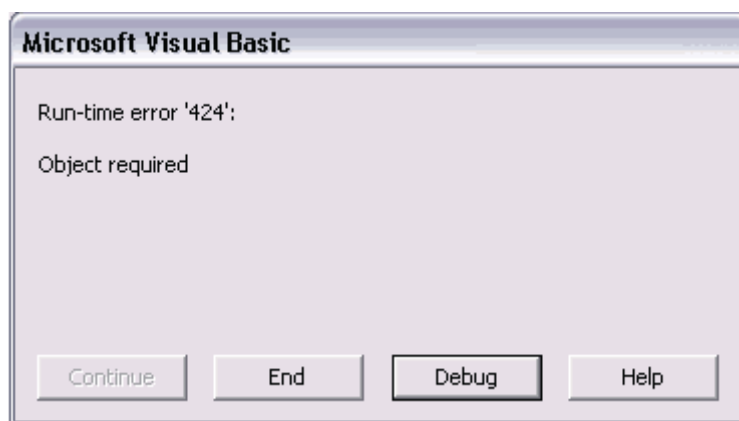


Рисунок 10 - Пример сообщения об ошибке времени выполнения

При появлении такого сообщения об ошибке времени выполнения следует щелкнуть в окне сообщения на кнопке **Debug** (Отладка). Проект перейдет в состояние отладки. Инструкция, в которой обнаружена ошибка будет выделена желтым цветом.

Ознакомьтесь с полным перечнем кодов ошибок представленным в п. 4.

1.3 Усовершенствование проекта

В разработанном проекте есть один недостаток: часы обновляют показания текущей даты и времени только после щелчка на кнопке.

Сделайте так, чтобы показания часов обновлялись через определенные промежутки времени, и чтобы не требовалось щелкать на

кнопке для обновления их показаний:

1. Найдите на панели элементов управления **Timer** (он изображен в виде маленького секундомера) и дважды щелкните на нем. Тем самым Вы указали VB добавить элемент **Timer** к форме **Form1**. Обратите внимание, VB автоматически нарисовал средство, выбрав размер по умолчанию. Таймер позволит Вам выполнять программный код через определенные промежутки времени. Вы можете передвинуть это средство управления, изменить его размер (где Вы поместите таймер, не играет роли, поскольку он не будет виден при выполнении программы).

2. Измените значение свойства **Interval** таймера на 1000 (время в миллисекундах). Если потребуется, щелкните на кнопке **Properties Window**, чтобы сделать активным окно свойств. Интервал 1000 означает, что таймер будет запускать процедуру через каждую секунду.

3. В окне **Form1** щелкните дважды на таймере, чтобы создать обработчик события для него – процедуру `Timer1_Timer`.

В окне программы появится исходный код процедуры обработчика события **Timer** объекта **Timer1** (Рисунок 11).

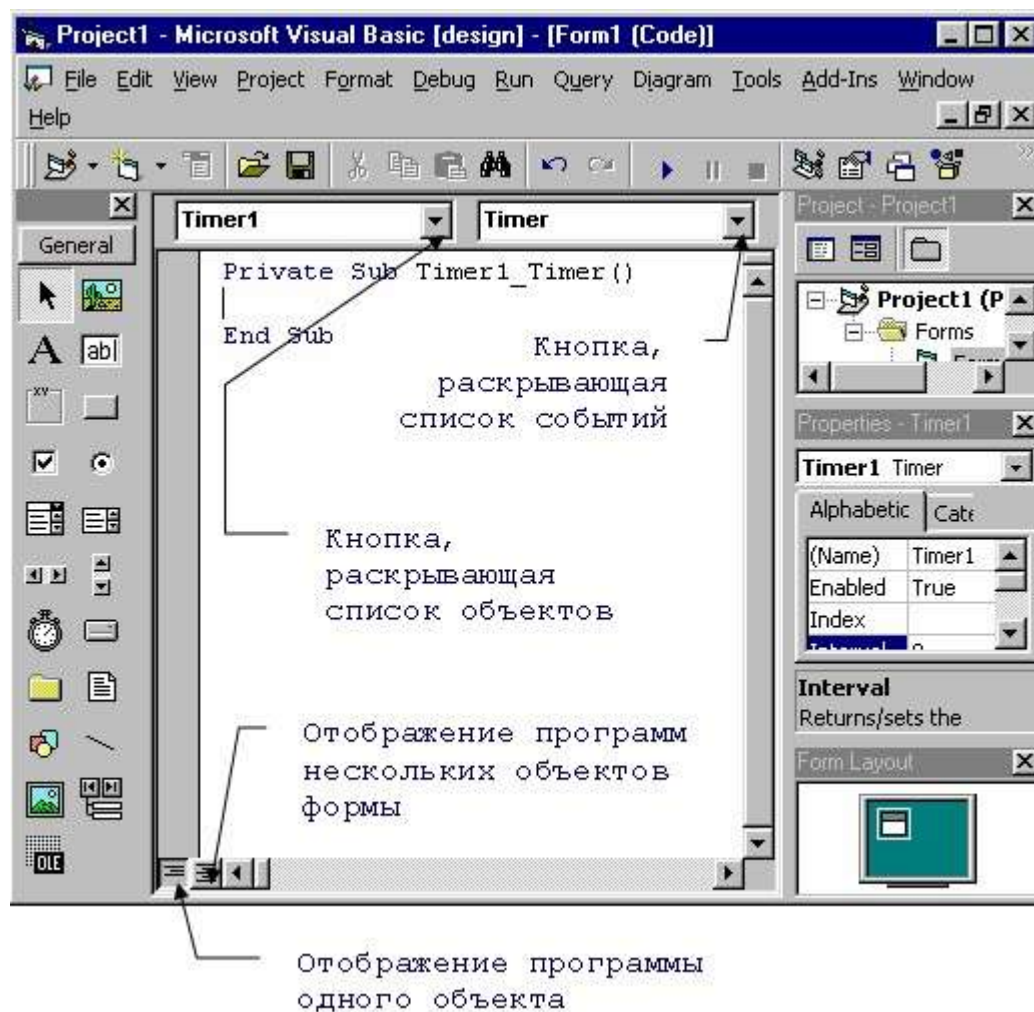


Рисунок 11 – Исходный код обработчика события

Для того чтобы видеть в окне программы одновременно программные тексты для различных объектов, щелкните на расположенной в нижней части окна кода кнопке отображения программ нескольких объектов формы. В окне программы рядом с текстом процедуры Timer1_Timer появится текст процедуры CmdTimer_Click.

4. Теперь, когда у Вас есть таймер, исправляющий значение часов каждую секунду, Вам больше не нужна кнопка. Скопируем исходный код обработчика события кнопки "Показать время" в процедуру таймера.

5. Выделите строку текста с функцией Now, которая выводит текущее время на надпись. Чтобы выбрать целую строку текста, переместите указатель мыши к началу строки, пока он не превратится в стрелку, затем щелкните.

6. Выберите в строке меню **Edit**, **Copy**, чтобы скопировать выделенный текст в буфер обмена.

7. Переместите курсор мыши в начало пустой строки в середине подпрограммы Timer1_Timer. Нажмите клавишу **Home**, убедитесь, что указатель вставки находится в самом начале строки. Это место, куда Вы вставите текст, который только что поместили в буфер обмена.

8. Выберите в строке меню **Edit**, **Paste**, чтобы вставить текст из буфера обмена в то место, куда показывает указатель вставки. Ваше окно программы теперь должно содержать текст, показанный ниже.

```
Private Sub CmdTime_Click()  
    Label1.Caption = Now  
End Sub
```

```
Private Sub Timer1_Timer()  
    Label1.Caption = Now  
End Sub
```

9. Сейчас следует удалить исходный код процедуры обработчика события CmdTime_Click. Ваше окно программы теперь должно содержать текст:

```
Private Sub Timer1_Timer()  
    Label1.Caption = Now  
End Sub
```

10. Закройте окно программы.

11. В окне формы Form1 выберите кнопку "Показать время" и нажмите клавишу **Del**, чтобы удалить ее. Эта кнопка больше не нужна, поскольку таймер автоматически корректирует показания времени и даты каждую секунду.

12. Выберите **Form1** в строке объектов окна свойств. Затем измените следующие свойства формы:

12.1.Caption: Часы Ваша фамилия. Например, Часы Иванова И.И.

12.2.BorderStyle: 3 - Fixed Dialog (просто введите 3 и нажмите клавишу Enter).

Caption – это текст, который отображается в заголовке окна.

Свойство **BorderStyle** определяет стиль обрамления окна формы **Form1**, могут ли размеры окна быть изменены в режиме выполнения, а также будут ли появляться кнопки **Maximize** (увеличение окна) и **Minimize** (восстановление окна). Заданное Вами значение 3 – FixedDialog определяет окно неизменяемых размеров со сплошной границей без кнопок **Maximize** и **Minimize**.

13. Выберите **File, Save Project**, чтобы сохранить изменения, внесенные во все файлы Вашего проекта. То же самое можно сделать, щелкнув на кнопке **Save Project** сохранения проекта на панели инструментов.

14. Запустите проект. Сравните Ваши часы с часами на рисунке (Рисунок 12). Благодаря таймеру (который исчез из виду, как только Вы запустили проект) показания времени теперь меняются каждую секунду. Завершите работу прикладной программы.

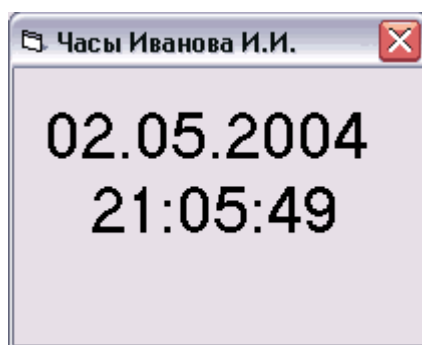


Рисунок 12 - Окончательный вид часов

1.4 Компиляция проекта

Чтобы можно было запускать разработанную Вами на VB программу на компьютере с Windows без установленного VB, необходимо создать выполняемый файл (с расширением .exe). Такой файл может быть получен в результате компиляции проекта.

Скомпилируйте только что разработанный проект цифровых часов.

15. Выберите **File, Make myclock.exe**, чтобы открыть диалоговое окно, в котором Вы можете выбрать папку назначения Вашего исполняемого файла.

16. Поместите исполняемый файл в рабочую папку.

17. Щелкните на **ОК**. VB создаст исполняемый файл.

18. Проверьте работу файла myclock.exe, который Вы только что создали.

19. Чтобы завершить работу VB, выберите **File, Exit**.

20. Запустите **Проводник**, и выберите файл myclock.exe и дважды щелкните на нем, чтобы запустить эту программу. Обратите внимание, программа работает без VB.

21. Завершите работу программы.

Разработанный Вами проект потребуется при выполнении задания следующей лабораторной работы.

1.5 Отображение окон в среде VB

В среде VB используются различные окна. С некоторыми из них Вы познакомились в п. 1.1.1. Иногда по ошибке может быть закрыто одно из окон среды VB. Поэтому нужно уметь восстанавливать отображение нужных окон. Наиболее универсальным способом является выполнение соответствующей команды пункта меню View. Но чаще оказывается удобнее пользоваться кнопками панели инструментов (Рисунок 13).

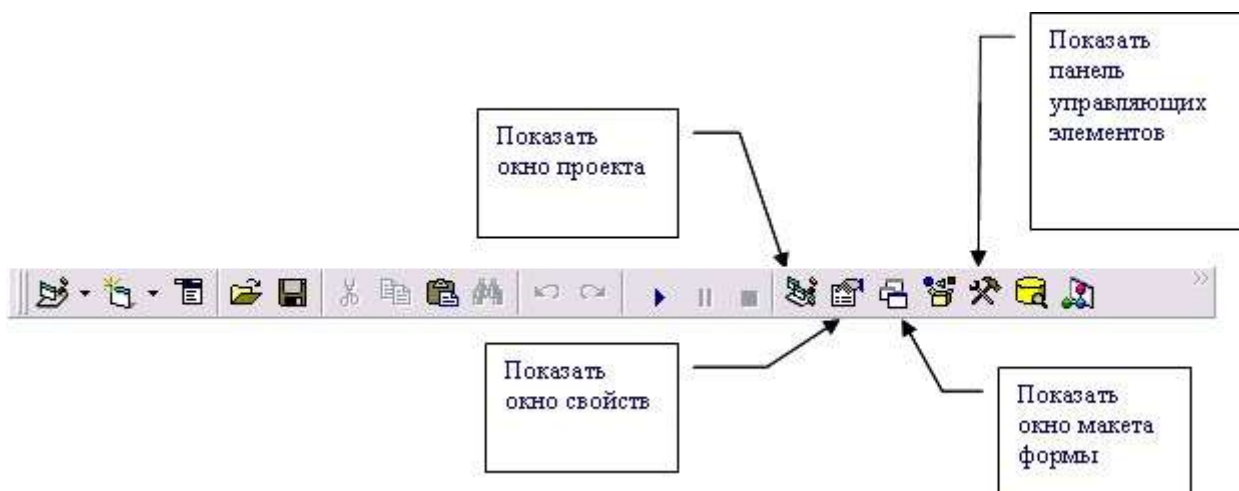


Рисунок 13 - Управление отображением окон среды VB

Окно может быть соединено к другому окну. Для этого нужно дважды щелкнуть на заголовке окна, которое требуется присоединить.

Окно можно отсоединить от другого окна. Для этого нужно установить указатель мыши на заголовок окна, нажать левую кнопку мыши и переместить окно на новое место.

1.6 Вопросы для контроля

– Найдите в окне VB следующие элементы интерфейса и поясните их назначение:

- строка меню;
- панель элементов;
- панель элементов управления;
- окно проекта;
- окно конструктора форм;
- форма;
- окно свойств;
- окно макета формы.

– Как выполнить запуск системы VB и выход из неё?

– Шаги разработки проекта.

– Как отобразить на экране и как закрыть:

- окно свойств;
- окно проекта;
- панель элементов управления?

– Поясните назначение свойств: *Alignment*, *Caption*, *Font*, *Height*, *Width*.

- Как вставить в форму надпись, кнопку?
- Как сохранить проект?
- Как запустить проект и как прекратить его работу?
- Как узнать, находится система VB в режиме разработки или в режиме исполнения?
- Как отобразить исходный код процедуры обработчика события для объекта?
- Как увидеть список событийных процедур, связанных с объектом?
- Как увидеть список объектов, включенных в форму?
- Можно ли изменять значения свойства *Caption* надписи при исполнении проекта?
- Поясните назначение свойства *Interval* таймера.
- Как создать выполняемый файл для разработанного проекта?
- Как запустить выполняемый файл?

2 Переменные и константы

При выполнении вычислений и других операций обработки информации возникает необходимость запоминать некоторые промежуточные результаты. Для каждого из этих результатов выделяется область памяти компьютера, состоящая из одного или нескольких байт. Для того чтобы данные области памяти можно было различать, а также иметь возможность сослаться на ее содержимое, ей сопоставляют имя.

Пара, включающая область памяти доступную во время исполнения, как для чтения, так и для записи данных, и сопоставленное ей имя, называется переменной.

Переменная – это в некотором смысле ячейка для хранения информации, например, числа, строки символов. При этом имеется возможность неоднократно считывать значение переменной, а также возможность записывать в эту ячейку другое значение. Переменная может изменять свое значение в процессе выполнения программы.

Имена переменных в VB должны удовлетворять следующим требованиям:

- начинаться с буквы;
- включать только буквы, цифры, символ подчеркивания (), который на клавиатуре находится под тире (-);
- содержать не более 255 символов.

Обратите внимание, имя не может содержать пробел (), точку (.), запятую (,), восклицательный знак (!) или символы (@), (&), (\$), (#). Не следует использовать имена, совпадающие с ключевыми (зарезервированными) словами языка.

В языке VB не различаются строчные и прописные буквы.

Множество возможных значений переменной, допустимые операции, которые к ней применимы, количество байтов, отведенных для нее, определяется типом переменной. В каждом языке программирования определена своя система типов переменных.

К константам относится все то, что сказано выше о переменных. Но есть одно существенное отличие: константа не может изменить значение при выполнении программы. Переменные и константы – это данные.

2.1 Основные типы данных VB

Познакомимся с системой типов данных VB:

Таблица 1 - Основные типы данных VB

Тип данных	Размер ячейки в байтах	Диапазон значений
Boolean (логический)	2	True или False
Byte (байт)	1	От 0 до 255
Integer (целый)	2	От -32 768 до 32 767
Long (длинный целый)	4	От -2 147 483 648 до 2 147 483 647
Single (с плавающей точкой обычной точности)	4	Абсолютное значение находится в диапазоне от $1,401298E-45$ до $3,402823E38^5$
Double (с плавающей точкой двойной точности)	8	Абсолютное значение находится в диапазоне от $4,94065645841247E-324$ до $1,79769313486232E308$
Currency (денежный)	8	От -922 337 203 685 477,5808 до 922 337 203 685 477,5807
Date (дата/время)	8	От 1 января 100 г. до 31 декабря 9999 г.

⁵ Под записью (Число1EЦелоеЧисло2) следует понимать Число1, умноженное на 10, возведенное в степень ЦелоеЧисло2.

Тип данных	Размер ячейки в байтах	Диапазон значений
Object (объект)	4	Любой указатель объекта
String (строка переменной длины)	10 байт + длина строки	Длина строки от 0 до приблизительно 2 миллиардов символов
Variant (универсальный тип)	16 байт+ длина строки	Дата/время; числовые значения с тем же диапазоном, что и для Double ; строки с тем же диапазоном, что и для String
Type (тип, определяемый пользователем)	Определяется компонентами	Диапазон каждого элемента определяется его типом данных

Если, например, в программе имеется переменная типа **Integer** с именем КоличествоСтудентов, то где-нибудь в программе можно написать:

```
КоличествоСтудентов = 1000
```

и для всех инструкций, которые появятся после этого, переменная КоличествоСтудентов будет всегда равна числу 1000 – пока, возможно, не появится инструкция

```
КоличествоСтудентов = 1050
```

КоличествоСтудентов в этой инструкции является именем переменной, а 1000 в этом примере – её значением.

Инструкцию КоличествоСтудентов = 1000 называют оператором присваивания: переменная КоличествоСтудентов получает значение, равное 1000.

Эту инструкцию нельзя понимать в том смысле, что левая часть равна правой части. Оператор присваивания – это действие, заключающееся в том, что значение правой части записывается в ячейку памяти, отведенную для хранения значения переменной, имя которой указано слева от знака

равенства в инструкции присваивания. Будет ошибкой записать эту инструкцию так: 1000 = КоличествоСтудентов

Слева от знака равенства должна находиться переменная.

Переменные типов **Byte**, **Integer**, **Long**, **Single**, **Double**, **Currency** принимают числовые значения.

Если в программе имеется переменная а типа **Single**, то такой переменной можно присвоить числовое значение с дробной частью, например: a = – 62.697

В этой инструкции присваивания справа от знака равенства находится константа с плавающей точкой. В VB для отделения целой части от дробной части применяется символ (.). Возможен другой способ записи констант с плавающей точкой – с порядком. Например, 1.5E–16 означает $1.5 \cdot 10^{-16}$ (или иначе 0.000000000000000015).

К числовым переменным можно применять арифметические операции сложения (+), вычитания (–), умножения (*), деления (/), возведения в степень (^).

Значением переменной типа **String** может быть символ или строка символов.

Значением переменной типа **Date** может быть, дата, время или дата и время.

Переменная типа **Boolean** может принимать всего два значения. Такая переменная может иметь значение **True** (истина) или значение **False** (ложь).

Тип **Variant** является универсальным. Переменные типа **Variant** могут принимать числовые значения, значения символов и строк символов, значение даты, времени и даты и времени. Остальные типы данных пока комментировать не будем.

Если UserName является переменной типа **String**, тогда можно написать: UserName = "Иван" (здесь очень важны кавычки, так как иначе VB может принять Иван за имя переменной). Этот пример показывает, что константа типа **String** должна быть заключена в двойные кавычки.

К строковым переменным и константам можно применять операцию сцепления, которая обозначается символом (&) или символом (+). Например, можно написать:

```
UserName = UserName & " Иванов"
```

После выполнения этой инструкции переменная UserName будет иметь значение "Иван Иванов".

Если ДеньРождения и EndOfTime являются именами переменных типа **Date**, тогда можно записать:

```
ДеньРождения = #29.10.1970#
```

```
EndOfTime = #8:30#
```

(как символы выделяют кавычками, так дату или время выделяют символом (#)). Дату и время можно поместить в одну переменную. Для переменной DateAndTime типа **Date** может быть записано:

```
DateAndTime = #13.2.2000 11:30#
```

2.2 Структура проекта

Проект обычно включает одну или несколько форм, а также может включать модули. С формой Вы уже имели дело при выполнении предыдущих заданий. Форма при выполнении проекта отображается на экране монитора в виде окна. Она может содержать объекты (надписи, кнопки и т.д.). Модуль отличается от формы тем, что он никак не отображается на экране монитора при выполнении проекта и не может содержать объекты.

Каждая форма и каждый модуль проекта имеют главную секцию (**General**), в которой может быть записан программный код.

В главной секции могут быть объявлены константы и переменные.

Наряду с главной секцией форма и модуль могут включать секции, каждая из которых является объявлением процедуры или функции.

2.3 Объявление переменных и констант

Если переменная или константа применяется в проекте, она должна быть объявлена. Существуют следующие уровни объявления переменных и констант:

- уровень процедуры (функции). Имя, объявленное в процедуре, действует только внутри этой процедуры и не действует вне этой процедуры;
- уровень формы. Имя, объявленное в главной секции формы, действует во всех процедурах этой формы, но не действует в других формах и модулях;
- уровень проекта. Имя, объявленное в главной секции модуля с предваряющим словом **Public** (Общий), действует во всех формах и модулях проекта.

Надо иметь в виду, если при объявлении переменной тип явно не указан, то будет назначен тип **Variant**. Такой способ объявления называется объявлением по умолчанию.

При объявлении константы необходимо задать ее имя, тип, область действия и значение.

Синтаксис объявления константы:

[{**Public****Private**}] **Const** ИмяКонстанты [**As** Тип] = Значение

В подобных определениях синтаксиса прямоугольные скобки [] означают, что конструкция, находящаяся внутри этих скобок, не обязательна. Символ (|) означает, что должно быть выбрано одно из слов, между которыми он поставлен. Сами же символы ([, |), (|) в текст объявления не включаются.

Приведенное определение синтаксиса означает, что объявление константы начинается с обязательного слова **Const** (Константа). Перед **Const** может стоять одно из слов: **Private** (Локальный) или **Public** (Общий), задающих область действия константы. Затем следует имя константы. После имени может стоять слово **As** и наименование типа. Затем следует знак равенства и значение константы.

В определении подчеркнуто наименование той области действия, которая может быть задана по умолчанию.

В следующем примере в главной секции модуля описывается глобальная константа Age целого типа, и ей присваивается значение 54.

```
Public Const Age As Integer = 54
```

Допускается также описание нескольких констант в одной строке. В этом случае, чтобы задать тип данных, надо указать название типа для каждой константы.

В следующем примере описываются локальные константы Ag и Wg как **Single**.

```
Const Ag As Single = 3.14, Wg As Single = 2.78
```

Синтаксис объявления переменной:

```
{Static|Public|Private|Dim} ИмяПеременной [As Тип]
```

Зарезервированное слово **Dim** (Размерность) при объявлении переменных применяется чаще всего.

Статические переменные, описанные на уровне процедуры со словом **Static** вместо слова **Dim**, сохраняют свои значения даже после выхода из процедуры при повторном входе в эту процедуру.

Вот пример объявления переменной типа строки символов:

```
Dim strName As String
```

Если не указать тип переменной, ей будет присвоен тип **Variant**. При этом следует учитывать, что переменные типа **Variant** занимают больше места в памяти и обращение к ним происходит медленнее.

В одной строке можно объявить несколько переменных, но при этом следует для каждой указывать имя типа:

```
Dim a As Integer, b As Integer, c As Long
```

```
Dim e As Integer, f, g
```

В первой строке объявлены две переменные типа **Integer** и одна переменная типа **Long**. Во второй строке – одна переменная типа **Integer**, а две другие – типа **Variant** по умолчанию.

При объявлении переменная не получает значения, ей только отводится память, а ее значение не определено.

Можно вообще не объявлять переменные. Однако такая практика является источником ошибок, и ее надо избегать. Чтобы VB расценивал любую необъявленную переменную формы или модуля как ошибочную, в главной секции формы или модуля первой должна следовать инструкция **Option Explicit**. Эта инструкция налагает требование явного описания всех переменных этой формы или модуля. Если форма содержит инструкцию **Option Explicit**, при попытке использования неопisanного или неверно введенного имени переменной при запуске проекта возникает сообщение об ошибке.

Можно так настроить среду VB, что инструкция **Option Explicit** будет автоматически помещаться системой VB в главной секции создаваемой формы или модуля. Для этого надо выполнить команды меню **Tools** (Инструменты), затем **Options** (Свойства), затем на вкладке **Editor** (Редактор) установить флажок в окошке **Require Variable Declaration** (требовать объявления переменных) и, наконец, щелкнуть на кнопке **OK**. Но это следует сделать до создания формы или модуля.

2.4 Преобразование и совместимость типов

В этом разделе речь будет идти о тех ситуациях, когда при выполнении операции или инструкции участвуют данные разных типов. Например, что произойдет, если переменной целого типа присваивается числовое значение типа Single.

В отношении типов VB проявляет максимальную терпимость – когда это возможно, преобразование одного типа в другой будет выполнено автоматически и программисту нет нужды об этом заботиться. При этом соблюдаются перечисленные ниже правила.

При преобразовании числа с плавающей точкой в целое происходит округление до ближайшего целого.

При преобразовании целого числа в число с плавающей точкой дробная часть принимается равной нулю.

В случае преобразования целого типа в другой целый тип возможна ситуация, когда целый тип с большим диапазоном значений преобразуется в целый тип с меньшим диапазоном значений. Если значение, присваиваемое «короткому» типу, выйдет за пределы диапазона его допустимых значений, произойдет ошибка времени исполнения.

Строковые и числовые типы совместимы. Можно присвоить числовое значение строковой переменной и наоборот.

Например, не возникнет ошибки времени исполнения при присваивании свойству *Caption*, которое имеет тип String, значения 125 переменной типа Integer. Произойдет преобразование целого числового значения 125 в строку символов "125".

И наоборот. При присваивании переменной целого типа значения свойства *Text* некоторого текстового окна, которое имеет тип String (пусть, например, в этом текстовом окне набрано три символа 125), произойдет преобразование строки символов "125" в целое число и целочисленная переменная получит значение 125. Однако строка символов должна быть такой, чтобы она могла трактоваться как число.

Возможно также выполнение арифметической операции, например умножения (*), когда один или оба операнды являются строкой символов, но при условии, что их значения, можно интерпретировать как числа. При этом особую осторожность следует соблюдать с операцией (+), которая в зависимости от контекста может означать либо операцию сложения, либо операцию сцепления.

В операции операнд1 + операнд2 символ (+) VB будет воспринимать как:

- операцию сложения, если оба операнда имеют какой-либо из числовых типов;
- операцию сцепления, если один операнд имеет числовой тип, а

второй является строкой символов, значение которой может быть интерпретировано как число;

- операцию сцепления, если один операнд имеет числовой тип, а второй является строкой символов, значение которой не может быть интерпретировано как число;

- операцию сцепления, если оба операнда имеют строковый тип независимо от их значений.

Теперь можно приступить к разработке проекта.

2.5 Разработка проекта

Создайте рабочую папку проекта.

Запустите VB. Ознакомьтесь с окном проекта.

Если появилось окно *New Project*, то выберите *Standart EXE* (стандартный) вкладки *New* (новый) и щелкните на кнопке *Открыть*.

Поместите на открывшейся форме шесть текстовых окон, три надписи и три кнопки (Рисунок 14).

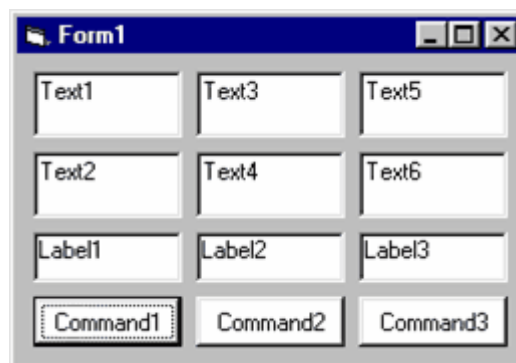


Рисунок 14 - Вид формы

Щелкните дважды на форме Form1. В открывшемся окне кода *Project1 – Form1 (Code)* раскройте список объектов и выберите главную секцию (*General*).

Введите в главной секции код, указанный далее.

Option Explicit

'Требование явного объявления переменных

```
'в пределах формы Form1
Dim i As Integer
'Объявлена переменная i целого типа
Dim r As Single
'Объявлена переменная r с плавающей точкой
'обычной точности
Dim st1 As String, st2 As String
'Объявлены переменные st1 и st2 строкового типа
```

На примере этого фрагмента программного кода можно увидеть, как документируется программа с помощью комментариев. Все, что находится в строке правее символа (') рассматривается как комментарий, присутствующий в программном коде, но не оказывающий влияния на результаты его выполнения.

Сохраните проект в своей рабочей папке.

Раскройте список объектов и выберите объект **Command1**. В окне кода появилась заготовка процедуры **Command1_Click**.

Задайте программный код:

```
Private Sub Command1_Click()
    i = Text1.Text
    r = Text2.Text
    Label1.Caption = r + i
End Sub
```

Щелкните на кнопке **Start** для запуска проекта. В появившемся окне приложения **Form1** наберите в поле **Text1** значение 2 (предварительно удалив строку Text1), наберите в поле **Text2** значение -1,3 (предварительно удалив строку Text2). После этого щелкните на кнопке **Command1**. Последнее событие приведет к выполнению инструкций процедуры **Command1_Click** и надпись отобразит результат $2+(-1.3)$. На следующем рисунке (Рисунок 15) показан вид окна приложения **Form1** после указанных действий.

Рисунок 15 - Результат счета

Обратите внимание, при вводе строки в окне *Text2*, которая интерпретируется программой как число с плавающей точкой, для разделения целой и дробной частей применена не точка, а запятая. Это связано с тем, что настройкой операционной системы в качестве разделителя целой и дробной части чаще всего установлена именно запятая. Вид разделителя можно изменить. Для этого необходимо выполнить команды *Настройка* и *Панель управления* меню кнопки *Пуск* панели задач. Затем следует выбрать *Язык и стандарты* и на вкладке *Числа* в окне *Разделитель дробной и целой частей числа* установить нужный разделитель и щелкнуть на кнопке *ОК*. Для защиты от недостаточно квалифицированных пользователей команда *Настройка* в учебных компьютерных классах может быть исключена из меню кнопки *Пуск*. Поэтому изменить разделитель целой и дробной частей числа может оказаться проблематично.

Итак, будем исходить из того, что при записи констант в программном коде следует применять точку для разделения целой и дробной частей, а во входном потоке при вводе значений переменных надо применять запятую.

Остановимся подробнее на существенных, но на первый взгляд не заметных, особенностях выполнения инструкций процедуры *Command1_Click*. При выполнении этих трех инструкций четыре раза осуществляется преобразование типа.

Так при выполнении оператора присваивания $i = \text{Text1.Text}$ тип *String*

значения свойства **Text** текстового окна **Text1** преобразуется в тип Integer переменной *i*.

При выполнении оператора присваивания `r = Text2.Text` тип String значения свойства **Text** текстового окна **Text2** преобразуется в тип Single переменной *r*.

В инструкции `Label1.Caption = r + i` при выполнении арифметической операции сложения тип Integer операнда *i* преобразуется к типу Single операнда *r*, так как арифметическая операция может быть выполнена над однотипными операндами, а тип с плавающей точкой старше целого типа.

Наконец, результат сложения типа Single преобразуется к типу String значения свойства **Caption** надписи **Label1**.

Остановите выполнение проекта, щелкнув на кнопке **End** панели инструментов, и сохраните проект.

Щелкните два раза на кнопке **Command2**. В окне кода появилась заготовка процедуры `Command2_Click`.

Задайте программный код:

```
Private Sub Command2_Click()  
    st1 = Text3.Text  
    st2 = Text4.Text  
    Label2.Caption = st1 + st2  
End Sub
```

Щелкните на кнопке **Start** для запуска проекта и в появившемся окне приложения **Form1** наберите в текстовом окне **Text3** строку «Вас» (предварительно удалив строку «Text3»), а также наберите в текстовом окне **Text4** строку «илий» (предварительно удалив строку «Text4»). После этого щелкните на кнопке **Command2**. Последнее событие приведет к выполнению инструкций процедуры `Command1_Click` и надпись отобразит результат «Василий» операции сцепления (Рисунок 16).

Рисунок 16 - Результат счета

Полученный результат Вас не должен удивлять. В инструкции `Label2.Caption = st1 + st2` символ операции (+) воспринимается как символ операции сцепления, а не операции сложения, поскольку оба операнда `st1` и `st2` имеют строковый тип. А что произойдет, если в текстовых окнах задать числа?

Удалите в текстовом окне *Text3* строку «Вас» и наберите «25», а также удалите в текстовом окне *Text4* строку «илий» и наберите «15». После этого щелкните на кнопке *Command2*. На приведенном ниже рисунке (Рисунок 17) показан результат.

Рисунок 17 - Результат счета

Опять произошло сцепление операндов. Это следовало ожидать, поскольку оба операнда (свойства *Text*) имеют строковый тип. Посмотрите, что произойдет, если один операнд будет иметь строковый, а другой – числовой, например, целый тип.

Остановите выполнение проекта, щелкнув на кнопке **End** панели

инструментов, и сохраните проект.

Щелкните два раза на кнопке **Command3**. В окне кода появилась заготовка процедуры Command3_Click.

Задайте программный код:

```
Private Sub Command3_Click()  
    st1 = Text5.Text  
    st2 = Text6.Text  
    Label3.Caption = 25 + st1 + st2  
End Sub
```

Щелкните на кнопке **Start** для запуска проекта и в появившемся окне приложения Form1 наберите 2 в текстовом окне **Text5** (предварительно удалив строку Text5), а также наберите 3 в текстовом окне **Text6** (предварительно удалив строку Text6). После этого щелкните на кнопке **Command3**. Последнее событие приведет к выполнению инструкций процедуры Command3_Click и надпись отобразит результат операции (Рисунок 18).

Рисунок 18 - Результат счета

При выполнении вычисления значения выражения $25 + st1 + st2$ первой выполняется операция $25 + st1$. В этой операции один операнд (константа 25) имеет числовой тип, а другой (строка st1) имеет значение, которое может быть интерпретировано как число. Поэтому здесь символ (+) воспринят как операция сложения. Результат операции сложения (27) тоже имеет числовой тип, поэтому символ (+) в следующей операции $27 + 3$ также

воспринят как операция сложения и, в конце концов, получен понятный результат 30.

Остановите выполнение проекта, щелкнув на кнопке **End** панели инструментов, и сохраните проект.

Щелкните два раза на кнопке **Command3**. В окне кода появился программный код процедуры Command3_Click.

Измените порядок следования операндов в инструкции присваивания Label3.Caption = 25 + st1 + st2. Замените ее инструкцией Label3.Caption = st1 + st2 + 25.

Щелкните на кнопке **Start** для запуска проекта и в появившемся окне приложения Form1 снова наберите 2 в текстовом окне **Text5** (предварительно удалив строку Text5), а также наберите 3 в текстовом окне **Text6** (предварительно удалив строку Text6). После этого щелкните на кнопке **Command3**. Надпись отобразит на первый взгляд неожиданный результат вычислений (Рисунок 19)

Рисунок 19 - Результат счета

Изменение порядка слагаемых привело к изменению значения результатов вычислений! Почему? Дело в том, что первой теперь выполняется операция st1 + st2, в которой символ (+) воспринимается как операция сцепления, поскольку оба операнда имеют строковый тип.

Результат операции соответственно тоже имеет строковый тип и равен "23".

При выполнении операции "23" + 25 символ (+) будет воспринят, как

операция сложения, поскольку один операнд (константа 25) имеет числовой тип, а другой операнд ("23") может быть интерпретирован как число.

Остановите выполнение проекта, щелкнув на кнопке **End** панели инструментов, и сохраните проект.

Покажите результат выполнения задания преподавателю.

Подведем итоги:

В VB предусмотрено несколько типов для переменных, принимающих числовые значения. Какой тип применять в каждом конкретном случае? Для ответа на этот вопрос надо учитывать следующее:

- Тип **Variant** является универсальным. Его применение делает программный код более компактным, так как этот тип можно не объявлять – он предусматривается по умолчанию. Однако каждое данное этого типа занимает в памяти больше места. А обращение к данным типа **Variant** происходит медленнее, чем к данным простых типов.

- Арифметические операции с данными целого типа (**Integer, Long**) выполняются быстрее, чем с данными с плавающей точкой (**Single, Double**).

- Арифметические операции с данными целого типа (**Integer, Long**) выполняются точно, а с данными с плавающей точкой (**Single, Double**) – приближенно.

В соответствии с этими соображениями не рекомендуется применять тип **Variant** в тех случаях, когда проект должен выполняться быстро, а также должен занимать мало места в памяти.

Целый тип рекомендуется применять, если данное может принимать только целые значения (например, количество участников или номер элемента некоторой последовательности).

Необходимо по возможности избегать преобразования типа. Неправильная трактовка этого действия приводит к трудно диагностируемым ошибкам.

2.6 Вопросы для контроля

- Что общего и чем различаются переменные и константы?
- Что определяет тип данного?
- Когда возникает необходимость преобразования типа данного?
- Поясните синтаксис объявления константы.
- Поясните синтаксис объявления переменной.
- Как определяется область действия данного?
- Какова структура проекта?
- Каковы результаты операций $2+3$, $2+"3"$, $"2" + "3"$?
- Как зависит от типа числовой переменной точность и скорость выполнения арифметической операции?

3 Ввод и вывод значений переменных

Для ввода значения переменной можно воспользоваться функцией с именем `InputBox`. Она выводит на экран диалоговое окно, содержащее сообщение и поле ввода, возвращает значение типа **String**, содержащее текст, введенный в поле ввода.

Упрощенный синтаксис обращения к функции:

```
InputBox(Сообщение[,Заголовок])
```

У этой функции первый слева аргумент обязательный, а все остальные – необязательные. Назначение аргументов:

- сообщение – выражение типа **String**, отображаемое в диалоговом окне;
- заголовок – выражение типа **String**, отображаемое в строке заголовка диалогового окна.

3.1 Ввод данных с помощью функции `InputBox`

Создайте рабочую папку, в которой будет храниться разрабатываемый Вами проект.

Запустите VB и сохраните проект в рабочей папке. При этом сохраните форму под именем **Form1**, проект - под именем **Project1**.

Разместите на форме **Form1** восемь объектов, как это изображено на рисунке (Рисунок 20).

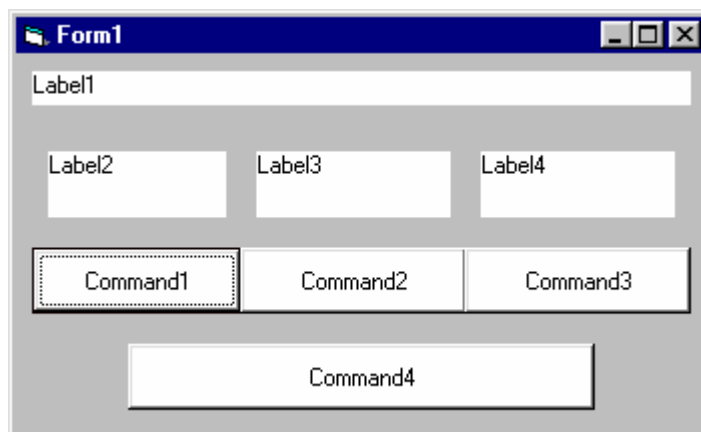
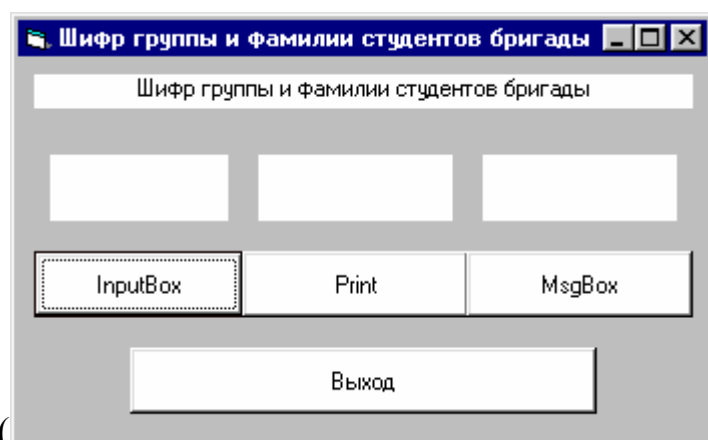


Рисунок 20 - Положение объектов

Здесь предполагается следующее назначение объектов:

- надпись **Label1** предназначена для размещения информации об исполнителях задания;
- **Label2, Label3, Label4** – отображают результаты;
- щелчок на кнопках **Command1, Command2, Command3** приведет к выполнению соответствующих событийных процедур;
- щелчок на кнопке **Command4** будет означать прекращение выполнения проекта.

Задайте значения свойств такими, чтобы форма приобрела вид, показанный



следующем рисунке (

Рисунок 21).

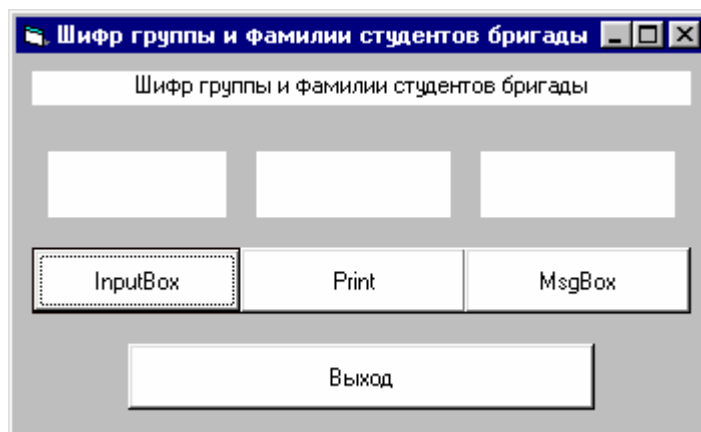


Рисунок 21 - Вид формы

Сохраните проект.

Дважды щелкните на кнопке **Выход**. В появившемся окне кода будет отображена процедура **Command4_Click**. Введите инструкцию **End** для

прекращения выполнения программы. После этого программный код процедуры должен выглядеть так:

```
Private Sub Command4_Click()  
    End  
End Sub
```

Дважды щелкните на кнопке **InputBox**. В появившемся окне кода будет отображена процедура Command1_Click.

Введите операторы для описания двух переменных и вызова функции InputBox:

```
Dim Подсказка, ПолноеИмя  
Подсказка = "Пожалуйста, введите Ваше имя"  
ПолноеИмя = InputBox(Подсказка)  
Label2.Caption = ПолноеИмя
```

Вы объявили с помощью инструкции Dim две переменные Подсказка и ПолноеИмя (тип переменных явно не указан, следовательно, объявлен по умолчанию, а по умолчанию VB назначает тип Variant). Во второй строке переменной Подсказка присваивается текстовая константа. В следующей строке вызывается функция InputBox. При выполнении эта функция выводит диалоговое окно с запросом-подсказкой на ввод данных пользователем (Рисунок 22).

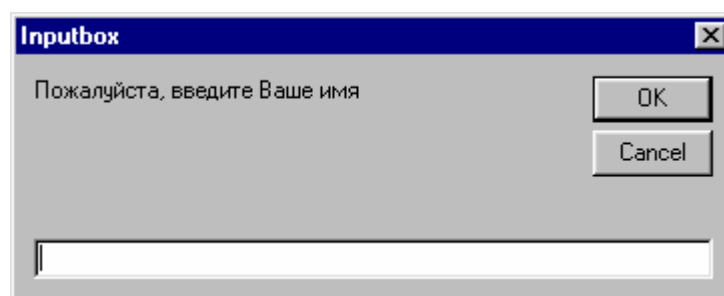


Рисунок 22 - Окно функции InputBox

Значение аргумента функции InputBox (в данном случае – переменной Подсказка) выведено в виде запроса-подсказки в окне функции InputBox. Значением функции InputBox будет строка символов, которую пользователь

наберет в поле окна функции InputBox. Это возвращаемое значение функцией InputBox в третьей строке кода присваивается переменной ПолноеИмя. Четвертый оператор значение переменной ПолноеИмя присваивает свойству *Caption* надписи *Label2*.

Сохраните проект.

Щелкните на кнопке **Start** (Пуск) на панели инструментов для запуска программы. Программа будет запущена.

Щелкните на кнопке *InputBox*. VB начнет выполнять процедуру Command1_Click, и на экране появится диалоговое окно изображенного выше запроса-подсказки.

Введите Ваше имя и отчество, затем щелкните на кнопке **OK**. Функция InputBox возвращает программе набранную Вами строку символов, значение которой присваивается переменной ПолноеИмя. Затем программа отображает значение переменной ПолноеИмя в поле надписи *Label2*.

Щелкните на кнопке **Выход** для остановки программы.

С помощью функции InputBox могут быть заданы значения числовым переменным. Дополните код процедуры Command1_Click:

```
Private Sub Command1_Click()  
    Dim Приглашение, ПолноеИмя  
    Dim i As Integer, s As Single  
    Приглашение = "Пожалуйста, введите Ваше имя!"  
    ПолноеИмя = InputBox(Приглашение)  
    Label2.Caption = ПолноеИмя  
    i = InputBox("Задайте значение i")  
    'В этой инструкции  
    'выполняется преобразование возвращаемой  
    'функцией InputBox строки символов  
    'в целое число  
    Label3.Caption = "i=" & i  
    s = InputBox("Задайте значение s")
```

```
'В этой инструкции  
'выполняется преобразование возвращаемой  
'функцией InputBox строки символов в число  
'с плавающей точкой обычной точности  
Label4.Caption = "s=" & s  
End Sub
```

Теперь в процедуре `Command1_Click` дополнительно объявлены переменная `i` целого типа и переменная `s` с плавающей точкой обычной точности. Этим переменным также задаются значения с помощью обращений к функции `InputBox`. Полученные переменными `i` и `s` значения затем отображаются в полях надписей **Label3** и **Label4** соответственно.

Сохраните проект.

Щелкните на кнопке **Start** (Пуск) на панели инструментов для запуска программы.

Щелкните на кнопке `InputBox`. VB начнет выполнять процедуру `Command1_Click`. Теперь диалоговое окно запроса-подсказки функции `InputBox` последовательно появится на экране три раза.

Введите Ваше имя и отчество и щелкните на кнопке **OK**. Затем введите значение переменной `i` и щелкните на кнопке **OK**. Наконец, введите значение переменной `s` и щелкните на кнопке **OK**. При вводе значения переменной `s` не забудьте, что для отделения целой части от дробной части следует применять символ запятая (,), если именно он установлен в качестве разделителя при настройке операционной системы. Иначе выполнение программы закончится ошибкой времени исполнения № 13 с сообщением о несоответствии типов. Если ввод данных выполнен корректно, то введенные значения будут отображены в полях надписей **Label2**, **Label3** и **Label4**.

Щелкните на кнопке **Выход** для остановки программы.

3.2 Вывод данных с помощью инструкции **Print**

При помощи инструкции **Print** можно печатать тексты на форме, а

также на изображении. Текст должен при этом стоять в кавычках (").

Пример:

```
Print " Привет!"
```

Пишет на форме:

Привет!

В инструкции **Print** можно выполнять вычисления, например:

```
Print 2*3
```

выдаст на форме:

6

Всё, что стоит после слова **Print** в кавычках, при выполнении компьютером, будет символ за символом выведено на форме. А всё, что стоит справа от **Print** не в кавычках, будет вычислено компьютером и выведено на форме.

Так, к примеру, $2*3$ для компьютера означает результат умножения 2 на 3.

Синтаксис инструкции **Print**:

```
[объект.] Print [СписокВывода]
```

Здесь объект – это имя формы, может быть опущено для текущей формы, СписокВывода – текст, который печатается на форме. Несколько элементов в списке вывода отделяются друг от друга точкой с запятой (;) или запятой (,). При использовании запятой между выводимыми элементами СпискаВывода будут сделаны интервалы. При использовании точки с запятой между выводимыми элементами СпискаВывода они будут напечатаны друг за другом без перерыва. Например, следующая команда напечатает сообщение в форме *Form1*:

```
Form1.Print "Это форма Form1"
```

Если же форма *Form1* является текущей, то имя объекта в команде может быть опущено и команда будет записана короче:

```
Print "Это форма Form1"
```

Если список вывода не заканчивается точкой с запятой (;) или запятой

(,), то каждое обращение к инструкции **Print** начинает вывод в следующей строке. Если список вывода заканчивается точкой с запятой (;), то следующее обращение к методу **Print** приведет к продолжению печати в той же строке без перерыва. Если же список вывода заканчивается запятой (,), то следующее обращение к методу **Print** приведет к продолжению печати в той же строке с некоторым отступом.

Если нужно распечатать значения нескольких данных в одной строке, то все их нужно перечислить после команды **Print** в СпискеВывода, например, в результате выполнения пары команд:

```
Print "2*2 всё ещё равно "; 2*2;"
```

```
Print " а 59 умножить на ноль равно "; 59 * 0;"
```

получим:

2*2 всё ещё равно 4,

а 59 умножить на ноль равно 0.

Для очистки формы от результата выполненной на ней ранее печати может быть применен метод **Cls**.

Например, следующая команда очистит форму **Form1**:

```
Form1.Cls
```

Если форма **Form1** является текущей, то имя формы можно не указывать.

Координаты вывода устанавливаются в форме командами:

```
[объект.] CurrentX = координата
```

```
[объект.] CurrentY= координата
```

Например, выполнение вывода текста в текущей форме, начиная с левого верхнего угла, устанавливается командами:

```
CurrentX=0
```

```
CurrentY=0
```

Следует иметь в виду, что начало координат находится в левом верхнем углу объекта, ось координат *X* направлена вправо, ось координат *Y* направлена вниз, значения координат задаются в твипах (1 сантиметр равен

примерно 567 твипам).

Для форматирования чисел, дат и времени применяется функция **Format**, которая преобразует их в строку символов. Она позволяет задать, сколько десятичных разрядов отведено на запись числа, нужны или нет лидирующие нули, замыкающие нули, обозначения валюты, разделители тысяч.

Вот упрощенный синтаксис функции **Format**:

Format(Выражение, Формат)

Аргумент **Выражение** определяет значение, которое требуется преобразовать. Аргумент **Формат** – это текстовая строка, сформированная из символов: (0) – в этой позиции должен быть напечатан замыкающий или лидирующий ноль; (#) – в этой позиции не следует печатать замыкающий или лидирующий ноль; (.), (.),(-), (+), (\$), (пробел) – эти символы помещаются в той позиции, где они указаны.

Ниже приводятся примеры преобразования чисел функцией **Format**:

Обращение к функции	Результат
Format (315.4,"00000.00")	00315.40
Format (315.4,"#####.##")	315.4
Format (6315.4,"##,##0.00")	6,315.40
Format (315.4,"\$##0.00")	\$315.40

Теперь надо опробовать работу инструкции **Print**. Продолжайте работу с проектом **Project1**.

Создайте вторую форму, на которой будут выводиться результаты печати. Для этого щелкните на кнопке **Add Form** панели инструментов, выберите **Form** на вкладке **New** и щелкните на кнопке **Открыть**. В окне конструктора форм появится форма **Form2**, а в окне проекта видно, что теперь проект содержит две формы.

Если окна макета форм на экране нет, то откройте его щелчком на кнопке **Form Layout Window** панели инструментов. Методом перетаскивания

разместите обе формы в окне макета форм так, чтобы они не перекрывали друг друга. Тогда при выполнении проекта Вы их сможете видеть на экране одновременно.

Откройте окно кода процедуры `Command2_Click` и введите приведенный далее код.

```
Private Sub Command2_Click()  
    Dim i As Integer, s As Single, d As Date  
    i = InputBox("Задайте целое число")  
    s = Rnd 'Функция Rnd возвращает случайное число  
           'из диапазона [0,1], т.е. 0<=Rnd<1  
    d = InputBox("Задайте дату в формате дд.мм.гг")  
    Form2.Show 'Метод Show делает форму видимой  
    Form2.Print "i= " & i  
    Form2.Print "s= " & s  
    Form2.Print "d= " & d  
End Sub
```

Чтобы результаты вывода на форме сохранялись при ее перекрытии другим окном или при ее сворачивании и восстановлении, нужно установить перерисовывание изображения. Для этого задайте свойству ***AutoRedraw*** формы ***Form2*** значение `True`.

Щелкните на кнопке ***Start*** (Пуск) на панели инструментов для запуска программы.

Щелкните на кнопке ***Print***. VB начнет выполнять процедуру `Command2_Click`.

Введите данные и проанализируйте полученные результаты.

Остановите выполнение приложения.

Повторите несколько раз предыдущие четыре пункта задания, варьируя вводимые данные.

Замените две последние инструкции процедуры `Form2.Print`:

```
Form2.Print "s= " & s
```

```
Form2.Print "d= " & d
```

на следующие инструкции:

```
Form2.Print "s= " & Format(s, "+0.####")
```

```
Form2.Print "d= " & Format(d, _  
"dddd, dd mmmm, уууу год")
```

Сохраните проект.

Щелкните на кнопке **Start** (Пуск) на панели инструментов для запуска программы.

Щелкните на кнопке **Print**. Введите данные. В качестве значения переменной *d* задайте дату своего рождения. Проанализируйте полученные результаты. Если Вы не знали, в какой день недели Вы родились, то теперь Вам это известно.

Остановите выполнение приложения.

3.3 Вывод данных с помощью функции MsgBox

Для вывода значения некоторого выражения может быть применена функция MsgBox. Для этого следует обратиться к функции MsgBox, пользуясь, например, упрощенным синтаксисом:

```
MsgBox (Сообщение) [, ,Заголовок]
```

Здесь Сообщение – это выражение, которое будет преобразовано в строку символов и выведено в окне MsgBox, а Заголовок – строка символов, отображаемая в заголовке этого окна.

Дважды щелкните на кнопке **Command3** на **Form1**. В окне **Code** отобразится процедура Command3_Click.

Скопируйте в эту процедуру код инструкций процедуры Command2_Click.

Для этого откройте окно кода процедуры Command2_Click.

Выделите все инструкции процедуры, кроме первой и последней строки, щелкните на кнопке **Copy** (Копировать) панели инструментов и закройте окно процедуры Command2_Click.

В открывшемся окне процедуры Command3_Click установите курсор в начало первой после заголовка процедуры строки и щелкните на кнопке **Paste** (Вставить) панели инструментов.

Удалите строку с инструкцией:

```
Form2.Show 'Метод Show делает форму видимой
```

Замените три указанные ниже инструкции кода:

```
Form2.Print "i= " & i
Form2.Print "s= " & Format(s, "+0.####")
Form2.Print "d= " & Format(d, _
"dddd, dd mmmm, уууу год")
```

на три инструкции:

```
MsgBox ("i= " & i), , "Вывод переменной i"
MsgBox ("s= " & Format(s, "+0.####")), , _
"Вывод переменной s"
MsgBox ("d= " & Format(d, _
"dddd, dd mmmm, уууу год")), , _
"Вывод переменной d"
```

Данные инструкции поочередно осуществляют вызов функции MsgBox и отобразят значения переменных i, s, d в соответствующем окне сообщения.

После всех выполненных изменений программный код процедуры Command3_Click должен выглядеть так:

```
Private Sub Command3_Click()
    Dim i As Integer, s As Single, d As Date
    i = InputBox("Задайте целое число")
    s = Rnd 'Функция Rnd возвращает случайное число
    'из диапазона [0,1), т.е. 0<=Rnd<1
    d = InputBox("Задайте дату в формате дд.мм.гг")
    MsgBox ("i= " & i), , "Вывод переменной i"
    MsgBox ("s= " & Format(s, "+0.####")), , _
```

```
"Вывод переменной s"  
MsgBox ("d= " & Format(d, _  
"dddd, dd mmmm, уууу год")), , _  
"Вывод переменной d"  
End Sub
```

Щелкните на кнопке **Start** (Пуск) на панели инструментов для запуска программы.

Щелкните на кнопке **MsgBox**, введите данные и проанализируйте полученные результаты. Первое из трех окон функции MsgBox будет выглядеть примерно так, как показано на рисунке (Рисунок 23).

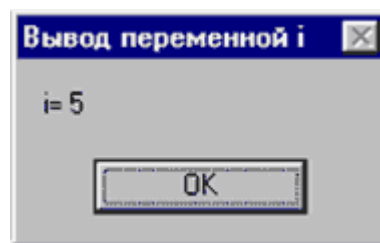


Рисунок 23

Щелкните на кнопке **OK** для закрытия окна каждого сообщения функции MsgBox.

Щелкните на кнопке **Выход** для остановки программы.

Закройте VB.

3.4 Вопросы для контроля

- Каково назначение функции InputBox?
- Какой символ применяется во входном потоке для отделения целой части числа от дробной части?
- В чем заключается действие метода Print?
- Как выполнить очистку формы?
- Как сделать форму видимой?
- В чем состоит назначение функции Format?
- Каково назначение функции MsgBox?

– Можно ли управлять положением на экране диалогового окна функции InputBox?

4 Выражения в VB

При выполнении вычислительных операций над данными в VB используются выражения.

Выражения – это конструкции определенного типа, позволяющие выполнять операции над данными.

Операция – это некоторое действие по вычислению, обработке данных.

Данные, обрабатываемые с помощью операций, называются **операндами**.

Примеры операций: сложение, вычитание, возведение в степень, объединение (конкатенация) строк.

В общем случае, по количеству операндов операции могут быть **унарными** (имеют один операнд), **бинарными** (имеют два операнда), **тернарными** (имеют три операнда) и т.д.

В VB определены унарные и бинарные операции. Подробно они будут рассмотрены ниже.

Принято классифицировать выражения по типу обрабатываемых ими данных. Выделяют следующие типы выражений:

- Арифметические;
- Логические;
- Символьные.

Арифметические выражения используются для выполнения арифметических операций над данными. В VB определены следующие арифметические операции:

- Сложение;
- Вычитание;
- Умножение;
- Деление;
- Возведение в степень;

- Целочисленное деление;
- Остаток от деления;
- Унарный минус.

Логические выражения используются для выполнения операций булевой логики над данными. В булевой логике используются 2 значения: истина (True) и ложь (False). В VB определены следующие логические операции:

- Логическое И;
- Логическое ИЛИ;
- Логическое НЕ;
- Исключающее ИЛИ;
- Логическая эквивалентность;
- Логическая импликация.

Кроме того, в логических выражениях могут использоваться так называемые операции отношения, которые используются для сравнения значений величин:

- Больше;
- Меньше;
- Равно;
- Не равно;
- Больше равно;
- Меньше равно.

Рассмотрим более подробно операции соответствующих типов и операторы, которые их реализуют.

4.1 Арифметические операции

Операндами арифметических выражений могут выступать данные целочисленных, вещественных типов либо логические или символьные значения, преобразуемые к какому либо целочисленному или вещественному

значению (VB пытается преобразовать значение операнда требуемому операцией к типу данных, такое преобразование называется **неявным преобразованием типов данных**).

Примеры неявного преобразования типов данных приведены в главе 2.

4.1.1 Сложение

Данная операция является бинарной. Операнды операции записываются слева и справа от оператора.

Данная операция производит арифметическое сложение операндов.

Оператор: «+»

Синтаксис операции: <Операнд> + <Операнд>

Примеры выполнения операции:

$5 + 3$ – результат 8;

$12.7 + 6$ – результат 18.7.

4.1.2 Вычитание

Данная операция является бинарной. Операнды операции записываются слева и справа от оператора.

Данная операция производит арифметическое вычитание из значения первого операнда значение второго.

Оператор: «-»

Синтаксис операции: <Операнд> - <Операнд>

Примеры выполнения операции:

$5 - 3$ – результат 2;

$12.7 - 6$ – результат 6.7.

4.1.3 Умножение

Данная операция является бинарной. Операнды операции записываются слева и справа от оператора.

Данная операция производит арифметическое умножение операндов.

Оператор: «*»

Синтаксис операции: <Операнд> * <Операнд>

Примеры выполнения операции:

$5 * 3$ – результат 15;

$12.7 * 6$ – результат 76.2.

4.1.4 Деление

Данная операция является бинарной. Операнды операции записываются слева и справа от оператора.

Данная операция производит арифметическое деление значения первого операнда на значение второго.

Оператор: «/»

Синтаксис операции: <Операнд> / <Операнд>

Примеры выполнения операции:

$5 / 3$ – результат 1.66666666667;

$12.6 / 6$ – результат 2.1.

4.1.5 Возведение в степень

Данная операция является бинарной. Операнды операции записываются слева и справа от оператора.

Данная операция производит возведение значения операнда в степень, соответствующую значению второго.

Оператор: «^»

Синтаксис операции: <Операнд> ^ <Операнд>

Примеры выполнения операции:

$5 ^ 3$ – результат 125;

$12.6 ^ (0.25)$ – результат 1.88405092018761 (обратите внимание, что, в данном случае, вычисленное значение соответствует корню 4 степени из первого операнда).

4.1.6 Целочисленное деление

Данная операция является бинарной. Операнды операции записываются слева и справа от оператора.

Данная операция производит целочисленное деление значения первого операнда на значение второго (результатом операции является целая часть от результата операции деления).

Оператор: «\»

Синтаксис операции: <Операнд> \ <Операнд>

Примеры выполнения операции:

$5 \setminus 3$ – результат 1;

$12.6 \setminus 6$ – результат 2

Обратите внимание, что от результата операции деления «отбрасывается» дробная часть, а не производится округление ($5 / 3 = 1.66666666667$, а $5 \setminus 3 = 1$).

4.1.7 Остаток от деления

Данная операция является бинарной. Операнды операции записываются слева и справа от оператора.

Данная операция вычисляет остаток от деления значения первого операнда на значение второго.

Оператор: «mod»

Синтаксис операции: <Операнд> mod <Операнд>

Примеры выполнения операции:

$5 \bmod 3$ – результат 2;

$12.6 \bmod 6$ – результат 1;

Обратите внимание, что результатом операции является целое число, которое может быть вычислено путем округления остатка от деления. Например, результатом операции $12.3 \bmod 6$, является значение 0. (Если говорить более точно, то операндами данной операции являются целые числа и округление производится до выполнения операции.)

4.1.8 Унарный минус

Данная операция является унарной. Единственный операнд операции записывается справа от оператора.

Данная операция используется для указания отрицательных значений (для инвертирования знака операнда).

Оператор: «-»

Синтаксис операции: - <Операнд>

Примеры выполнения операции:

-3 – результат -3;

-12.6 – результат -12.6;

Заметьте, что при применении данной операции в более сложных выражениях (совместно с бинарными операциями) возникает ситуация, когда записываются два оператора подряд. Например, результат выполнения выражения $5 + -3$ равен 2. Также возможно многократное использование данного оператора: $5 + --3$ (результат 8), хотя, обычно, это не целесообразно.

4.2 Логические операции

Операндами логических выражений могут выступать данные логического типа либо целочисленные, вещественные или символьные значения, преобразуемые к значению логического типа.

Результаты логических операций будут определяться с помощью специальных таблиц (таблиц результата), где в названиях строк будут приведены значения первого операнда, в названиях столбцов – второго, а в других ячейках значения результатов операций для значений операндов, определяемых столбцом и строкой ячейки.

Ниже приведено описание операций логическое И, логическое ИЛИ и логическое НЕ. С использованием данных операций может быть построено логическое выражение любой сложности.

Подробное описание операций исключающее ИЛИ, логическая эквивалентность и логическая импликация приводится в курсе лекций.

4.2.1 Логическое И

Данная операция является бинарной. Операнды операции записываются слева и справа от оператора.

Оператор: «And»

Синтаксис операции: <Операнд> And <Операнд>

Таблица результата:

Таблица результатов операции логическое И

Операнд 1 \ Операнд 2	True	False
	True	False
True	True	False
False	False	False

Примеры выполнения операции:

True And True – результат True;

False And True – результат False.

4.2.2 Логическое ИЛИ

Данная операция является бинарной. Операнды операции записываются слева и справа от оператора.

Оператор: «Or»

Синтаксис операции: <Операнд> Or <Операнд>

Таблица результата:

Таблица результатов операции логическое ИЛИ

Операнд 1 \ Операнд 2	True	False
	True	False
True	True	True
False	True	False

Примеры выполнения операции:

True Or True – результат True;

False Or True – результат True.

4.2.3 Логическое НЕ

Данная операция является унарной. Операнд операции записываются справа от оператора.

Оператор: «Not»

Синтаксис операции: Not <Операнд>

Таблица результата:

Таблица результатов операции логическое НЕ

Операнд 1	Результат
True	False
False	True

Примеры выполнения операции:

Not True – результат False;

Not False – результат True.

Отметьте, что данная операция может быть применена к целочисленным значениям. В этом случае выполняется побитовое инвертирование операнда (все биты, представляющие число инвертируются, то есть единицы заменяются нолями и наоборот).

Примеры выполнения операции:

Not 0 – результат -1

Not -4 – результат 3

Not 8 – результат -9

Можно заметить, что при выполнении операции побитового инвертирования отрицательных значений результат определяется следующим образом: $-1 * (<\text{Значение}> + 1)$.

Такой результат операции определяется форматом представления

целочисленных значений в оперативной памяти.

4.2.1 Операции отношения

Данные операции являются бинарными. Операнды операции записываются слева и справа от оператора.

Операторы операций представлены в следующей таблице:

Операция отношения	Оператор
Больше	>
Меньше	<
Равно	=
Не равно	<>
Больше равно	>=
Меньше равно	<=

Синтаксис операций: <Операнд> <Оператор операции> <Операнд>

Примеры выполнения операций над целочисленными и вещественными значениями:

5 >= 3 – результат True;

Пример выполнения операции над логическими значениями:

True < False – результат True⁶.

Примеры выполнения операций над строковыми значениями:

“Б” > “А” – результат True

“б” > “А” – результат True

“Б” > “а” – результат False

“б” > “а” – результат True

“ф” > “f” – результат True

При выполнении операций отношения над строками производится

⁶ В оперативной памяти логические значения True и False представляются как двухбайтовые значения -1 и 0 соответственно.

посимвольное сравнение строк, начиная с самого левого символа. При этом сравниваются ASCII коды символов, которые приведены в приложении 10. Так в первом примере для строк, состоящих из одного символа, производится сравнение “Б” > “А”, то есть сравниваются следующие коды символов: 193 > 192 – результат истина.

Из приложения 10 видно, что символы чисел имеют меньшие последовательные значения кодов, чем заглавные символы латинского алфавита, которые, в свою очередь имеют меньшие значения, чем символы русского алфавита. Заглавные же символы русского алфавита также имеют меньшие значения кодов, чем строчные символы русского алфавита. Ознакомьтесь с таблицей кодов символов более подробно, найдите символы «пробел», «точка», «запятая», «точка с запятой», «двоеточие», «дефис». Запомните их коды.

Для облегчения работы пользователя в VB определен ряд встроенных констант, имеющих наглядные названия, значения которых соответствуют ASCII кодам некоторых, часто используемых символов. С данными константами и их значениями вы можете ознакомиться в приложении 11. Имена данных констант легче запомнить, чем коды соответствующих символов, что поможет вам сократить время разработки некоторых приложений.

Примеры сравнения строк, состоящих из нескольких символов:

“БВБ” > “БВАБ” – результат True;

“Холод” >= “Холодный” – результат False.

При вычислении результата первой операции отношения первого примера производится посимвольное сравнение строк:

- Сравниваются первые символы строк “Б” и “Б” данные символы равны, поэтому осуществляется переход к следующим символам строк;
- Сравниваются вторые символы строк “В” и “В” данные символы равны, поэтому осуществляется переход к следующим

символам строк;

- Сравниваются третьи символы строк “Б” и “А” код символа “Б” больше кода символа “А”, следовательно, первая строка больше второй. Результат Истина.

Заметьте, что сравнение символов прекратилось на первой паре неравных символов и последующие символы далее не рассматривались (символ “Б” второй строки).

Второй пример демонстрирует сравнение строк, в которых одна строка является началом другой. При достижении конца более короткой строки (“Холод”) сравнение завершается, и данная строка считается меньшей, чем более длинная (“Холодный”).

4.3 Символьные операции

Над строковыми значениями в VB определена одна операция – **конкатенация** (объединение строк).

Данная операция может быть выполнена с помощью любого из двух операторов: «+» или «&».

Однако рекомендуется использовать оператор «&», так как при использовании оператора «+» выражение может быть интерпретировано как арифметическое (см. главу 2).

4.4 Синтаксис выражений

Выражение можно определить следующим образом:

<операнд> [<знак операции> <операнд>] [<знак операции> <операнд>] ...

, где, в зависимости от типа выражения, используются соответствующие знаки операций.

Операнды выражения:

- неименованная константа (указанное в выражении значение);
- константа;
- переменная;

- свойство объекта;
- элемент массива (будет рассмотрен позже);
- обращение к функции;
- подвыражение в скобках.

Примеры выражений:

$66 + "4" - 2^3 * (1 - -4)$ – результат 30;

True And (False Or True) – результат True.

Вычисление выражений выполняется путем выполнения операций слева на право. При этом на порядок выполнения операций оказывают влияние приоритет операции и скобки.

Так в арифметических выражениях операции имеют следующие приоритеты (в порядке убывания приоритета):

- возведение в степень,
- умножение и деление с плавающей точкой,
- целочисленное деление,
- вычисление остатка,
- сложение и вычитание.

Соответственно при вычислении результата первого выражения примера производятся следующие действия:

- Вычисляется результат подвыражения $66 + "4" = 70$;
- Выражение принимает следующий вид: $70 - 2^3 * (1 - -4)$;
- Для выполнения операции вычитание необходимо вычислить результат операции возведение в степень, имеющую более высокий приоритет: $2^3 = 8$;
- Выражение принимает следующий вид: $70 - 8 * (1 - -4)$;
- Для выполнения операции вычитание необходимо вычислить результат операции умножение, причем вторым операндом является выражение в скобках, результат которого необходимо вычислить до выполнения операции умножение: $(1 - -4) = 5$;
- Выражение принимает следующий вид: $70 - 8 * 5$;

- Результат операции умножение: $8 * 5 = 40$;
- Выражение принимает следующий вид: $70 - 40$;
- Результат операции вычитание: $70 - 40 = 30$;
- Результат выражения $66 + "4" - 2^3 * (1 - 4) = 30$.

Выполните самостоятельно вычисление второго выражения примера.

4.5 Явное преобразование типов данных

Неявное преобразование типов данных в VB является удобным инструментом, однако в некоторых случаях его использование приводит к непредсказуемым результатам. Например, в приложение, запрашивающее у пользователя число в виде строки и увеличивающее его значение на 2, может работать некорректно при вводе пользователем строки, которую невозможно преобразовать к числу. В этом случае, возможно выполнение преобразование числа 2 к строке и выполнение операции объединения строк, что приведет к вычислению неверного результата.

Для исключения подобных ошибок возможно использование явного преобразования типов данных. Явное преобразование типов данных осуществляется с помощью встроенных функций языка программирования: CBool, CByte, CCur, CDate, CDb1, CInt, CLng, CSng, CStr, CVar (синтаксис, назначение и примеры использования встроенных функций VB приведены в приложении 12 Приложение: Дополнительные материалы по вычислению некоторых функций).

Функции – это именованные предопределенные процессы, которые возвращают результат вычисления в то выражение, где они были вызваны.

Именованные предопределенные процессы могут принимать данные, над которыми будут производиться вычисления. Значения, передаваемые предопределенным процессам, называются **аргументами** или **параметрами**.

Например, $\text{abs}(-5)$ – функция, вычисляющая модуль от переданного ей значения -5 . Здесь -5 является аргументом функции. Результатом данного выражения является число 5 .

Ознакомьтесь со встроенными функциями VB, а также дополнительными материалами, представленными в приложении 13.

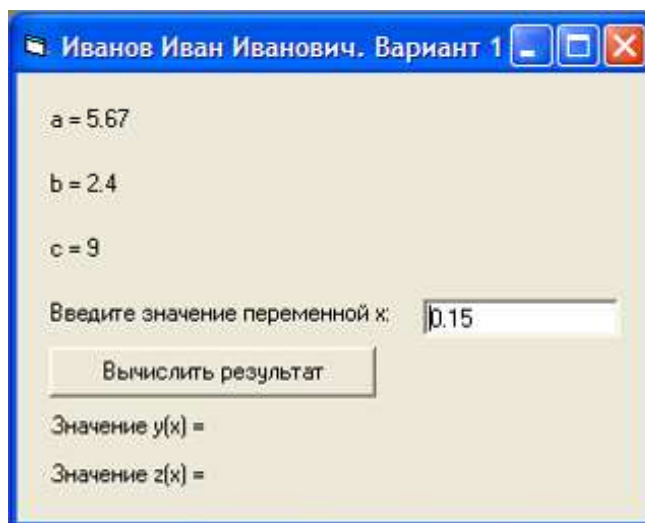
4.6 Задание для самостоятельной работы

Задание: составить приложение для вычисления значений переменных y и z для заданных пользователем значений переменной x и констант a , b , c , соответствующих варианту.

Перед выполнением задания изучите синтаксис, назначение и повторите примеры использования встроенных арифметических функций языка программирования: abs , atn , cos , exp , fix , int , log , rnd , sgn , sin , sqr , tan .

1 Вариант задания необходимо получить у преподавателя.

2 Разработать форму приложения, соответствующую изображенной на следующем рисунке:



3 Разработать исходный код приложения. При разработке необходимо учесть следующие требования:

- Значение константы π необходимо задавать равным 3,14159265358979.
- Все переменные и константы приложения должны быть объявлены в явном виде, при этом необходимо выбрать

оптимальные типы данных, необходимые и достаточные для хранения принимаемых ими значений.

- При выводе результатов необходимо отформатировать значения с точностью до 10 знаков после запятой.
- Необходимо исключить неявные преобразования типов данных при вычислении значений выражений, используя функции явного преобразования типов данных.

4 Выполнить отладку приложения. Убедиться в правильности вычисляемых результатов.

5 Показать результат работы преподавателю.

Варианты практических заданий

№	x	a	b	c	y	z
1	0.015 0.37 -0.065	5.67	2.4	9	$\frac{\pi/2 + \arctg(\cos(x/a) - \sin(x/b))}{(a-b)(x+a)}$	$\frac{ax - bx}{\sqrt[3]{ ax/c + 1 }}$
2	0.005 -0.456 -3.89	2	5.2	5	$\frac{\arctg(a \cdot \tg(b/x))}{ab \cdot \log_c x+a }$	$\frac{a+bx}{\sqrt[3]{ ax+x^2 }}$
3	0.567 -3.56 -0.67	7	1.11	4	$\frac{\sqrt{ x+a } + \sqrt[3]{ x }}{\log_{3.5} b+x }$	$\frac{x+c}{(b+4x)(x+a)(x+b)}$
4	-0.789 0.123 -4.6	4.34	3	2	$\frac{e^{\sqrt[3]{ x }} - 2x}{b \cdot \sqrt{ x-2\pi }}$	$\frac{\log_a 2b-x }{(a-b)\log_c [bx]}$
5	-7.34 0.678 -0.004	3	6	3	$\frac{\sqrt{ x-b } + \sqrt[3]{ 4x/3+b }}{(b-a^2)(x+b)}$	$\frac{b \cdot \log_a b/2+x }{a \cdot \log_c 2x-c }$
6	6.34 3.234 -0.04	1.81	3.7	2	$\frac{\tg(\pi \cdot x/a) + \sqrt[3]{ x-8 }}{x(b+a)}$	$\frac{\log_c x+2b }{\sin(x/a) + \lg x/b }$
7	-0.45 -1.34 3.56	4.9	-7	7	$\frac{\sin(x/a) + b \cdot \cos x}{\log_c ax-b }$	$\frac{a \cdot \pi - \sqrt[3]{ x+b }}{(x+a)(x+b)}$
8	3.78 -2.567 -0.345	6	-90	6	$\frac{a \cdot \sin x + b \cdot \cos(x/a)}{b \cdot \sqrt[3]{ x-a/b }}$	$\frac{\sqrt{ x-2b }}{a \cdot \log_c ax-bx }$

№	x	a	b	c	y	z
9	0.43 -0.0014 1.78	23	5.12	7	$\frac{\cos^2(a \cdot \arctg x + 3)}{\sin(b \cdot \arctg(ax))}$	$\frac{\sqrt[3]{ x+6x }}{x+2b}$
10	-0.987 -0.021 8.45	244	3	4	$\frac{\sin(bx + \pi) + \cos(19 - \arctg x/a)}{ab}$	$\frac{\sqrt[3]{ x-2c }}{tg \pi/x - \log_c x+cb }$
11	0.432 -0.21 -4.28	32	123	4	$\frac{\arctg(\sin ax) - \sin^2 x/b}{\log_c x + e^{-x} }$	$\frac{\sqrt[3]{ x+ca }}{(a-x)(b-x)}$
12	0.013 -0.321 5.765	66	5.8	3	$\frac{\arctg(\pi/a + bx) - \sqrt[a]{ x }}{\log_c x/a - b }$	$\frac{\cos 2x + bx}{(x-a)(x^2 - b)}$
13	0.016 -0.34 7.12	33	-6.1	7	$\frac{tg \log_a x - \sqrt[b]{ x+ab }}{ax}$	$\frac{\sqrt{ 2x+3ac }}{c \cdot \log_a x-acb }$
14	3.76 -0.98 -1.56	1.90 8	3.98	8	$\frac{tg(\arctg x/a + 1) - \cos^2 bx}{(x-a) \cdot (b^2 + a)}$	$\frac{a \sqrt[3]{ x+y }}{(b-a) \log_c a-2x }$
15	0.43 -0.0012 1.56	65	5.32	1.01	$\frac{\log_a x + b \sqrt{ x } }{2ax - x/b}$	$\frac{x/y - \sqrt[3]{ xy }}{\log_c x-b }$
16	1.23 -0.765 0.124	-23	2.4	9	$\frac{e^{\sqrt{ x/a }} - tg(x/b)}{(x-a)(x-b)}$	$\frac{ay + bx}{(a-b) \log_c 2x-a }$
17	0.43 -0.54 0.0045	5.67	5.2	5	$\frac{\sqrt[a]{ tg x/2 + \sin x/3 }}{e^x - e^{-x}}$	$\frac{ax + by}{(a-b) \log_c 2x-b }$
18	0.21 -0.453 9.87	2	1.11	4	$\frac{tg(\arctg x/a + 1) - \cos^2 bx}{(x-a)(b^2 - a)}$	$\frac{2x - y}{b \cdot \log_c x-c }$
19	-0.231 -5.21 0.0015	7.1	3.09	2	$\frac{\log_a x+a-b }{\sqrt{ a+b }}$	$\sqrt{tg \left(3 \arctg \frac{x-4}{3} \right) + x^3}$
20	0.0038 -0.14 2.45	4.34	6	3	$\frac{a \cdot \arctg(\cos x + tg x/b)}{\log ax + a/b }$	$\frac{\sqrt[3]{ x+c }}{(x-a)(y-b)c}$

№	x	a	b	c	y	z
21	3.678 -1.54 -0.003	3	3.7	2	$\frac{\sqrt{ x-a } + \sqrt[3]{ x/b+a }}{(x+b)(b-a)}$	$\frac{a \cdot \log_b x+c }{b \cdot \log_c 2x-x/2 }$
22	0.555 -0.211 0.011	1.81	-7	7	$\frac{\arctg(\cos x/a + b \cdot \tg x)}{\sqrt{ ax-b/x }}$	$\frac{a \cdot \log_c x+a }{(x-b)(a-b)}$
23	0.483 1.934 -0.005	4.9	-90	6	$\frac{\sqrt{ ax + \arctg(c - \tg \pi \cdot x/2) }}{x \cdot (a-b)}$	$\frac{\sqrt[3]{ x+b }}{\log_c 2x-b }$
24	-0.147 -2.498 0.0005	6	5.12	7	$\frac{\cos x/3 + \sqrt[4]{ \tg x - b }}{\sqrt{ x-a \cdot b }}$	$\frac{xa + e^x}{(x+c)(b-x)}$
25	17.54 -0.167 0.545	23	3.67	4	$\sqrt[4]{ \sin x \cdot \cos 2x } + \sqrt{ b+x }$	$\frac{(a+b) \cdot \log_a c+x }{\log_c x }$
26	-0.5443 0.323 4.544	244	123	4	$\frac{\sin(\arctg(x/a) + \cos(x/b))}{b \cdot \log_c 2x-\pi }$	$\frac{bx}{\sqrt[3]{ ax-b }}$
27	5.612 -0.433 0.004	32	5.8	3	$\frac{\sqrt[4]{ x-b } + e^{\sqrt{ x }}}{\log_c x/a + b/c }$	$\frac{\sqrt[3]{ x+2a }}{(x-a)(x-c)(a+c)}$
28	-0.004 0.5443 7.213	66	-6.1	7	$\frac{\sqrt{ x -b} + \sqrt[4]{ 2x-4 }}{\log_c x-b/a }$	$\frac{2xb - \sqrt{ x }}{(x-c)(x-a)}$
29	4.987 -0.43 0.3221	33	3.98	8	$\frac{\sqrt[4]{ \arctg(b \cdot \tg 2x) - \pi/8 }}{(a-b)(x-a)}$	$\frac{2abc - 3x}{ab \cdot \log_c x+abc }$
30	7.32 -0.54 0.0021	1.98	5	1.9	$a \cdot e^{-x} + \sqrt[3]{ x+1.4 }$	$\frac{\sqrt{ x^2 - \cos x }}{\log_c x+acb }$

Примечание: в приложении представлены графики функций, для соответствующих вариантов заданий. Убедитесь в соответствии результатов вычисляемых приложением со значениями, приведенными на графиках функций.

5 Конструкции VB, реализующие базовые алгоритмические структуры

5.1 Базовые структуры алгоритмов

Алгоритм – это правило получения решения некоторой задачи, выраженное в виде совокупности конечного числа элементарных действий. Мы часто пользуемся алгоритмами. Например, когда в столбик складываем два числа, то применяем алгоритм. Известны различные способы записи алгоритма. Любая программа на языке VB – это также запись соответствующего алгоритма. Однако наибольшей наглядностью обладает изображение алгоритма в виде блок-схемы. В сложных случаях, чтобы правильно разработать алгоритм иногда полезно перед использованием средств VB изобразить его блок-схему.

При разработке алгоритма необходимо применять такие технологические рекомендации, при соблюдении которых алгоритм получался бы наиболее понятным, а ошибки при записи алгоритма в виде программы были бы наименее вероятны. В соответствии с современными воззрениями в сегодняшней технологии разработки программ, которые представляют собой одну из сторон метода структурного программирования, алгоритм должен быть структурирован. Он может включать только базовые структуры, которых всего три: следование, разветвление и цикл.

5.1.1 Следование (композиция)

Данная алгоритмическая структура, предполагает последовательное выполнение входящих в нее некоторых инструкций F1 и F2. Существенно, что структура следование, рассматриваемая как единое целое (на рисунке она заключена в пунктирный прямоугольник), имеет один вход и один выход. Отметим, что во всех приложениях, которые вы разработали на текущий момент, использовалась только данная алгоритмическая структура: тело каждого обработчика события представляло собой описание цепочки действий, расположенных в отдельных последовательных строках исходного

кода.

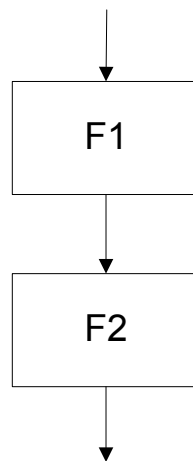


Рисунок - Следование

5.1.2 Ветвление

Ветвление, блок-схема которого приведена на следующем рисунке, предполагает проверку некоторого условия. В зависимости от того выполняется это условие или нет, выполняется либо одна инструкция, либо другая.

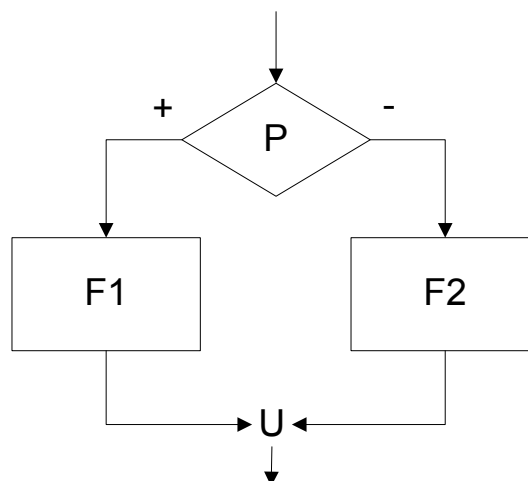


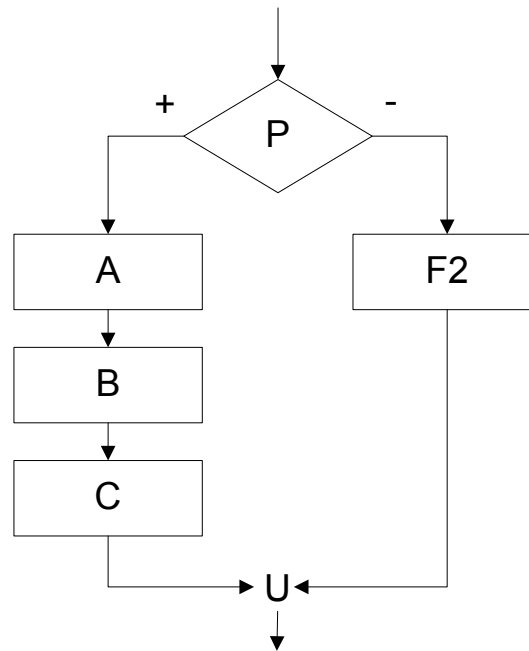
Рисунок - Ветвление

Если на момент проверки условие оказалось выполнено, то будет выполнена инструкция F1, а инструкция F2 игнорируется. Если же оказывается, что условие не выполнено, то будет выполнена инструкция F2, а

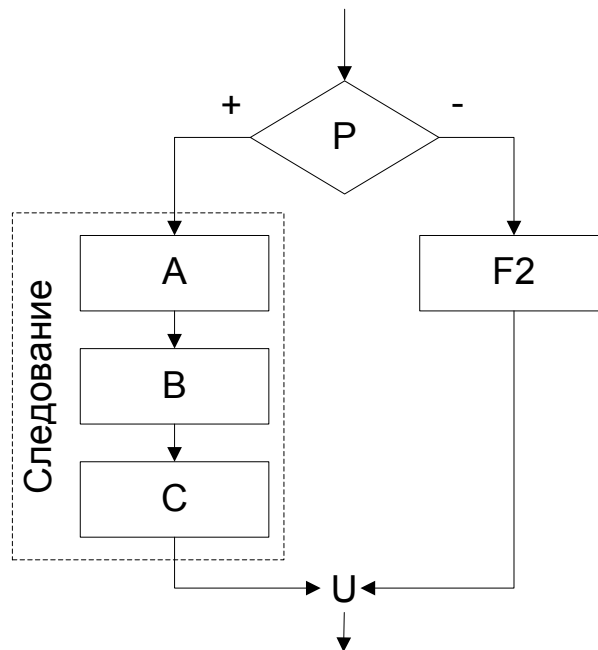
инструкция F1 игнорируется.

Разветвление также имеет один вход и один выход.

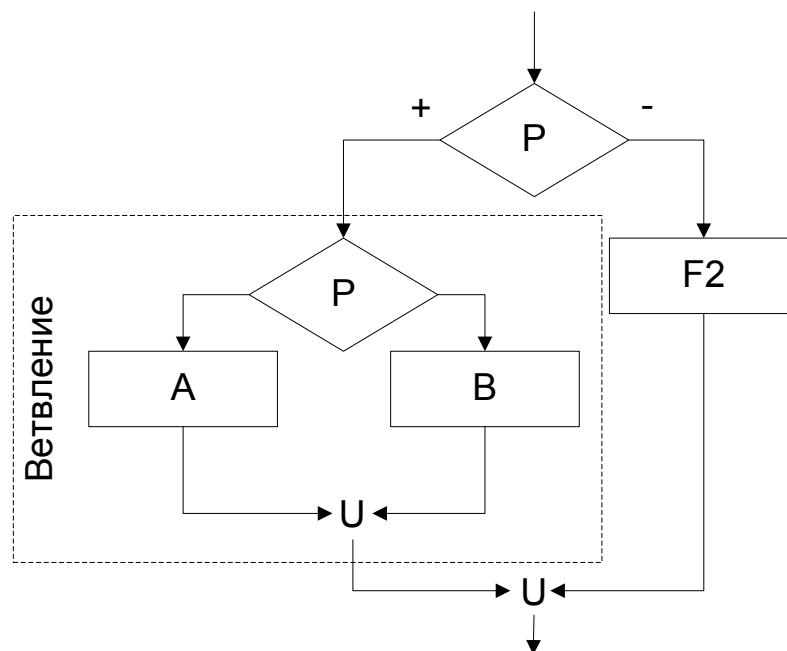
Отметьте, что в каждой ветви алгоритмической конструкции, вместо одного действия может быть описана некоторая иная последовательность: например левая ветвь может содержать описание последовательности инструкций A, B, C.



В представленном на рисунке алгоритме последовательность действий A, B, C представляет собой алгоритмическую инструкцию следование, то есть инструкция F1 в данном алгоритме была заменена вложенной алгоритмической конструкцией следование.

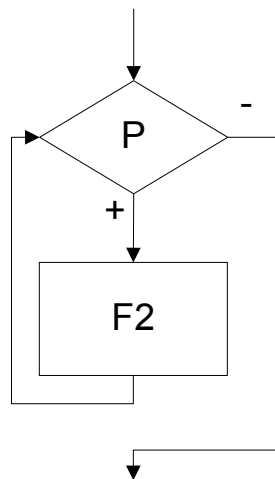


Благодаря тому, что любая алгоритмическая конструкция имеет один вход и один выход и представляет собой единое целое, то любое укрупненное описание действия может быть заменено на вложенный алгоритм, представленный какой либо алгоритмической конструкцией. Так на следующем рисунке вложенной алгоритмической конструкцией является ветвление.



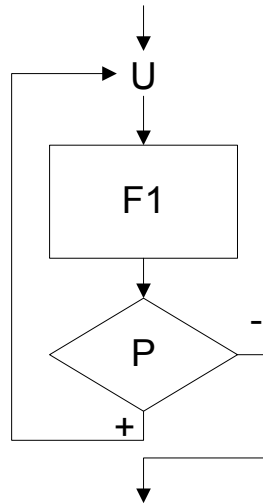
5.1.3 Цикл

Цикл предполагает повторение выполнения некоторой инструкции (тела цикла), а также проверку некоторого условия необходимости повторения (итерации) этой инструкции. Различают два вида базовых циклов в зависимости от порядка выполнения этих действий: сначала проверка условия необходимости выполнения инструкции, а затем ее выполнение (цикл с предусловием), или сначала выполнение инструкции, а затем проверка условия необходимости повторения ее выполнения (цикл с постусловием).



На рисунке изображена блок-схема цикла с предусловием: если условие, представленное логическим выражением P выполняется (выход по стрелке со знаком +), то инструкция $F2$ будет выполнена. Если же условие не выполняется (выход по стрелке со знаком -), то инструкция не будет выполнена и произойдет выход из цикла. Элементарная алгоритмическая конструкция цикл также имеет один вход и один выход.

Блок-схема алгоритма цикл с постусловием представлена на следующем рисунке:



5.2 Конструкция If..Then

В VB алгоритмическая структура ветвление реализуется с помощью конструкции If..Then, которая имеет следующий синтаксис:

Строчная конструкция:

```
If <Логическое_выражение> Then <оператор1> [Else <оператор2>]
```

или

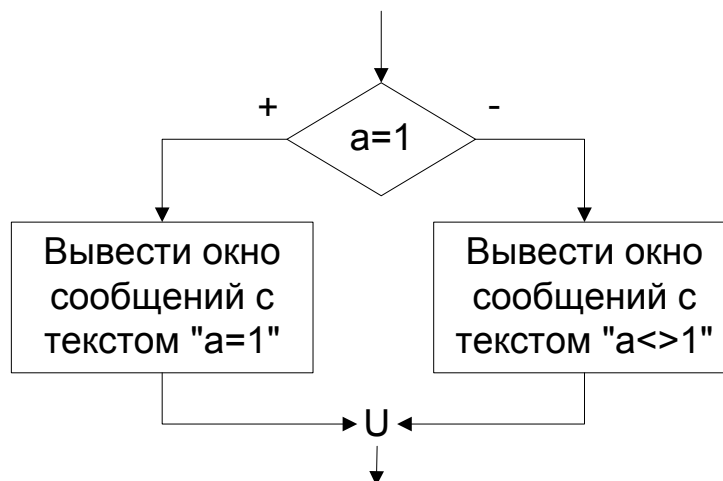
Блочная конструкция:

```
If <Логическое_выражение> Then
    [<Блок операторов1>]
[ElseIf <Логическое_выражение> Then
    [<Блок операторов2>]]
...
[Else
    [<Блок операторовN>]]
End If
```

Пример применения строчной конструкции ветвление:

```
If a=1 Then MsgBox "a=1" Else MsgBox "a<>1"
```

В представленном примере описывается следующий алгоритм:



То есть в зависимости от результата логического выражения $a=1$, которое имеет результат ИСТИНА при значении переменной a равном 1 и ЛОЖЬ при значении данной переменной 2, осуществляется выполнение соответствующих команд, сообщающих о результате вычисления логического выражения.

Таким образом конструкция

If $a=1$ **Then** MsgBox “ $a=1$ ” **Else** MsgBox “ $a \neq 1$ ”

читается следующим образом:

Если логическое выражение $a=1$ имеет значение истина **Тогда** вывести окно сообщений с текстом “ $a=1$ ” **Иначе** вывести окно сообщений с текстом “ $a \neq 1$ ”.

При этом часть конструкции **Else (Иначе)** является не обязательной. При отсутствии данной части конструкции никаких действий при значении логического выражения ЛОЖЬ не выполняется, а осуществляется немедленный выход из конструкции и переход к следующей за ней строке.

Блочная конструкция позволяет описывать последовательности действий, выполняемые при соответствующих значениях логического выражения.

Блочная конструкция также позволяет описывать вложенные алгоритмические конструкции ветвления с помощью опциональной части конструкции **ElseIf (Иначе Если)**.

Ознакомьтесь с примером использования блочной конструкции ветвления:

If a=1 Then

$R = a * 2$

Msgbox "R = a * 2 = " & R

Elseif a>=2 And a < 5 Then

$R = a + 2$

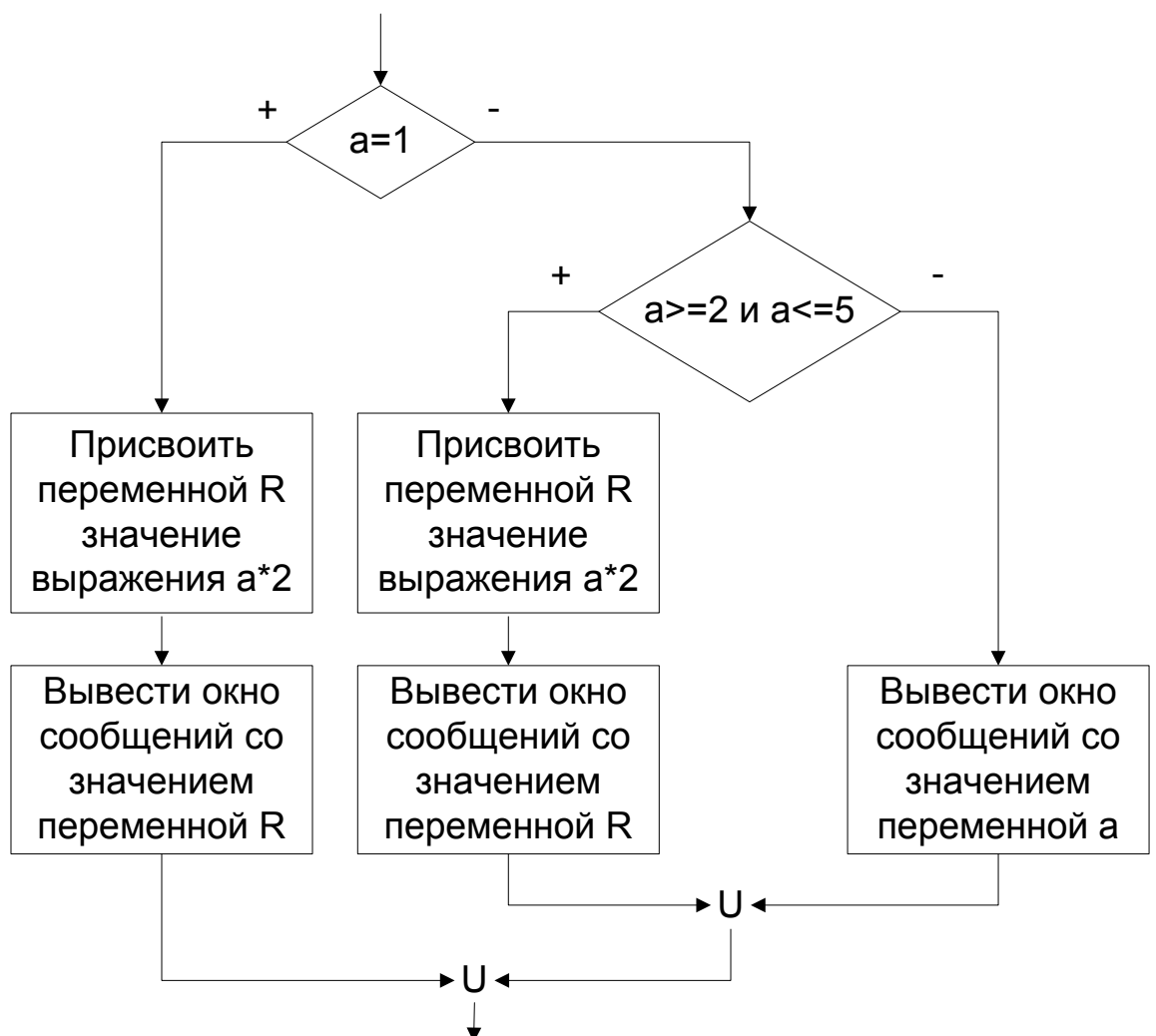
Msgbox "R = a + 2 = " & R

Else

Msgbox "R = a = " & a

End If

В представленном примере описан следующий алгоритм:



По данному алгоритму осуществляется выбор одного из трех описанных вариантов выполняемых в соответствующих случаях.

5.3 Конструкция Select ... Case

Конструкция Select ... Case предназначена для реализации сложных алгоритмов ветвления, где осуществляется выбор одного из нескольких вариантов исполнения. В соответствии со своим назначением конструкция получила название – **Конструкция Выбора**.

Конструкция имеет следующий синтаксис:

```
Select Case <Тестовое_Выражение>
[Case <Список_Проверочных_Выражений1>
    [<Блок операторов1>]]
[Case <Список_Проверочных_Выражений2>
    [<Блок операторов2>]]
...
[Case Else
    [<Блок операторовN>]]
End Select
```

где <Список_Проверочных_Выражений>:

Может быть задан список элементы которого описываются:

- указанием значения, например **11**;
- указанием интервала значений, например **1 To 4**;
- указанием логического выражения, в котором значение тестового выражения сравнивается со значением указанного выражения. Для ссылки на значение тестового выражения здесь используется ключевое слово **Is**. Пример: **Is > MaxNumber**.

Примеры списков проверочных выражений:

2 – список состоит из одного элемента указанного значением 2;

2, 4 To 9 – список содержит два элемента: значение и интервал.

Рассмотрим пример использования конструкции выбора, соответствующий последнему примеру для конструкции ветвления:

Select Case a

Case 1

R = a * 2

Msgbox "R = a * 2 = " & R

Case 2 To 5

R = a + 2

Msgbox "R = a + 2 = " & R

Case Else

Msgbox "R = a = " & a

End Select

При интерпретации данного исходного кода выполняются следующие действия: При исполнении строки **Case 1** значение тестового выражения сравнивается с указанным значением и если они равны, то выполняется соответствующий блок операторов, после чего осуществляется выход из конструкции. Если значения неравны, тогда осуществляется переход к следующей строке Case и так далее, пока не будет исполнен какой либо блок операторов, либо достигнута строка **End Select** (это может произойти при только при отсутствии блока **Case Else**, исполняемого безусловно).

5.4 Конструкция Do...Loop

Конструкция Do...Loop реализует в VB циклы, как с предусловием, так и циклы с постусловием.

Синтаксис конструкции реализующей цикл с предусловием:

Do {While | Until} <Логическое_Выражение>

[<Блок операторов>]

[Exit Do]

[<Блок операторов>]

Loop

где ключевое слово While указывает, что тело цикла должно быть исполнено в том случае, если значение логического выражения ИСТИНА;

ключевое слово `Until` указывает, что тело цикла, напротив, должно быть исполнено в том случае, если значение логического выражения ЛОЖЬ;

Ключевые слова `Exit Do` позволяют досрочно прерывать исполнение цикла.

Пример исходного кода цикла с предусловием:

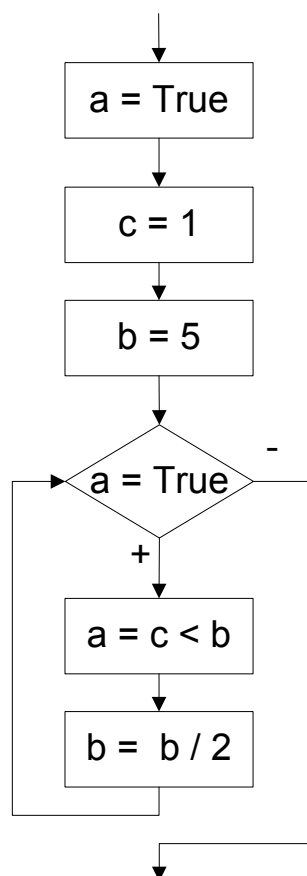
```
a = True
c = 1
b = 5
Do While a = True
```

```
    a = c < b
```

```
    b = b / 2
```

```
Loop
```

Блок-схема алгоритма примера представлена на следующем рисунке:



При исполнении алгоритма переменная **a** получает значение ИСТИНА, переменная **c** – 1, **b** – 5. Логическое выражение `a = True` имеет значение ИСТИНА, соответственно осуществляется исполнение тела цикла.

Во время исполнения тела цикла переменная **a** получает значение ИСТИНА если значение переменной **c** меньше значения переменной **b**. Значение переменной **b** уменьшается в 2 раза, после чего осуществляется переход к повторному вычислению логического выражения $a = \text{True}$.

Таким образом, тело цикла выполняется до тех пор, пока на некоторой итерации, при вычислении логического выражения, определяющего необходимость исполнения тела цикла, не будет получено значение ЛОЖЬ.

Разработайте приложение, реализующее пример, выполните его отладку и выясните сколько раз исполняется тело цикла.

При применении ключевого слова **Until** данный тот же результат будет получен при использовании следующего исходного кода:

```
a = True
c = 1
b = 5
Do Until a = False
    a = c > b
    b = b / 2
Loop
```

Разработайте блок-схему алгоритма для данного примера.

При реализации циклов с постусловием конструкция **Do...Loop** имеет следующий синтаксис:

```
Do
    [<Блок операторов>]
[Exit Do]
    [<Блок операторов>]
Loop {While | Until} <Логическое_Выражение>
```

Отличие состоит лишь в том, что ключевые слова **While** и **Until** и логическое выражение используются в строке **Loop**.

При использовании циклов с постусловием необходимо помнить, что тело цикла в них исполняется хотя бы один раз, так как осуществляется

безусловный вход в цикл.

Разработайте приложение аналогичное примеру цикла с предусловием и разработайте блок-схему алгоритма для него. При разработке приложения используйте сначала ключевое слово **While**. Измените приложение таким образом, чтобы оно выполняло те же действия при использовании ключевого слова **Until**.

5.5 Цикл со счетчиком. Конструкция **For...Next**

При разработке приложений часто возникает необходимость организации циклов, в которых при каждой итерации тела цикла некоторая переменная, называемая счетчиком цикла, изменяет свое значение на заданную величину, а завершение исполнения цикла должно осуществляться при достижении данной переменной определенного значения.

Рассмотрите следующий пример:

```
i = 1
Do While i <= 4
    t = t + i
    i = i + 1
Loop
```

В данном примере на каждом проходе цикла значение переменной **i** увеличивается на заданное значение 1. При достижении переменной **i** (счетчика цикла) значения 5 происходит выход из цикла.

Выполните отладку данного приложения. Обратите внимание на то, что при завершении данного алгоритма в переменной **t** хранится значение равное сумме чисел от 1 до 4.

Циклы со счетчиком применяются достаточно часто, однако при их разработке с использованием конструкции **Do...Loop** существует возможность допустить следующую ошибку: забыть описать строку, в которой производится изменение значения счетчика цикла. При исполнении такого исходного кода значение логического выражения всегда будет равно

ИСТИНЕ, что приведет к заикливанию – вечному исполнению тела цикла. Приложение при заикливании зависает, что крайне не желательно как для разработчика, так и для пользователя приложения.

Упростить описание циклов со счетчиком и практически исключить возможность зависания приложения при разработке позволяет конструкция циклов со счетчиком, имеющая следующий синтаксис:

```
For <Счетчик > = <Начальное_зн-е> To _<Конечное_зн-е> [Step <Шаг>]  
    [<Блок операторов>]  
[Exit For]  
    [<Блок операторов>]  
Next [<Счетчик>]
```

где <Счетчик> - имя переменной счетчика цикла;

<Начальное значение> - начальное значение счетчика цикла, которое он получает при входе в цикл.

<Конечное значение> - конечное значение счетчика цикла (при выходе значения счетчика цикла за данное значение исполнение цикла прекращается);

<Шаг> - значение, на которое изменяется счетчик цикла после каждого прохода (может быть отрицательным).

Отметьте, что данные значения могут быть заданы выражениями.

Ключевые слова Exit For используются для организации досрочного выхода из цикла.

Разработайте приложение выполняющее расчет суммы чисел от единицы до введенного пользователем значения, опишите блок-схему алгоритма данного приложения.

5.6 Задание для самостоятельной работы

Задание: в соответствии с заданным вариантом вычислить выражение, изменяя переменную x от начального значения x_i с шагом h_i .

1 Вариант задания необходимо получить у преподавателя.

2 Разработать форму приложения, соответствующую изображенной на следующем рисунке:

The image shows a Windows application window titled "Form1". Inside the window, there is a label "Вариант 1". Below it, there are two input fields: the first is labeled "X:" and contains the number "2"; the second is labeled "h:" and contains the number "3". Below these fields is a button labeled "Вычислить". At the bottom of the form, there is a label "Результат:" followed by a large empty space for the output.

3 Разработать блок-схему алгоритма и исходный код приложения. При разработке необходимо учесть следующие требования:

- Все переменные и константы приложения должны быть объявлены в явном виде, при этом необходимо выбрать оптимальные типы данных, необходимые и достаточные для хранения принимаемых ими значений.
- При выводе результатов необходимо отформатировать значения с точностью до 10 знаков после запятой.
- Необходимо исключить неявные преобразования типов данных при вычислении значений выражений, используя функции явного преобразования типов данных.

4 Выполнить отладку приложения. Убедиться в правильности вычисляемых результатов.

5 Показать результат работы преподавателю.

Варианты практических заданий

№	X_i	h_x	y
1	2	3	4
1	0	0,5	$\sum_{i=1}^{18} \frac{\log_5(1+x_i)}{\sqrt{1+x_i^2}}$
2	0	0,2	$\prod_{i=1}^{16} (1+\cos x_i^2)$

N_2	X_i	h_x	y
3	0,565	0,05	$\sum_{i=1}^{14} \log_7 \sqrt[5]{x_i} - e^{x_i}$
4	1	0,25	$\sum_{i=1}^{11} \frac{\sin x_i \cdot \log_5(11+x_i)}{1+2\cos^2 x_i}$
5	0	0,3	$\prod_{i=1}^{16} (1.5 - \sin x_i^2)^2$
6	1	0,025	$\sum_{i=1}^{17} \frac{x_i \log_2\left(\frac{1}{x_i}\right)}{\sqrt{1+x_i^2}}$
7	0,01	0,025	$\prod_{i=1}^{16} \frac{1 - \arcsin x_i}{2 + \sqrt{x_i}}$
8	0,5	0,097	$\sum_{i=1}^6 \frac{e^{-2x_i}}{\sqrt{x_i}} \cos \sqrt{\frac{2}{3}x_i}$
9	0,5	0,5	$\prod_{i=1}^{17} \sin(x_i - 1.125) - \sqrt{x_i} $
10	0,1	0,2	$\sum_{i=1}^{17} \frac{(1 - \sqrt{x_i^3})(1 - \sqrt[3]{x_i^2})}{\ln(x_i + 1)}$
11	2,1	0,15	$\prod_{i=1}^{15} \ln\left(\frac{e^{x_i}}{x_i^2 - 1}\right)$
12	$\pi/2$	0,11	$\sum_{i=1}^{17} \frac{e^{x_i}}{x_i} \sin \frac{x_i}{3} \sin \frac{x_i}{2}$
13	0,435	0,05	$\prod_{i=1}^{10} (x_i + \sin x_i) \cdot \ln x_i$
14	0.565	0,05	$\sum_{i=1}^{14} \left(\frac{\ln x_i^2}{0.5 - \cos x_i} - 1 \right)$
15	1	0,5	$\prod_{i=1}^{13} (\pi - e^{x_i} \ln x_i)$
16	0,1	0,15	$\sum_{i=1}^{16} \frac{\left(\ln \frac{1}{x_i}\right) e^{-2x_i} \sin 1.5x_i}{x_i}$
17	1	0,035	$\prod_{i=1}^{18} \ln(x_i^2 + 2x_i - 0.55)$

N_2	X_i	h_x	y
18	0	0,15	$\sum_{i=1}^{16} \frac{\ln\left(1 + \frac{\cos x_i}{2}\right)}{\cos(x_i - 1)}$
19	5	0,13	$\prod_{i=1}^{11} (0.5 + \log_3 x_i^2)$
20	0,1	0,01	$\sum_{i=1}^{17} \frac{(\ln(tg^2 0.4x_i))}{2 + x_i^2}$
21	π	0,06	$\prod_{i=1}^9 \log_2 x_i^2 + tg x_i $
22	$\pi/8$	0,05	$\sum_{i=1}^{17} \frac{\sin(0.8x_i)}{x_i} \ln \frac{1}{x_i}$
23	1	0,15	$\sum_{i=1}^{17} \frac{x_i \ln\left(\frac{1}{x_i}\right)}{\sqrt{1 + x_i^2}}$
24	2,1	0,2	$\prod_{i=1}^{15} \log_2 \left(e^{x_i} / (x_i^2 - 1) \right)$
25	0	0,134	$\sum_{i=1}^{17} \frac{2 \log_2 (1 + x_i^2)}{1 + x_i^2}$
26	-10	0,05	$\prod_{i=1}^{13} \frac{1.25 \cdot \sqrt[5]{ x_i }}{\arctg x_i}$
27	0,1	0,035	$\sum_{i=1}^{21} \left(x_i + \cos \left(\ln \left tg \frac{x_i}{2} \right \right) \right)$
28	-10,5	0,035	$\prod_{i=1}^{17} e^{\sin x_i} (\sqrt[5]{ x_i } - 1)$
29	$\pi/8$	0,156	$\sum_{i=1}^{17} \frac{\sin(0.8x_i)}{x_i^2} \lg x_i$
30	2,1	0,15	$\prod_{i=1}^{15} \log_5 \left(x_i / (x_i^2 - 1) \right)$

6 Массивы в VB

Часто при разработке приложений возникает необходимость обработки больших объемов данных, причем в большинстве случаев, данные являются однотипными, а обработка сводится к выполнению однотипных операций над ним.

Для эффективной работы с такими данными применяются особые структуры данных – массивы.

Массив – это **индексированный набор однотипных значений**. Подобно переменной, позволяющей хранить одно значение заданного типа, массив имеет имя. Однако с данным именем связано определенное количество значений. Для того чтобы получить требуемое в текущий момент значение недостаточно указать одно лишь имя, как в случае с переменной. Для четкого указания конкретного значения из набора необходимо дополнительно указать индекс значения в именованном наборе значений массива. Рассмотрим пример, представленный на следующем рисунке:

Массив А

Индексы элементов:	0	1	2	3	4	5	6	7	8	9
Значения элементов:	5	4	6	3	2	6	1	0	9	3

Представленный на рисунке массив с именем А содержит десять значений называемых элементами. Каждый элемент массива А имеет **индекс** (номер), по которому можно производить обращение к нему. Например, элемент массива А с индексом 6 имеет значение 1, а элемент массива А с индексом 2 значение 6.

Для обращения к элементу массива в VB используется следующая конструкция: <Имя массива>(<Индекс элемента массива>). Каждый конкретный элемент массива, как и переменная, доступен во время исполнения для чтения и для записи значений. Например, при выполнении следующей строки исходного кода:

```
A(1) = A(6) + A(2)
```

При выполнении оператора присваивания будет вычислено значение

выражения $A(6) + A(2)$. При вычислении данного выражения будут прочитаны значения элементов массива с индексами 6 и 2 и найдена их сумма. Таким образом, значение выражения $A(6) + A(2) = 1 + 6 = 7$. Вычисленный результат будет присвоен элементу массива A с индексом 1. При этом будет произведена запись вычисленного значения 7 в область памяти связанную с элементом массива A с индексом 1. После выполнения данной строки массив будет содержать следующие значения:

Массив A

Индексы элементов:	0	1	2	3	4	5	6	7	8	9
Значения элементов:	5	7	6	3	2	6	1	0	9	3

Массивы объявляются с использованием конструкций, подобных конструкциям объявления переменных и могут иметь те же области видимости, что и переменные.

Синтаксис конструкции объявления массива:

```
{Static|Public|Private|Dim} <ИмяМассива>([[<Нижняя граница1> To <Верхняя граница1>], >([<Нижняя граница2> To <Верхняя граница2>[,...]])] [As Тип]
```

Здесь

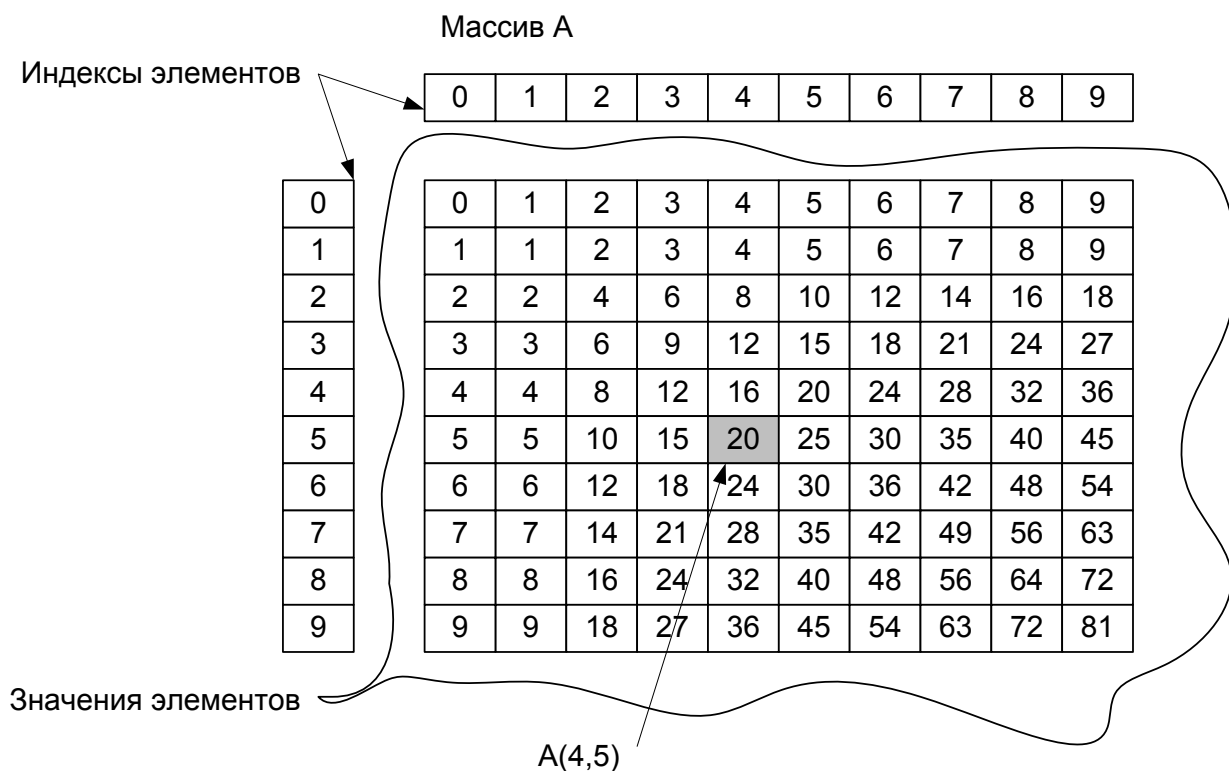
<Нижняя границаN> - это значение индекса, с которого начинают нумеровать элементы массива;

<Верхняя границаN> - это значение индекса, с которого начинают нумеровать элементы массива;

N – это номер размерности.

Если параметр «Нижняя граница» не указан, то по умолчанию массив индексируется с 0. Однако при необходимости значение по умолчанию для данного параметра может быть изменено с помощью опции интерпретатора Option Base. Найдите описание и ознакомьтесь с синтаксисом данной опции (Приложение: Операторы VB).

В общем случае массив может быть многомерным. Так представленный на следующем рисунке массив является двухмерным:



На рисунке представлен двухмерный массив, в котором хранится таблица умножения. Элементы такого массива «располагаются» в пересечениях соответствующих столбцов и строк. Причем нет принципиальной разницы в том, чтобы считать первым индексом (координатой) номер столбца или строки. Однако, для того чтобы исключить бесконечные уточнения принятой в каждом конкретном случае «системы координат», условимся, что первый индекс – это номер строки, а второй – столбца. На рисунке предыдущего примера выделен серым цветом элемент с индексами 4,5, что следует понимать как элемент 4-й строки и 5-го столбца.

В чем преимущество массивов по сравнению с переменными? Во-первых, для хранения таблицы умножения требуется создать 100 переменных или один массив, состоящий из 100 элементов. То есть при использовании переменных для хранения такого объема данных требуется использовать на 99, а в общем случае на $N-1$, где N – количество хранимых значений, конструкций больше, чем при использовании массивов. Но основное преимущество массивов состоит даже не в том, что при их использовании появляется возможность автоматизации обработки данных, которая

позволяет значительно сократить требуемый для описания необходимых последовательностей действий объем исходного кода. При использовании массивов данных также становится возможной разработка компактного кода, позволяющего обрабатывать не строго определенное количество значений, а то которое требуется в контексте текущей задачи, расходуя столько оперативной памяти, сколько требуется для ее решения. Например, одно и то же приложение может обрабатывать в одном случае 50 значений, в другом 100, а если потребуется, то 1000 или более.

Возможность автоматизированной обработки появляется благодаря синтаксису конструкции обращения к элементу массива. Как вы помните для обращения к конкретному элементу массива необходимо указать его имя и в скобках индекс. Индекс элемента массива задается выражением, то есть его можно задать неименованной константой, как в тех примерах, которые мы рассматривали ранее, именованной константой, переменной или более сложным выражением. Рассмотрим примеры таких обращений к элементам массива:

$A(2)$ – индекс элемента задан неименованной константой;

$A(t)$ – индекс элемента задан переменной;

$A(t+1)$ – индекс элемента задан выражением $t+1$.

Таким образом, становится уместным использование для индексирования массивов переменных счетчиков. Рассмотрим пример такого исходного кода:

...

Dim A(0 To 10) As Byte

Dim I as Byte

For I = 0 To 10

$A(I) = I$

Next

...

В данном исходном коде переменная счетчик используется в качестве

индекса массива A. Разработайте приложение, в котором данный исходный код включен в обработчик события «Загрузка формы», выполните его отладку и оформите отчет о проделанной работе по следующей форме:

- исходный код приложения с подробными комментариями применяемых в строках исходного кода конструкциях;

- список строк, исполненных интерпретатором при выполнении приложения, с комментариями, в которых описываются производимые при исполнении строки действия, состояния переменных до и после исполнения строки. (Для строк, где производится изменение значений элементов массива, необходимо записывать значения всех элементов до и после изменения в табличной форме подобной рисункам примеров приведенных выше.) Приведите заключение о том, что производится с массивом в данном исходном коде;

- вывод об эффективности применения массивов. (В выводе следует привести исходный код, выполняющий подобную обработку одиннадцати значений, при использовании для их хранения переменных. Численно оценить эффективность применения массивов через требуемое количество строк исходного кода при реализации подобных задач с использованием переменных и массивов для хранения и обработки данных.)

Рассмотрим подробнее эффективное использование памяти при решении однотипных задач над различными объемами данных. Вернитесь к описанию синтаксиса конструкции объявления массива и обратите внимание на пару квадратных скобок, в которые заключены описания размерностей массива. Они показывают, что при объявлении массива границы размерностей могут не описываться. То есть вполне уместна следующая конструкция объявления массива:

`Dim A() As Byte`

Такая конструкция позволяет определить массив без указания границ, сообщить интерпретатору, что в приложении будет использоваться массив с именем A, но границы массива будут заданы позже. Массив, для которого

границы не заданы на стадии объявления, называется **динамическим** (он может динамически изменять количество хранимых элементов по требованию). Изменение границ динамического массива производится с помощью специальной конструкции **Redim**. Найдите описание и ознакомьтесь с синтаксисом данной конструкции (Приложение: Операторы VB).

Благодаря применению динамических массивов вы можете разрабатывать приложения, которые в каждый момент времени занимают только такой объем оперативной памяти, какой им требуется для работы. Например, при разработке приложения, выполняющего создание массива с количеством элементов, задаваемых пользователем во время исполнения приложения, разработчик может эффективно использовать динамический массив. Действительно на стадии разработки нельзя заранее знать размер массива, требуемого пользователю при решении задачи. При использовании статического массива разработчику потребовалось бы оценить решаемую задачу и задать границы массива максимально возможными, а при дальнейшем исполнении приложения использовать лишь те элементы массива, которые необходимы. Если пользователю требуется меньшее количество элементов, чем указано при объявлении статического массива, то все невостребованные элементы также будут занимать оперативную память.

Самостоятельно разработайте приложение создающее массив, индексированный с 1 и с количеством элементов заданным пользователем. Задайте элементам массива значения равные их индексам и выведите значения элементов массива в текстовое поле в формате «A(1)=1, A(2)=2, ...».

В отчете приведите эскиз формы, блок-схему приложения и исходный код.

Модифицируйте приложение таким образом, чтобы массив заполнялся случайными числами, а в дополнительное текстовое поле выводился отчет о сумме значений элементов массива. Дополните отчет.

Разработайте приложение, выполняющее создание двумерного массива размерностью 10×10 , заполненного значениями, равными произведениям индексов соответствующих элементов (таблица умножения). Доработайте приложение таким образом, чтобы по требованию пользователя для указанной им строки матрицы выполнялся расчет суммы элементов.

7 Алгоритмы сортировки

В данной главе рассматриваются различные алгоритмы сортировки.

7.1 Алгоритм сортировки выборкой

Алгоритм сортировки выборкой достаточно прост в реализации, однако при обработке больших объемов данных его эффективность не приемлемо снижается.

Суть алгоритма заключается в последовательном поиске минимальных элементов из оставшихся неотсортированных и обмен местами минимального и текущего сортируемого элемента.

За текущий сортируемый элемент принимаются все элементы, начиная с первого по предпоследний элементы массива. Для каждого текущего сортируемого элемента осуществляется поиск минимального среди оставшихся неотсортированных элементов (под оставшимися неотсортированными здесь понимаются все элементы, которые ранее не были приняты за текущий сортируемый элемент, то есть все элементы, начиная со следующего за текущим сортируемым по последний элемент массива).

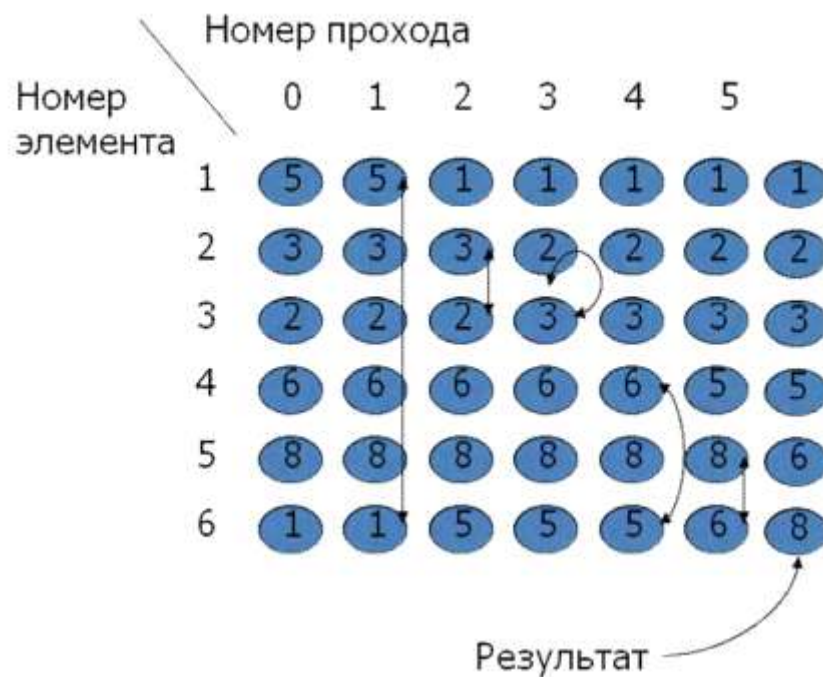
Поиск минимального среди оставшихся неотсортированных осуществляется следующим образом:

- за текущий минимальный элемент принимается текущий сортируемый элемент массива;
- значение текущего минимального элемента массива последовательно сравнивается со всеми оставшимися неотсортированными элементами массива (элементами начиная со следующего по последний). Если при сравнении значения текущего минимального элемента, с каким либо из рассматриваемых оказывается, что текущее минимальное значение больше чем имеет рассматриваемый неотсортированный элемент, тогда за текущий минимальный элемент принимается рассматриваемый неотсортированный. Таким образом, после просмотра всех неотсортированных элементов

текущим минимальным элементом будет элемент, имеющий минимальное значение;

- далее значения найденного минимального и текущего сортируемого элементов меняются местами;
- осуществляется переход к следующему сортируемому элементу (следующий элемент принимается за текущий сортируемый и действия по обработке текущего сортируемого элемента повторяются).

Рассмотрим сортировку выборкой для следующего массива:
[5;3;2;6;8;1].



При начале выполнения первого прохода в качестве текущего сортируемого элемента принимается первый элемент массива (элемент с индексом 1 и значением 5). Данный элемент принимается за текущий минимальный. Начинается выполнение поиска минимального среди оставшихся неотсортированных элементов массива. При поиске минимального среди оставшихся неотсортированных элементов массива значение текущего минимального элемента 5 сравнивается со значением следующего элемента, имеющего значение 3. При этом элемент с индексом 2 и значением 3 оказывается меньшим. В данном случае за текущий

минимальный элемент принимается элемент с индексом 2 и его значение далее последовательно сравнивается с оставшимися неотсортированными элементами. Так на следующем «шаге» алгоритма сортировки выборкой значение текущего минимального элемента сравнивается со значением следующего неотсортированного элемента (элемент с индексом 3 и значением 2). Оно вновь оказывается меньшим, чем значение текущего минимального элемента, поэтому за текущий минимальный элемент принимается элемент с меньшим значением 2. Те же действия производятся со всеми оставшимися неотсортированными элементами (выполните самостоятельно действия для всех оставшихся неотсортированных элементов). Таким образом, после просмотра всех оставшихся неотсортированных элементов текущим минимальным (минимальным среди всех просмотренных) элементов является элемент с индексом 6 и значением 1. Значения данного элемента и текущего сортируемого меняются местами (см. рисунок). После выполнения данных действий можно с уверенностью сказать, что сортировка для первого элемента массива завершена и в первой позиции массива помещено значение минимально элемента массива.

Далее за текущий сортируемый элемент принимается элемент с индексом 2 и значением 3, повторяются действия по поиску минимального среди оставшихся неотсортированных элементов и найденный минимальный элемент (с индексом 3 и значением 2) меняется местами с текущим сортируемым. Затем действия последовательно выполняются со всеми элементами до предпоследнего элемента массива (самостоятельно ответьте на вопрос: почему нет необходимости принимать за текущий сортируемый элемент последний элемент массива?).

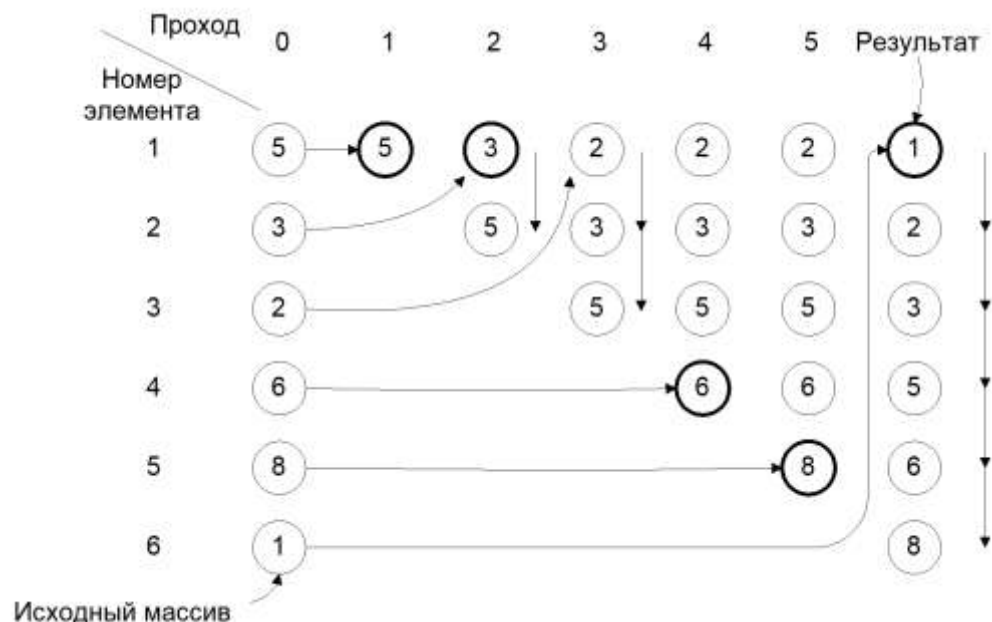
Задание: самостоятельно разработайте приложение выполняющее сортировку массива заданной пользователем размерности, заполненного случайными числами из указанного пользователем интервала. В отчете по работе приведите исходный код приложения, прокомментируйте действия выполняемые приложением, приведите блок-схему алгоритма.

7.2 Алгоритм сортировки вставкой

Алгоритм сортировки вставкой также прост в реализации и эффективен только при обработке небольших массивов данных.

Суть алгоритма заключается в последовательном извлечении элементов сортируемого массива и вставке их в результирующий массив в такую позицию, чтобы в каждый момент времени результирующий массив содержал отсортированные значения.

Рассмотрим пример исполнения алгоритма сортировки вставкой массива [5;3;2;6;8;1] (см. рисунок).



На первом проходе из сортируемого массива извлекается первый элемент и помещается в первую позицию результирующего массива. После выполнения данной операции результирующий массив содержит один элемент и является отсортированным.

При выполнении второго прохода из сортируемого массива извлекается второй элемент. В соответствии с требованиями алгоритма данный элемент должен быть вставлен в такую позицию, чтобы результирующий массив был отсортирован. Для этого вставляемый элемент сравнивается с первым элементом результирующего массива. Если вставляемый элемент оказывается меньшим, то элементы результирующего

массива сдвигаются на одну позицию вниз, а он вставляется в освободившуюся позицию. Если значение вставляемого элемента оказывается большим, чем значение первого элемента, то осуществляется переход к следующей позиции, снова производится сравнение, сдвиг элементов или переход к следующей позиции до тех пор, пока вставленные к текущему моменту значения не закончатся. При достижении такой («последней») позиции элемент вставляется в нее, так как в данном случае он является максимальным среди рассмотренных элементов. В рассматриваемом примере второй элемент сортируемого массива имеет значение 3. Данное значение необходимо сравнить со значением вставленного на предыдущем проходе элемента. Вычислим логическое выражение $3 < 5$. Его результат ИСТИНА. Вставляемый элемент имеет меньшее значение и должен быть расположен в результирующем массиве до элемента со значением 5, следовательно, необходимо сместить все ранее вставленные элементы на одну позицию вниз, а данный элемент вставить в освободившуюся позицию. К текущему моменту в массив было вставлено только одно значение, то есть необходимо сместить только значение 5. Смещение выполняется копированием на одну позицию вниз значений элементов: начиная с последнего из имеющихся в текущий момент в результирующем массиве до того элемента, в позицию которого необходимо осуществить вставку. То есть для вставки элемента со значением 3 в первую позицию присваиваем значение единственного ранее вставленного элемента со значением 5 в позицию 2. Значение вставляемого элемента присваиваем первому элементу массива. После выполнения данного прохода в результирующем массиве содержатся два элемента 3 и 5. Результирующий массив является отсортированным.

Аналогично выполняются следующие проходы:

- на третьем проходе элемент со значением 2 вставляется в первую позицию, ранее вставленные элементы смещаются на одну позицию вниз;

- элементы со значением 6 оказывается большим, чем все ранее вставленные элементы. Он вставляется в следующую за последним из ранее вставленных элементов результирующего массива позицию;
- элемент со значением 8 также вставляется в последнюю позицию;
- элемент со значением 1 оказывается меньшим, чем элемент со значением 3. Он вставляется в первую позицию результирующего массива.

Задание: самостоятельно разработайте приложение выполняющее сортировку вставкой массива заданной пользователем размерности, заполненного случайными числами из указанного пользователем интервала. В отчете по работе приведите исходный код приложения, прокомментируйте действия выполняемые приложением, приведите блок-схему алгоритма.

7.3 Алгоритм пузырьковой сортировки

7.4 Алгоритм быстрой сортировки

7.5 Алгоритм сортировки слиянием

7.6 Алгоритм сортировки подсчетом

8 Приложение: Коды ошибок VB

№	Название	Описание
3	Return without GoSub	Оператор Return без GoSub
5	Invalid procedure call	Неверный вызов процедуры
6	Overflow	Переполнение
7	Out of memory	Не хватает памяти
9	Subscript out of range	Индекс массива вышел за допустимые пределы
10	This array is fixed or temporarily locked	Массив зафиксирован или временно недоступен
11	Division by zero	Деление на ноль
13	Type mismatch	Неверный тип
14	Out of string space	Нет места для строки
16	Expression too complex	Выражение слишком сложное
17	Can't perform requested operation	Не могу выполнить запрашиваемую операцию
18	User interrupt occurred	Произошло прерывание по указу пользователя
20	Resume without error	Оператор Resume без ошибки
28	Out of stack space	Нет места для стека
35	Sub, Function, or Property not defined	Процедура, функция или свойство не определено
47	Too many DLL application clients	Слишком много клиентов DLL приложения
48	Error in loading DLL	Ошибка при загрузке DLL
49	Bad DLL calling convention	Неверный вызов DLL
51	Internal error	Внутренняя ошибка
52	Bad file name or number	Неверное имя файла или неверный номер

№	Название	Описание
53	File not found	Файл не найден
54	Bad file mode	Неверный тип доступа к файлу
55	File already open	Файл уже открыт
57	Device I/O error	Ошибка устройства ввода вывода
58	File already exists	Файл уже существует
59	Bad record length	Неверный размер записи
61	Disk full	Диск переполнен
62	Input past end of file	Чтение файла не возможно, т.к. достигнут его конец
63	Bad record number	Неверный номер записи
67	Too many files	Слишком много файлов
68	Device unavailable	Устройство не доступно
70	Permission denied	Доступ запрещён
71	Disk not ready	Диск не готов
74	Can't rename with different drive	Нельзя переименовать файл на другой носитель (т.е. оператором Name)
75	Path/File access error	Ошибка доступа к файлу или каталогу
76	Path not found	Каталог не найден
91	Object variable or With block variable not set	Ссылка на объект или блок оператора With не установлен
92	For loop not initialized	Не найдено начало цикла
93	Invalid pattern string	Неверная маска
94	Invalid use of Null	Неверное использование Null
97	Can't call Friend procedure on an object that is not an instance of the defining class	Не могу вызывать процедуру Friend не являющуюся экземпляром класса

№	Название	Описание
98	A property or method call cannot include a reference to a private object, either as an argument or as a return value (Error 98)	Обращение к свойству или методу не может включать ссылку на Private объект. Этот объект также не может быть в качестве аргумента или возвращаемого значения.
298	System DLL could not be loaded	Системная DLL не может быть загружена
320	Can't use character device names in specified file names	Невозможно использовать в качестве имени файла название порта
321	Invalid file format	Неверный формат файла
322	Can't create necessary temporary file	Не могу создать необходимый временный файл
325	Invalid format in resource file	Неверный формат файла ресурсов
327	Data value named not found	Значение не может быть найдено
328	Illegal parameter; can't write arrays	Недопустимый параметр; не могу записать массив
335	Could not access system registry	Не могу получить доступ к реестру
336	ActiveX component not correctly registered	ActiveX компонент неправильно зарегистрирован
337	ActiveX component not found	ActiveX компонент не найден
338	ActiveX component did not run correctly	ActiveX компонент не правильно запущен (?)
360	Object already loaded	Объект уже загружен
361	Can't load or unload this object	Не могу загрузить или выгрузить объект

№	Название	Описание
363	ActiveX control specified not found	ActiveX компонент не найден
364	Object was unloaded	Объект был выгружен
365	Unable to unload within this context	В данном случае невозможно выгрузить объект
368	The specified file is out of date. This program requires a later version	Данный файл устарел. Эта программа требует более новой версии
371	The specified object can't be used as an owner form for Show	Данный объект не может быть использован как родитель формы для её показа
380	Invalid property value	Неверное значение свойства
381	Invalid property-array index	Неверный индекс свойства-массива
382	Property Set can't be executed at run time	Процедура установки свойства не может быть запущена во время работы программы
383	Property Set can't be used with a read-only property	Процедура установки свойства не может быть использована для свойств, доступных только для чтения
385	Need property-array index	Необходим индекс для свойства-массива
387	Property Set not permitted	Процедура установки свойства не разрешена
393	Property Get can't be executed at run time	Процедура чтения свойства не может быть запущена во время работы программы
394	Property Get can't be executed on	Процедура чтения свойства не

№	Название	Описание
	write-only property	может быть использована для свойств, доступных только для записи
400	Form already displayed; can't show modally	Форма уже показана; не могу показать форму модально
402	Code must close topmost modal form first	Код должен сначала закрыть модальную форму самого высшего уровня
419	Permission to use object denied	Запрещено использование данного объекта
422	Property not found	Свойство не найдено
423	Property or method not found	Свойство или метод не найден
424	Object required	Необходим объект
425	Invalid object use	Неверное использование объекта
429	ActiveX component can't create object or return reference to this object	ActiveX компонент не может создать объект или вернуть ссылку на этот объект
430	Class doesn't support Automation	Класс не поддерживает технологию Automation
432	File name or class name not found during Automation operation	Имя файла или класса не найдено в процессе операции Automation
438	Object doesn't support this property or method	Объект не поддерживает данное свойство или метод
440	Automation error	Ошибка Automation
442	Connection to type library or object library for remote process has been lost	Связь с библиотекой типов или библиотекой объектов для удалённого процесса была утеряна

№	Название	Описание
443	Automation object doesn't have a default value	Automation объект не имеет значения по умолчанию
445	Object doesn't support this action	Объект не поддерживает данную операцию
446	Object doesn't support named arguments	Объект не поддерживает названные аргументы
447	Object doesn't support current locale setting	Объект не поддерживает текущие локальные настройки
448	Named argument not found	Названный аргумент не найден
449	Argument not optional or invalid property assignment	Аргумент обязателен или неверное присваивание свойству
450	Wrong number of arguments or invalid property assignment	Неверное количество аргументов или неверное присваивание свойству
451	Object not a collection	Объект не является коллекцией
452	Invalid ordinal	Неверное числительное
453	Specified DLL function not found	Данная функция DLL не найдена
454	Code resource not found	Не найден код ресурса
457	This key is already associated with an element of this collection	Этот ключ уже ассоциирован с элементом коллекции
458	Variable uses a type not supported in Visual Basic	Переменная использует неподдерживаемый Visual Basic'ом тип
459	This component doesn't support the set of events	Невозможно использовать оператор WithEvents и Implements.
460	Invalid Clipboard format	Неверный формат буфера обмена
461	Specified format doesn't match format	Данный формат не совпадает с

№	Название	Описание
	of data	форматом данных
480	Can't create AutoRedraw image	Не могу создать AutoRedraw изображение
481	Invalid picture	Неверное изображение
482	Printer error	Ошибка принтера
483	Printer driver does not support specified property	Драйвер принтер не поддерживает данное свойство
484	Problem getting printer information from the system. Make sure the printer is set up correctly	Проблема при чтении информации принтера в системе. Убедитесь, что принтер правильно установлен
485	Invalid picture type	Неверный тип изображения
486	Can't print form image to this type of printer	Не могу распечатать изображение формы на таком типе принтера
520	Can't empty Clipboard	Не могу очистить буфер обмена
521	Can't open Clipboard	Не могу открыть буфер обмена
735	Can't save file to TEMP directory	Не могу сохранить файл в директорию TEMP (временную)
744	Search text not found	Искомый текст не найден
746	Replacements too long	Текст для замещения слишком длинный
31001	Out of memory	Не хватает памяти
31004	No object	Нет объекта
31018	Class is not set	Класс не установлен
31027	Unable to activate object	Не могу активизировать объект
31032	Unable to create embedded object	Не могу создать внедрённый объект
31036	Error saving to file	Ошибка записи в файл

№	Название	Описание
31037	Error loading from file	Ошибка чтения из файла

9 Описание событий VB

Событие	Причина возникновения
Activate, Deactivate	<p>Это событие имеет только форма.</p> <p>Activate - вызывается в тот момент, когда форма становится активной (получает фокус). Однако если перейти к другому приложению Windows, а затем вернуться к своему, то это событие не произойдет. Оно срабатывает только при переключении между формами внутри программы.</p> <p>Deactivate - Событие, обратное событию Activate. Вызывается при потере фокуса формы.</p>
Change	<p>ComboBox — меняется текст в текстовой части элемента. Происходит только тогда, когда свойство Style установлено в 0 (Dropdown Combo) или 1 (Simple Combo) и пользователь изменяет текст или вы меняете его в коде программы.</p> <p>DirListBox — Меняется выбранная директория. Происходит, когда пользователь выполняет двойной щелчок кнопкой мыши на новой директории или когда меняется свойство Path в коде.</p> <p>DriveListBox — Меняется выбранное устройство. Происходит, когда пользователь меняет устройство, выбрав его из списка, или когда меняется свойство Drive в коде.</p> <p>HScrollBar и VScrollBar (горизонтальная и вертикальная прокрутки) — Подвинулся ползунок полосы прокрутки. Происходит, когда пользователь передвинул и отпустил полосу прокрутки или меняется свойство Value в коде.</p> <p>Label — Меняется содержимое метки. Происходит когда меняется свойство Caption в коде.</p> <p>PictureBox — Меняется содержимое PictureBox. Происходит при смене свойства Picture. (а также при использовании</p>

Событие	Причина возникновения
	<p>LoadPicture, прим.eax)</p> <p>TextBox — Меняется текст в текстовом поле. Происходит при смене текста пользователем или при смене свойства Text в коде.</p>
Click	<p>Происходит когда пользователь нажимает и отпускает кнопку мышки над объектом. Оно также может происходить при смене некоторого значения объекта.</p> <p>Для формы такое событие выполняется при клике на свободном месте формы, или по отключенному элементу управления (т.е. когда его Enabled = False).</p> <p>Вообще событие происходит для правой кнопки мыши и для левой. Но для элементов CheckBox, CommandButton, Listbox, и OptionButton происходит только при нажатии левой кнопки мыши.</p> <p>Для ComboBox или ListBox оно происходит и при клике кнопкой мышки и также при смене текущего элемента клавишами курсора.</p> <p>Происходит при нажатии на "ПРОБЕЛ" у элементов CommandButton, OptionButton, или CheckBox, когда те имеют фокус.</p> <p>При нажатии на ENTER при фокусе на элементе CommandButton и когда установлено свойство Default.</p> <p>Происходит при нажатии на ESC когда форма имеет Cancel кнопку - CommandButton с установленным свойством Cancel.</p> <p>Также происходит при нажатии на горячую последовательность. Например, если кнопка имеет Caption - "&Go", то при нажатии Alt+G запуститься событие.</p> <p>Также, событие Click может быть сгенерировано в</p>

Событие	Причина возникновения
	следующих случаях в коде: Установка значения Value для OptionButton и CheckBox.
DbClick	<p>Происходит при двойном клике по объекту.</p> <p>Для формы происходит при двойном клике по форме, а также по отключенному объекту.</p> <p>Для других элементов:</p> <p>Двойной клик по объекту левой кнопкой.</p> <p>Двойной клик по элементу в ComboBox, когда Style = 1. Или также в FileListBox, ListBox, DBCombo, или DBList.</p>
DragDrop	<p>Происходит при завершении операции перетаскивания (Drag&Drop). В обработку события передаются 3 аргумента - координаты курсора (X,Y), где был отпущен объект, и ссылка на объект (Source), который был перетащен.</p>
DragOver	<p>Происходит когда объект перетаскивается над получателем, но кнопка ещё не отпущена. Имеет 4 параметра. Координаты курсора, ссылка на объект, и текущее состояние (State As Integer):</p> <p>0 = Enter (вошёл) (источник вошёл в область объекта).</p> <p>1 = Leave (покинул) (источник ушёл из этой области).</p> <p>2 = Over (над) (произошёл сдвиг в пределах области).</p>
Error	<p>Происходит только при работе с базами данных, при ошибке в доступе к данным при выполнении кода.</p> <p>Имеет 2 параметра:</p> <p>dataerr - номер произошедшей ошибки</p> <p>response - номер, соответствующий выбранному в настройках (Settings):</p> <p>Если vbDataErrContinue, то response = 0 (Продолжить)</p> <p>Если vbDataErrDisplay, то response = 1 (Default) Показать</p>

Событие	Причина возникновения
	сообщение об ошибке.
GotFocus	Происходит, когда объект получает фокус, или при нажатии кнопки Tab или кликом по объекту, а также при запуске метода SetFocus в программе. Форма получает фокус только тогда, когда все видимые элементы отключены (Enabled = False).
Initialize	Это событие имеет только форма. Обрабатывается первым и один раз. Visual Basic вызывает его при первом создании формы. Здесь обычно размещают код, для инициализации значений переменных в программе.
ItemCheck	Происходит, когда у ListBox контрола свойство Style установлено в 1 (checkboxes) и выбран или сброшен флажок у какого-либо элемента (item) в контроле. Передаётся один параметр: index - номер элемента, который был выбран в ListBox.
KeyDown, KeyUp	Происходит, когда пользователь нажимает (KeyDown) или отпускает (KeyUp) клавишу, в то время как объект имеет фокус. Чтобы получить код клавиши, используйте событие KeyPress. Параметров нет.
KeyPress	Происходит, когда пользователь нажимает и отпускает клавишу на клавиатуре. Событие имеет один параметр: keyascii - код нажатой клавиши. Например, если нажать на клавишу "1" (основного ряда), то keyascii будет равен 49. Если нажать ESC, то 27, и и.д. Если присвоить этой переменной 0, то нажатая буква (символ) не появится в текстовом поле. Иногда это бывает очень удобным. Например, можно сделать так, чтобы в

Событие	Причина возникновения
	<p>текстовое поле можно было вводить только цифры. Вот пример:</p> <p>Таблица кодов клавиш представлена в п. 10.</p>
LinkClose LinkError LinkNotify LinkOpen	<p>LinkClose: Происходит когда DDE соединение закрывается. Может произойти в любое время.</p> <p>LinkError: Происходит при возникновении ошибки в ходе связи DDE.</p> <p>LinkNotify: Происходит когда у источника меняются данные, на которые установлена ссылка DDE.</p> <p>LinkOpen: Происходит при создании инициализации DDE связи с источником.</p> <p>Данные события используются очень редко.</p>
Load	<p>Это событие имеет только форма.</p> <p>Обрабатывается при загрузке формы в память. Происходит после события Initialize. Обычно код выполняется один раз.</p> <p>При запуске программы, это событие обрабатывается автоматически для той формы, которая загружается первой.</p> <p>Это событие может выполняться несколько раз. Т.е. если вы выгрузите форму оператором Unload, а затем вновь загрузите оператором Load или методом Show, то это событие будет выполнено.</p>
LostFocus	<p>Происходит при потере объектом фокуса, или при нажатии Tab пользователем, или при использовании метода SetFocus для другого объекта.</p>
MouseDown, MouseUp	<p>Происходят когда пользователь нажимает (MouseDown) или отпускает (MouseUp) кнопку мыши.</p> <p>Имеют 4 передаваемых параметра:</p> <p>button - содержит integer - номер кнопки мыши. С помощью</p>

Событие	Причина возникновения
	<p>этого события можно определить, какую кнопку нажал пользователь. Для этого используются константы: vbLeftButton (левая), vbMiddleButton (средняя), vbRightButton (правая). Проверить можно примерно так:</p> <p>If button = vbRightButton Then ...</p> <p>shift - содержит integer - указывающее на состояние клавиш Shift, Alt и Ctrl. Бит 0 - Shift, бит 1 - Ctrl, бит 2 - Alt. Для определения факта нажатия клавиш Ctrl и Alt можно использовать такой код:</p> <p>If (Shift And (vbCtrlMask Or vbAltMask)) = (vbCtrlMask Or vbAltMask) Then ...</p> <p>Т.е. мы проверяем содержит ли переменная Shift биты 1 и 2 (оператором And). Скобки везде обязательны. vbShiftMask, vbAltMask, vbCtrlMask - это обычные константы, содержащие маски битов:</p> <p>vbShiftMask = 1 (нулевой бит) Bin: 00000001</p> <p>vbCtrlMask = 2 (первый бит) Bin: 00000010</p> <p>vbAltMask = 4 (второй бит) Bin: 00000100</p> <p>x, y - координаты курсора мыши того места, где произошло событие MouseUp или MouseDown. Координаты всегда зависят от координатной системы, задаваемой свойствами ScaleHeight, ScaleWidth, ScaleLeft, ScaleTop объекта.</p>
MouseMove	<p>Происходит, когда курсор мыши изменяет своё положение над объектом. Т.е. когда курсор стоит на месте, событие не происходит. При каждом сдвиге курсора - срабатывает.</p> <p>Параметры те же, что и у MouseDown и MouseUp.</p>
ObjectMove	<p>Происходит немедленно после того, как OLE контейнер сдвигается или меняет размеры пока объект активен.</p>

Событие	Причина возникновения
	<p>Передаваемые параметры:</p> <p>left - координата левой грани OLE контейнера</p> <p>top - координата верхней грани OLE контейнера</p> <p>width - ширина OLE контейнера</p> <p>height - высота OLE контейнера</p>
OLECompleteDrag	<p>Происходит когда компонент - источник "брошен" на компонент - цель, информируя объект о том, что либо завершён, либо отменён процесс перетаскивания.</p> <p>Имеет один параметр:</p> <p>effect - привожу оригинал: A long integer set by the source object identifying the action that has been performed, thus allowing the source to take appropriate action if the component was moved (such as the source deleting data if it is moved from one component to another). The possible values are listed in Settings.</p> <p>A Settings может быть:</p> <p>vbDropEffectNone = 0 - операция "бросания" (Drop) была отменена.</p> <p>vbDropEffectCopy = 1 - показывать значок копирования данных.</p> <p>vbDropEffectMove = 2 - "сбрасываемые" данные являются ссылкой на оригинальные данные.</p>
OLEDragDrop	<p>Это событие происходит, когда на объект сбрасываются OLE данные. Например, происходит при перетаскивании на объект файлов из проводника, или перетаскивании изображения из Internet Explorer и т.п.</p> <p>Замечание: Это событие происходит только в том случае, когда свойство OLEDropMode установлено в 1 (Manual).</p>

Событие	Причина возникновения
	<p>Имеет очень много параметров:</p> <p>data - объект типа DataObject. Имеет 4 метода и одно свойство - коллекцию файлов, перетаскиваемых на объект. Для получения данных можно использовать метод GetData. Чтобы узнать формат данных, перекинутых на объект можно использовать метод GetFormat.</p> <p>effect Long - привожу оригинал - set by the target component identifying the action that has been performed (if any), thus allowing the source to take appropriate action if the component was moved (such as the source deleting the data). The possible values are listed in Settings.</p> <p>button - то же, что и в событии MouseDown(Up).</p> <p>shift - то же, что и в событии MouseDown(Up).</p> <p>x,y - то же, что и в событии MouseDown(Up).</p> <p>Пример. Обработаем переташенные на форму файлы. Не забудьте установить свойство OLEDropMode в 1.</p> <pre> Private Sub Form_OLEDragDrop _ (Data As DataObject, Effect As Long, _ Button As Integer, Shift As Integer, _ X As Single, Y As Single) ' перетаскиваются файлы? If Data.GetFormat(vbCFFiles) = True Then Dim c As Long ' пройдемся по всей коллекции For c = 1 To Data.Files.Count ' покажем имя переташенного файла MsgBox "Был переташен файл: " & Data.Files(c) Next c </pre>

Событие	Причина возникновения
	End If End Sub
OLEDragOver	Происходит при перетаскивании над объектом. Параметры - комбинация параметров событий OLEDragDrop и обычного DragOver.
OLEGiveFeedback	<p>Происходит после каждого события OLEDragOver. OLEGiveFeedback позволяет источнику обеспечивать визуальную "отдачу" пользователю, такое, как изменение курсора мыши (посмотрите на него при перетаскивании файлов в проводнике с нажатым Ctrl, - видите значок "+", вот это и есть та "отдача"), которое будет говорить пользователю о том, что происходит.</p> <p>Параметры - effect и defaultcursors - boolean - разрешает или запрещает использование курсора по умолчанию. Если True - то используется курсор по умолчанию, если False, то используется курсор, указанный пользователем в свойстве MousePointer объекта Screen.</p>
OLEStartDrag	<p>Происходит тогда, когда начинается перетаскивание (т.е. когда перетаскиваемый объект (файл, например) появляется над объектом приёмником). Имеет два параметра:</p> <p>data - то же что и у события OLEDragDrop.</p> <p>allowedeffects - Long - содержит поддерживаемые источником эффекты. Возможные значения описаны в Settings.</p>
Paint	Происходит когда часть или весь объект появляется на экране после сдвига или изменения размера. Событие происходит также при сдвиге окна, которое закрывает

Событие	Причина возникновения
	<p>объект.</p> <p>Разберём чуть подробнее:</p> <p>Событие Paint полезно, если вы используете графические методы объекта (Line, PSet...) в коде. С помощью этого события вы можете убедиться в том, что данные перерисовались, когда это необходимо.</p> <p>Событие Paint вызывается при запуске метода Refresh.</p> <p>Если AutoRedraw установлено в True, то перерисовка происходит автоматически, таким образом, это событие теряет свою необходимость.</p> <p>Если свойство ClipControls установлено в False, то графические методы в процедуре Paint воздействуют только на видимую часть формы; иначе, графические методы перерисовывают всю открытую часть формы (т.е. ту, которая не перекрыта такими элементами, как Image, Label, Line, и Shape).</p> <p>Используя метод Refresh в событии Resize вызывает перерисовку всего объекта каждый раз, когда происходит изменение его размеров (Resize).</p> <p>Замечание: Используя событие Paint для определённых задач, может произойти каскадирование событий (Т.е. просто напросто произойдёт рекурсия, когда Paint будет вызывать сам себя, и произойдёт переполнение стека).</p> <p>Чтобы этого избежать, нужно придерживаться следующих правил:</p> <ul style="list-style-type: none"> · Избегать вызова события Paint при сдвиге или изменении размеров объекта. · Внутри события Paint изменять свойства, которые могут

Событие	Причина возникновения
	<p>вызвать событие Paint. Такие, например, как BackColor.</p> <ul style="list-style-type: none"> · Включать метод Refresh метод внутрь Paint.
PathChange	<p>Происходит при смене пути, установкой свойства FileName или Path в коде.</p> <p>Замечание: Вы можете использовать это события, чтобы сообщить FileListBox'у о том, что путь у DirListBox изменился.</p>
PatternChange	<p>Происходит при изменении маски (такой, как "*.*)") установкой свойств FileName или Pattern в коде.</p>
QueryUnload	<p>Это событие имеет только форма.</p> <p>Происходит перед закрытием формы или приложения. Если закрывается MDI форма, то это событие происходит сначала для формы контейнера, и лишь потом для дочерних форм.</p> <p>Это событие происходит перед событием Unload.</p> <p>Обработчику данного события передаётся два параметра:</p> <p>cancel - integer - если в обработчике события присвоить переменной cancel значение True, то выгрузка формы будет отменена. Если оставить там False, то форма благополучно выгрузится.</p> <p>unloadmode - константа. Содержит значение - почему происходит выгрузка формы. Может принимать:</p> <p>vbFormControlMenu = 0 - пользователь выбрал команду Close (Закрыть) в меню приложения (слева вверху в заголовке формы).</p> <p>vbFormCode = 1 - произошёл вызов оператора Unload в коде программы.</p> <p>vbAppWindows = 2 - Windows завершает работу.</p> <p>vbAppTaskManager = 3 - закрытие приложения происходит</p>

Событие	Причина возникновения
	<p>через Ctrl+Alt+Del.</p> <p>vbFormMDIForm = 4 - дочерняя MDI форма закрывается, т.к. закрывается главная.</p> <p>Замечания:</p> <p>Вообще, это событие обычно применяется для проверки завершенности некоторых действий. Или также, например, здесь можно спросить пользователя, действительно ли он хочет выйти из приложения, или это произошло случайно. Следать это можно примерно так:</p> <pre>Private Sub Form_QueryUnload _ (Cancel As Integer, _ UnloadMode As Integer) Dim rez As VbMsgBoxResult rez = MsgBox("Вы действительно" & _ " хотите выйти?", _ vbQuestion + vbYesNo) If rez = vbNo Then Cancel = 1 End Sub</pre>
Reposition	<p>Происходит, когда запись становится текущей записью.</p> <p>Замечания:</p> <p>Когда Data контрол загружен, первая запись в объекте RecordSet становится текущей, и вызывается это событие.</p> <p>Когда бы пользователь не кликнул любую кнопку на контроле Data, передвигаясь с записи на запись, или вы используете одно из методов Move в коде (такие, как MoveFirst, MoveNext, FindFirst..), или любое другое свойство, которое может изменить текущую запись - происходит событие Reposition.</p>

Событие	Причина возникновения
	Событие Validate происходит перед этим событием.
Resize	<p>Происходит, когда объект первый раз появляется или когда меняется состояние окна (например, при свёртывании и развёртывании окна). А также при смене размеров окна.</p> <p>Это событие удобно использовать, если вы хотите сделать "растягивающийся" интерфейс. Т.е. когда все элементы на форме меняют свои размеры, в зависимости от текущих размеров формы. Код для изменения размеров этих элементов как раз можно поместить в это событие.</p>
Scroll	<p>Происходит тогда, когда сдвигается полоса прокрутки (ScrollBar).</p> <p>Замечания:</p> <p>Для DBGrid это событие происходит, когда двигается вертикальная или горизонтальная полоса прокрутки.</p> <p>Для ComboBox это событие происходит при сдвиге скролбара в выпадающем списке.</p> <p>Вы можете использовать это события для синхронизации изменения положения полосы прокрутки и другими объектами, с которыми эта полоска связана. К примеру, в обработку этого события вставить код, который будет синхронизировать текущую позицию с другим элементом.</p> <p>Поместите на форму HScroll и вставьте такой код:</p> <pre>Private Sub HScroll1_Scroll() Form1.Caption = HScroll1.Value End Sub</pre> <p>Теперь в заголовок формы будет меняться в соответствии со сменой положения полосы прокрутки.</p> <p>Кстати, избегайте использования функции MsgBox в этом</p>

Событие	Причина возникновения
	событии!
Terminate	<p>Это событие имеет только форма.</p> <p>Происходит когда все ссылки на экземпляры форм, MDI форм, элементы управления, классы удалены из памяти (установкой переменной в Nothing).</p> <p>Это событие выполняется после события Unload, и выполняется самым последним в программе. Можете использовать его по своему усмотрению.</p>
Timer	<p>Это событие имеется только у элемента управления Timer.</p> <p>Оно происходит через определённый интервал времени, указанный в его свойстве Interval.</p> <p>Timer невидим для пользователя, и полезен для внутренних процессов программы.</p> <p>Его можно использовать, когда необходимо, чтобы некоторый код программы выполнялся через определённый промежуток времени.</p> <p>Например, с его помощью можно сделать часы. Для этого достаточно поместить на форму элемент Timer, установите его свойство Interval в 1000 (1 сек = 1000 мс), и вставить следующий код:</p> <pre>Private Sub Timer1_Timer() Form1.Caption = Time End Sub</pre> <p>Функция Time возвращает текущее системное время. Таким образом, каждую секунду будет выполнять событие Timer, и следовательно изменение заголовка формы.</p> <p>Примечание: количество таймеров на форме не ограничено.</p>

Событие	Причина возникновения
Unload	<p>Это событие имеет только форма.</p> <p>Происходит, когда форма выгружается из памяти (оператором Unload, например, или нажатием на крестик). В дальнейшем она может быть загружена оператором Load.</p> <p>При перезагрузке формы ВСЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ ИНИЦИАЛИЗИРУЮТСЯ ЗАНОВО, т.е. все значения, которые были в них - теряются.</p>
Updated	<p>Происходит когда меняются данные (при работе с БД).</p> <p>Имеет параметр Code - integer - описывает, как объект был изменён (описано в настройках). Может принимать следующие константы:</p> <p>vbOLEChanged = 0 - Данные объекта изменились</p> <p>vbOLESaved = 1 - Данные объекта были сохранены приложением</p> <p>vbOLEClosed = 2 - Файл, на который ссылается объект, был закрыт</p> <p>vbOLERenamed = 3 - Файл, на который ссылается объект, был переименован</p>
Validate	<p>Это событие нужно для проверки введённых данных. Например, в TextBox. Это событие выполняется только тогда, когда свойство объекта CausesValidation = True.</p> <p>Рассмотрим пример, как можно проконтролировать данные, введённые в текстовое поле. Поместите на форму TextBox и вставьте код:</p> <pre>Private Sub Text1_Validate _ (Cancel As Boolean) If Not (IsNumeric(Text1.Text)) Then Text1.ForeColor = vbRed</pre>

Событие	Причина возникновения
	<pre> MsgBox "Введите в TextBox числа" Cancel = True Else Text1.ForeColor = vbButtonText End If End Sub </pre> <p>Этот код не примет введённый в TextBox текст, пока тот не будет являться числом. Если установить параметр Cancel в True, то фокус вернётся обратно TextBox.</p> <p>Это событие происходит при потере фокуса у объекта.</p>

10 Приложение: Таблица кодов ASCII

Dec	Hex	Символ
000	00	спец. NOP
001	01	спец. SOH
002	02	спец. STX
003	03	спец. ETX
004	04	спец. EOT
005	05	спец. ENQ
006	06	спец. ACK
007	07	спец. BEL
008	08	спец. BS
009	09	спец. Табуляция
010	0A	спец. LF (Возвр. каретки)
011	0B	спец. VT
012	0C	спец. FF
013	0D	спец. CR (Новая строка)
014	0E	спец. SO
015	0F	спец. SI
016	10	спец. DLE
017	11	спец. DC1
018	12	спец. DC2
019	13	спец. DC3
020	14	спец. DC4
021	15	спец. NAK

Dec	Hex	Символ
128	80	Ђ
129	81	Ѓ
130	82	Ѕ
131	83	Ї
132	84	„
133	85	...
134	86	†
135	87	‡
136	88	€
137	89	‰
138	8A	Љ
139	8B	Њ
140	8C	Њ
141	8D	Ќ
142	8E	ћ
143	8F	џ
144	90	ђ
145	91	‘
146	92	’
147	93	“
148	94	”
149	95	•
150	96	—

022	16	спец. SYN
023	17	спец. ETB
024	18	спец. CAN
025	19	спец. EM
026	1A	спец. SUB
027	1B	спец. ESC
028	1C	спец. FS
029	1D	спец. GS
030	1E	спец. RS
031	1F	спец. US
032	20	сцеп. SP (Пробел)
033	21	!
034	22	"
035	23	#
036	24	\$
037	25	%
038	26	&
039	27	'
040	28	(
041	29)
042	2A	*
043	2B	+
044	2C	,
045	2D	-
046	2E	.

151	97	—
152	98	□
153	99	™
154	9A	љ
155	9B	›
156	9C	њ
157	9D	ќ
158	9E	ћ
159	9F	џ
160	A0	
161	A1	Ў
162	A2	ў
163	A3	J
164	A4	Ѡ
165	A5	Ѓ
166	A6	Ѕ
167	A7	§
168	A8	Ё
169	A9	©
170	AA	€
171	AB	«
172	AC	¬
173	AD	
174	AE	®
175	AF	Ї

047	2F	/
048	30	0
049	31	1
050	32	2
051	33	3
052	34	4
053	35	5
054	36	6
055	37	7
056	38	8
057	39	9
058	3A	:
059	3B	;
060	3C	<
061	3D	=
062	3E	>
063	3F	?
064	40	@
065	41	A
066	42	B
067	43	C
068	44	D
069	45	E
070	46	F
071	47	G

176	B0	°
177	B1	±
178	B2	I
179	B3	i
180	B4	r
181	B5	μ
182	B6	¶
183	B7	·
184	B8	ë
185	B9	№
186	BA	€
187	BB	»
188	BC	j
189	BD	S
190	BE	s
191	BF	ï
192	C0	A
193	C1	Б
194	C2	B
195	C3	Г
196	C4	Д
197	C5	Е
198	C6	Ж
199	C7	З
200	C8	И

072	48	H
073	49	I
074	4A	J
075	4B	K
076	4C	L
077	4D	M
078	4E	N
079	4F	O
080	50	P
081	51	Q
082	52	R
083	53	S
084	54	T
085	55	U
086	56	V
087	57	W
088	58	X
089	59	Y
090	5A	Z
091	5B	[
092	5C	\
093	5D]
094	5E	^
095	5F	_
096	60	`

201	C9	Й
202	CA	К
203	CB	Л
204	CC	М
205	CD	Н
206	CE	О
207	CF	П
208	D0	Р
209	D1	С
210	D2	Т
211	D3	У
212	D4	Ф
213	D5	Х
214	D6	Ц
215	D7	Ч
216	D8	Ш
217	D9	Щ
218	DA	Ъ
219	DB	Ы
220	DC	Ь
221	DD	Э
222	DE	Ю
223	DF	Я
224	E0	а
225	E1	б

097	61	a
098	62	b
099	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y

226	E2	в
227	E3	г
228	E4	д
229	E5	е
230	E6	ж
231	E7	з
232	E8	и
233	E9	й
234	EA	к
235	EB	л
236	EC	м
237	ED	н
238	EE	о
239	EF	п
240	F0	р
241	F1	с
242	F2	т
243	F3	у
244	F4	ф
245	F5	х
246	F6	ц
247	F7	ч
248	F8	ш
249	F9	щ
250	FA	ъ

122	7A	z
123	7B	{
124	7C	
125	7D	}
126	7E	~
127	7F	□

251	FB	Ы
252	FC	Ь
253	FD	Э
254	FE	Ю
255	FF	Я

11 Приложение: Виртуальные коды клавиш

Константа WinApi	Hex значение	Клавиатурный или мышинный эквивалент
VK_LBUTTON	01	Left mouse button
VK_RBUTTON	02	Right mouse button
VK_CANCEL	03	Control-break processing
VK_MBUTTON	04	Middle mouse button (three-button mouse)
-	05-07	Undefined
VK_BACK	08	BACKSPACE key
VK_TAB	09	TAB key
-	0A-0B	Undefined
VK_CLEAR	0C	CLEAR key
VK_RETURN	0D	ENTER key
-	0E-0F	Undefined
VK_SHIFT	10	SHIFT key
VK_CONTROL	11	CTRL key
VK_MENU	12	ALT key
VK_PAUSE	13	PAUSE key
VK_CAPITAL	14	CAPS LOCK key
-	15-19	Reserved for Kanji systems
-	1A	Undefined
VK_ESCAPE	1B	ESC key
-	1C-1F	Reserved for Kanji systems
VK_SPACE	20	SPACEBAR
VK_PRIOR	21	PAGE UP key
VK_NEXT	22	PAGE DOWN key

Константа WinApi	Hex значение	Клавиатурный или мышиный эквивалент
VK_END	23	END key
VK_HOME	24	HOME key
VK_LEFT	25	LEFT ARROW key
VK_UP	26	UP ARROW key
VK_RIGHT	27	RIGHT ARROW key
VK_DOWN	28	DOWN ARROW key
VK_SELECT	29	SELECT key
-	2A	Original equipment manufacturer (OEM) specific
VK_EXECUTE	2B	EXECUTE key
VK_SNAPSHOT	2C	PRINT SCREEN key for Windows 3.0 and later
VK_INSERT	2D	INS key
VK_DELETE	2E	DEL key
VK_HELP	2F	HELP key
VK_0	30	0 key
VK_1	31	1 key
VK_2	32	2 key
VK_3	33	3 key
VK_4	34	4 key
VK_5	35	5 key
VK_6	36	6 key
VK_7	37	7 key
VK_8	38	8 key

Константа WinApi	Hex значение	Клавиатурный или мышинный эквивалент
VK_9	39	9 key
-	3A-40	Undefined
VK_A	41	A key
VK_B	42	B key
VK_C	43	C key
VK_D	44	D key
VK_E	45	E key
VK_F	46	F key
VK_G	47	G key
VK_H	48	H key
VK_I	49	I key
VK_J	4A	J key
VK_K	4B	K key
VK_L	4C	L key
VK_M	4D	M key
VK_N	4E	N key
VK_O	4F	O key
VK_P	50	P key
VK_Q	51	Q key
VK_R	52	R key
VK_S	53	S key
VK_T	54	T key
VK_U	55	U key
VK_V	56	V key

Константа WinApi	Hex значение	Клавиатурный или мышинный эквивалент
VK_W	57	W key
VK_X	58	X key
VK_Y	59	Y key
VK_Z	5A	Z key
VK_LWIN	5B	Left Windows key (Microsoft Natural Keyboard)
VK_RWIN	5C	Right Windows key (Microsoft Natural Keyboard)
VK_APPS	5D	Applications key (Microsoft Natural Keyboard)
-	5E-5F	Undefined
VK_NUMPAD0	60	Numeric keypad 0 key
VK_NUMPAD1	61	Numeric keypad 1 key
VK_NUMPAD2	62	Numeric keypad 2 key
VK_NUMPAD3	63	Numeric keypad 3 key
VK_NUMPAD4	64	Numeric keypad 4 key
VK_NUMPAD5	65	Numeric keypad 5 key
VK_NUMPAD6	66	Numeric keypad 6 key
VK_NUMPAD7	67	Numeric keypad 7 key
VK_NUMPAD8	68	Numeric keypad 8 key
VK_NUMPAD9	69	Numeric keypad 9 key
VK_MULTIPLY	6A	Multiply key
VK_ADD	6B	Add key
VK_SEPARATOR	6C	Separator key

Константа WinApi	Hex значение	Клавиатурный или мышинный эквивалент
VK_SUBTRACT	6D	Subtract key
VK_DECIMAL	6E	Decimal key
VK_DIVIDE	6F	Divide key
VK_F1	70	F1 key
VK_F2	71	F2 key
VK_F3	72	F3 key
VK_F4	73	F4 key
VK_F5	74	F5 key
VK_F6	75	F6 key
VK_F7	76	F7 key
VK_F8	77	F8 key
VK_F9	78	F9 key
VK_F10	79	F10 key
VK_F11	7A	F11 key
VK_F12	7B	F12 key
VK_F13	7C	F13 key
VK_F14	7D	F14 key
VK_F15	7E	F15 key
VK_F16	7F	F16 key
VK_F17	80H	F17 key
VK_F18	81H	F18 key
VK_F19	82H	F19 key
VK_F20	83H	F20 key
VK_F21	84H	F21 key

Константа WinApi	Hex значение	Клавиатурный или мышинный эквивалент
VK_F22	85H	F22 key
VK_F23	86H	F23 key
VK_F24	87H	F24 key
-	88-8F	Unassigned
VK_NUMLOCK	90	NUM LOCK key
VK_SCROLL	91	SCROLL LOCK key
-	92-B9	Unassigned
-	BA-C0	OEM specific
-	C1-DA	Unassigned
-	DB-E4	OEM specific
-	E5	Unassigned
-	E6	OEM specific
-	E7-E8	Unassigned
-	E9-F5	OEM specific
VK_ATTN	F6	Attn key
VK_CRSEL	F7	CrSel key
VK_EXSEL	F8	ExSel key
VK_EREOF	F9	Erase EOF key
VK_PLAY	FA	Play key
VK_ZOOM	FB	Zoom key
VK_NONAME	FC	Reserved for future use.
VK_PA1	FD	PA1 key
VK_OEM_CLEAR	FE	Clear key

12 Приложение: Функции VB

Abs (number) - возвращает модуль числа.

Пример:

```
Dim MyNumber
MyNumber = Abs(50.3) ' возвратит 50.3.
MyNumber = Abs(-50.3) ' возвратит 50.3.
```

Array (arglist) - возвращает переменную типа variant содержащую массив.

Пример:

```
Dim MyWeek, MyDay
MyWeek = Array("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
' Возвращает значение с нижней границей массива равной 1
MyDay = MyWeek(2) ' Теперь MyDay содержит "Tue".
MyDay = MyWeek(4) ' Теперь MyDay содержит "Thu".
```

Asc (string) - возвращает код первого символа в строке параметре.

Пример:

```
Dim MyNumber
MyNumber = Asc("A") ' Возвратит 65.
MyNumber = Asc("a") ' Возвратит 97.
MyNumber = Asc("Apple") ' Возвратит 65.
```

Замечание:

AscB - используется со строкой, длина которой 1 байт

AscW - возвращает двухбайтовый код unicode. (В системах Unix)

Atn (number) - возвращает арктангенс числа.

Пример:

```
Dim pi
pi = 4 * Atn(1) ' Вычисляет значение числа pi.
```

Функции конвертирования типов:

CBool (expression) - возвращает boolean значение

CByte (expression) - возвращает значение типа Byte

CCur (expression) - возвращает значение типа Currency

CDate (expression) - возвращает значение типа Date

Cdbl (expression) - возвращает значение типа Double

CDec (expression) - возвращает значение типа Decimal

CInt (expression) - возвращает значение типа Integer

CLng (expression) - возвращает значение типа Long

CSng (expression) - возвращает значение типа Single

CVar (expression) - возвращает значение типа Variant

CStr (expression) - возвращает значение типа String

Примеры всех этих ф-ций:

CBool

```
Dim A, B, Check
A = 5: B = 5 ' Инициализация переменных.
Check = CBool(A = B) ' Check содержит True.
A = 0 ' Инициализация переменной.
Check = CBool(A) ' Check содержит False.
```

CByte

```
Dim MyDouble, MyByte
MyDouble = 125.5678 ' MyDouble имеет тип Double.
MyByte = CByte(MyDouble) ' MyByte содержит 126.
```

CCur

```
Dim MyDouble, MyCurr
MyDouble = 543.214588 ' MyDouble имеет тип Double.
MyCurr = CCur(MyDouble * 2) ' Конвертирует результат MyDouble * 2
' Double - (1086.429176).
' Currency - (1086.4292).
```

CDate

```
Dim MyDate, MyShortDate, MyTime, MyShortTime
MyDate = "February 12, 1969" ' Запись даты в переменную MyDate.
MyShortDate = CDate(MyDate) ' Конвертирование в тип Date.
MyTime = "4:35:47 PM" ' Запись времени в переменную MyTime.
MyShortTime = CDate(MyTime) ' Конвертирование в тип Date.
```

Cdbl

```
Dim MyCurr, MyDouble
MyCurr = CCur(234.456784) ' MyCurr имеет тип Currency.
```

```
MyDouble = CDb1(MyCurr * 8.2 * 0.01) 'Конвертирует результат в Double.
```

CInt

```
Dim MyDouble, MyInt
MyDouble = 2345.5678 ' MyDouble имеет тип Double.
MyInt = CInt(MyDouble) ' MyInt содержит 2346.
```

CLng

```
Dim MyVal1, MyVal2, MyLong1, MyLong2
MyVal1 = 25427.45: MyVal2 = 25427.55 'MyVal1, MyVal2 имеют тип Double.
MyLong1 = CLng(MyVal1) ' MyLong1 содержит 25427.
MyLong2 = CLng(MyVal2) ' MyLong2 содержит 25428.
```

CSng

```
Dim MyDouble1, MyDouble2, MySingle1, MySingle2
' MyDouble1, MyDouble2 имеют тип Double.
MyDouble1 = 75.3421115: MyDouble2 = 75.3421555
MySingle1 = CSng(MyDouble1) ' MySingle1 содержит 75.34211.
MySingle2 = CSng(MyDouble2) ' MySingle2 содержит 75.34216.
```

CStr

```
Dim MyDouble, MyString
MyDouble = 437.324 ' MyDouble имеет тип Double.
MyString = CStr(MyDouble) ' MyString содержит "437.324".
```

CVar

```
Dim MyInt, MyVar
MyInt = 4534 ' MyInt имеет тип Integer.
MyVar = CVar(MyInt & "000") 'MyVar содержит значение String "4534000".
```

Cos (number) - вычисляет косинус числа.

Пример:

```
Dim MyAngle, MySecant
MyAngle = 1.3 ' Устанавливаем угол в радианах.
MyCos = Cos(MyAngle) ' Вычисляем косинус.
MySecant = 1 / Cos(MyAngle)
```

Chr (charcode) - возвращает символ, соответствующий определённому коду.

Эта ф-ция является обратной Asc.

Пример:

```
Dim MyChar
MyChar = Chr(65) ' Возвратит А.
MyChar = Chr(97) ' Возвратит а.
```

```
MyChar = Chr(62) ' Возвратит >.
MyChar = Chr(37) ' Возвратит %.
```

Choose (index, choice-1[, choice-2, ... [, choice-n]]) - выбирает значение из списка аргументов.

Пример:

```
Function GetChoice(Ind As Integer)
GetChoice = Choose(Ind, "Speedy", "United", "Federal")
End Function
```

Замечание:

Можно ускорить код с использованием Choose:

Вы можете использовать Choose там, где можно заменить массив или построить таблицы результатов, на стадии компиляции (compile-time), вместо того, чтобы делать это на стадии выполнения (run time). Например, если Вам надо знать значения факториалов чисел от 1 до 10, попробуйте следующий пример (Choose производит выбор факториала из набора имеющихся значений вместо того, чтобы высчитывать факториал каждый раз заново):

```
Function Factorial (number As Integer) As Long
Factorial = Choose(number, 1, 2, 6, _
24, 120, 720, 5040, 40320, _
362880, 3628800)
End Function
```

Command - возвращает аргументы с командной строки (т.е. если запустить программу из командной строки, например: myprogram.exe arguments)

Пример:

Этот пример использует функцию Command для получения аргументов с командной строки. Функция GetCommandLine возвратит значение типа Variant содержащее их массив.

```
Function GetCommandLine(Optional MaxArgs)
Dim C, CmdLine, CmdLnLen, InArg, I, NumArgs
If IsMissing(MaxArgs) Then MaxArgs = 10
'Приводим массив в корректный размер
ReDim ArgArray(MaxArgs)
NumArgs = 0: InArg = False
'Получаем аргументы с командной строки.
CmdLine = Command()
CmdLnLen = Len(CmdLine)
```

```

'Перемещаемся по всем символам в командной строке
For I = 1 To CmdLnLen
C = Mid(CmdLine, I, 1)
'Tестируем на пробел или символ табуляции.
If (C <> " " And C <> vbTab) Then
'Если ни то и ни другое.
'Тест на текущий аргумент.
If Not InArg Then
'Новый аргумент начался.
'Tестируем на слишком большое кол-во аргументов.
If NumArgs = MaxArgs Then Exit For
NumArgs = NumArgs + 1
InArg = True
End If
'Соединение символа с текущим аргументом.
ArgArray(NumArgs) = ArgArray(NumArgs) & C
Else
'Ищем пробел или символ табуляции.
InArg = False
End If
Next I
ReDim Preserve ArgArray(NumArgs)
'Возвращаем массив как результат ф-ции.
GetCommandLine = ArgArray()
End Function

```

CreateObject (class) - возвращает ссылку на объект ActiveX класса class

Пример:

```

Dim xlApp As Object ' Объявляем переменную типа Object.
Set xlApp = CreateObject("excel.application")
' Должны изменить свойство Visible в True, если хотите,
' чтобы проект Excel был виден
xlApp.Visible = True
' Используем xlApp для доступа к другим объектам Microsoft Excel's.
xlApp.Quit ' используйте метод
' Quit для закрытия Excel
Set xlApp = Nothing ' Сбрасываем ссылку

```

CurDir [(drive)] - возвращает String, содержащий текущий путь

Пример:

```
' Предположим, что текущим путь на диске C - "C:\WINDOWS\SYSTEM".
' Предположим, что текущим путь на диске D - "D:\EXCEL".
' Предположим, что текущий диск - диск C.
Dim MyPath
MyPath = CurDir ' возвратит "C:\WINDOWS\SYSTEM".
MyPath = CurDir("C") ' возвратит "C:\WINDOWS\SYSTEM".
MyPath = CurDir("D") ' возвратит "D:\EXCEL".
```

CVErr (error number) - возвращает Variant субтипа Error, содержащий код ошибки, определённый пользователем.

Пример:

```
' Вызывает CalculateDouble с аргументом, приводящим к ошибке.
Sub Test()
Debug.Print CalculateDouble("345.45robert")
End Sub

' Определение ф-ции CalculateDouble.
Function CalculateDouble(Number)
If IsNumeric(Number) Then
CalculateDouble = Number * 2 ' Возвращает результат.
Else
CalculateDouble = CVErr(2001) ' Возвращает номер ошибки,
' определённый пользователем.
End If
End Function
```

Date - возвращает текущую дату.

Пример:

```
Dim MyDate
MyDate = Date ' MyDate содержит текущую системную дату.
```

DateAdd (interval, number, date) - Возвращает дату, в которой к date добавляет временной интервал.

Параметр interval может принимать следующие значения:

уууу - Год

q - Квартал

m - Месяц

y - День года

d - День

w - День недели

ww - Неделя

h - Час

n - Минута

s - Секунда

number - количество интервалов, которые вы хотите добавить.

Пример:

```
Dim FirstDate As Date
Dim IntervalType As String
Dim Number As Integer

IntervalType = "m"
'"m" устанавливает в качестве интервала - месяцы.
FirstDate = Date
Number = InputBox("Введите кол-во месяцев для добавки")
MsgBox = "Новая дата: " & _
DateAdd(IntervalType, Number, FirstDate)
```

DateDiff (interval, date1, date2[, firstdayofweek[, firstweekofyear]]) - возвращает разность между двумя датами. Параметр interval указывает в каком формате будет вычисляться разность (значения такие же, как и ф-ции DateAdd).

Пример:

```
Dim TheDate As Date
Dim Msg
TheDate = InputBox("Введите дату")
Msg = "Кол-во дней до сегодняшней даты: " & _
DateDiff("d", Now, TheDate)
MsgBox Msg
```

DatePart (interval, date[,firstdayofweek[, firstweekofyear]]) - Возвращает часть даты (например, месяц или день).

Пример:

```
Dim TheDate As Date
Dim Msg
TheDate = Date
Msg = "Номер месяца: " & DatePart("m", TheDate)
```

DateSerial (year, month, day) - Возвратит дату, указанную по частям.

Пример:

```
Dim MyDate
MyDate = DateSerial (2003, 2, 12)
' Возвратит дату February 12, 2003.
```

DateValue (string) - Возвращает дату, заданную в виде строки.

Пример:

```
Dim MyDate
MyDate = DateValue("February 12, 1969") ' Возвратит дату
```

Day (date) - Возвратит день из указанной даты.

Пример:

```
'Допустим на дворе 5 апреля 2001г.
Dim MyDate, MyDay
MyDate = Date
'MyDate содержит текущую системную дату(5 апреля 2001г.)
MyDay = Day(MyDate) ' MyDay содержит текущий день (т.е. 5).
```

Dir [(pathname[, attributes])] - Возвращает строку, содержащую имя файла или директории, которое совпадает с определённой маской, атрибутом или меткой диска.

Пример:

```
Dim MyFile, MyPath, MyName
MyFile = Dir("C:\WINDOWS\WIN.INI")
' Возвратит "WIN.INI" если файл существует.
MyFile = Dir("C:\WINDOWS\*.INI")
' Возвратит имя файла с расширением INI.
' Если файлов несколько, то возвратится имя первого файла.
' Снова вызываем ф-цию Dir, только без параметров,
' чтобы получить имя следующего файла
' с расширением INI в директории WINDOWS
MyFile = Dir
' Возвратит имя первого файла
' с расширением TXT и атрибутом "скрытый"
```



```

MyFile = Dir("*.TXT", vbHidden)

' Нижеприведённый кусок кода
' отобразит в окне Immediate список папок
' в корневой директории на диске C

MyPath = "C:\"
MyName = Dir(MyPath, vbDirectory)

' Возвратит первую директорию.
Do While MyName <> "" ' Начало цикла
' Игнорируем текущую директорию или заключительную
If MyName <> "." And MyName <> ".." Then
' Используем битовое сравнение, чтобы определить
' что MyName есть директория (а не файл).
If (GetAttr(MyPath & MyName) And vbDirectory) = vbDirectory Then
Debug.Print MyName ' Отобразим в окне Immediate MyName
End If
End If
MyName = Dir.
Loop

```

DoEvents () - Позволяет ОС реагировать на события, когда происходят длительные вычисления.

Пример:

```

Do
Loop

' Попробуйте подвигать форму (не реагирует)
Do
DoEvents
Loop

' А теперь?

```

Environ ({envstring | number}) - Возвращает строку, ассоциированную с переменной окружения ОС.

Пример:

```

Form1.Caption = Environ ("WINDIR")

' Возвратит путь к директории Windows
' В этом примере в Text1 отобразятся все
' переменные окружения (по строчкам)
Dim c As String

```

```
For a = 1 To 20
c = Environ(a)
If c <> "" Then
Text1.Text = Text1.Text & c & vbCrLf
Else
Exit For
End If
Next a
```

EOF (filenumber) - Возвращает True, если достигнут конец файла при чтении.

Пример:

```
Dim InputData
Open "MYFILE" For Input As #1 ' Открываем файл для чтения
Do While Not EOF(1) 'Проверяем не конец ли
Line Input #1, InputData ' Считываем строку
Debug.Print InputData ' Выводим её в Immediate окно
Loop
Close #1 ' Закрываем файл
```

Error [(errornumber)] - Возвращает строку, содержащую описание ошибки, определённой параметром errornumber.

Пример:

```
Dim ErrorNumber
For ErrorNumber = 61 To 64
' Выведет описание ошибок с 61 по 64
Debug.Print Error(ErrorNumber)
Next ErrorNumber
```

Exp (number) - Возвращает экспоненту числа number.

Пример:

```
Form1.Caption = Exp(1)
'Отобразит на Caption число e (т.е. e в степени 1).
```

FileAttr(filenumber, returntype) - Возвращает вид открытия файла ф-цией

Open. returntype должно равняться 1. Варианты возвращаемых значений (слева - тип, справа - значение):

Input 1

Output 2

Random 4

Append 8

Binary 32

Пример:

```
Dim FileNum, Mode
FileNum = 1
Open "TESTFILE" For Append As FileNum
Mode = FileAttr(FileNum, 1) ' Возвратит 8 (т.е. файл открыт
' методом Append).
Close FileNum ' Close file.
```

FileDateTime(pathname) - возвращает тип Date, даты последнего изменения файла.

Пример:

```
Dim MyStamp
' Предположим, что файл TESTFILE последний раз был изменён
' 12 февраля 1993 в 16:35:47
MyStamp = FileDateTime("TESTFILE") ' Возвратит "2/12/93 4:35:47 PM".
```

FileLen(pathname) - возвращает размер файла.

Пример:

```
Dim MySize
MySize = FileLen("TESTFILE") ' Возвратит длину файла в байтах
```

Fix(number) - возвратит целую часть числа.

Пример:

```
Dim MyNumber

MyNumber = Fix(99.2) ' возвратит 99.
MyNumber = Fix(-99.8) ' возвратит -99.
MyNumber = Fix(-99.2) ' возвратит -99.
```

Format(expression[, format[, firstdayofweek[, firstweekofyear]]]) - возвращает String отформатированной строки (как в C).

Пример:

```

Dim MyTime, MyDate, MyStr
MyTime = #17:04:23#
MyDate = #January 27, 1993#
' Возвратит текущее системное время в длинном формате.
MyStr = Format(Time, "Long Time")
' Возвратит текущую системную дату в длинном формате.
MyStr = Format(Date, "Long Date")
MyStr = Format(MyTime, "h:m:s") ' Возвратит "17:4:23".
MyStr = Format(MyTime, "hh:mm:ss AMPM") ' Возвратит "05:04:23 PM".
MyStr = Format(MyDate, "dddd, mmm d yyyy")
' Возвратит "Wednesday,
' Jan 27 1993".
' Если формат не задан, то возвратится просто строка.
MyStr = Format(23) ' Возвратит "23".
' Форматы определённые пользователем
MyStr = Format(5459.4, "##,##0.00") ' Возвратит "5,459.40".
MyStr = Format(334.9, "###0.00") ' Возвратит "334.90".
MyStr = Format(5, "0.00%") ' Возвратит "500.00%".
MyStr = Format("HELLO", "<") ' Возвратит "hello".
MyStr = Format("This is it", ">") ' Возвратит "THIS IS IT".

```

Если вам нужно подробное описание всех возможных форматов - обращайтесь к документации по VB.

FreeFile[(rangenumbers)] - возвратит следующий пустой файловый номер.

Пример:

```

Dim MyIndex, FileNumber
For MyIndex = 1 To 5 ' Начало цикла
FileNumber = FreeFile ' Получаем неиспользуемый номер
Open "TEST" & MyIndex For Output As #FileNumber
Write #FileNumber, "Это пример." ' Запишем текст в файл
Close #FileNumber ' Закроем файл
Next MyIndex

```

GetAllSettings(appname, section) - Возвратит список всех настроек в секции. (Которые были записаны ф-цией SaveSetting).

Пример:

```

Dim MySettings As Variant, intSettings As Integer
' Запишем кое-какие настройки
SaveSetting "MyApp", "Startup", "Left", 50

```

```
' Возвратит строку:
MySettings = GetAllSettings("MyApp", "Startup")
For intSettings = LBound(MySettings, 1) To UBound(MySettings, 1)
Debug.Print MySettings(intSettings, 0), MySettings(intSettings, 1)
Next intSettings
DeleteSetting "MyApp", "Startup" ' Удалим записанные настройки
```

GetAttr(pathname) - Возвратит атрибуты файла или папки.

Варианты возвращаемых значений:

vbNormal - 0 - Normal

vbReadOnly - 1 - Read-only

vbHidden - 2 - Hidden

vbSystem - 4 - System

vbDirectory - 16 - Directory or folder

vbArchive - 32 - File has changed since last backup

Пример:

```
Dim MyAttr
' Предположим файл TESTFILE имеет атрибут "скрытый"
MyAttr = GetAttr("TESTFILE") ' Возвратит 2
' Возвратит не ноль, если атрибут "скрытый" установлен
Debug.Print MyAttr And vbHidden
' Возвратит не ноль, если установлены "скрытый" и "только для чтения"
Debug.Print MyAttr And (vbHidden + vbReadOnly)
' Предположим что MYDIR папка.
MyAttr = GetAttr("MYDIR") ' Возвратит 16.
```

object.GetAutoServerSettings([progid], [clsid]) - с помощью этой ф-ции можно получить ProgID и CLSID, т.е. GUID компонента ActiveX.

Пример:

В этом примере мы получаем информацию о регистрации объекта Hello.

```
Dim oRegClass As New RegClass
Dim vRC As Variant
vRC = oRegClass.GetAutoServerSettings ("HelloProj.HelloClass")
If Not (IsEmpty(vRC)) Then
If vRC(1) Then
MsgBox "Hello удалённо зарегистрирован на сервере - " & vRC(1)
```

```
Else
MsgBox "Hello зарегистрирован локально."
End If
End if
```

GetObject([pathname] [, class]) - возвращает ссылку на объект ActiveX ассоциированного с определённым файлом.

Пример:

В это примере переменная MyXL получает ссылку на объект ActiveX Excel.

```
Dim MyXL As Object
Set MyXL = GetObject("C:\Мои документы\EXCEL\Русское лото.xls")
MyXL.Application.Visible = True
MyXL.Parent.Windows(1).Visible = True
'здесь производятся необходимые манипуляции с файлом
MyXL.Application.Quit ' Закрытие приложения Excel
Set MyXL = Nothing ' Освобождение ссылки
```

GetSetting(appname, section, key[, default]) - Возвращает настройку из реестра, записанную туда ф-цией SaveSetting. Настройки записываются по адресу:

HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings

Пример:

```
Dim znachenie As Long
znachenie = 1212 ' Это число взято от балды, для примера
SaveSetting "MyApp", "Sekcia", "Kluch", znachenie 'сохраняем настройки
Form1.Caption = GetSetting("MyApp", "Sekcia", "Kluch", 2222)
' получаем 1212 или если настроек нет, то
' получаем значение по умолчанию 2222.
```

FileAttr(filename, returntype) - Возвращает вид открытия файла ф-цией Open. returntype должно равняться 1. Варианты возвращаемых значений (слева - тип, справа - значение):

Input 1

Output 2

Random 4

Append 8

Binary 32

Пример:

```
Dim FileNum, Mode
FileNum = 1
Open "TESTFILE" For Append As FileNum
Mode = FileAttr(FileNum, 1) ' Возвратит 8 (т.е. файл открыт
' методом Append).
Close FileNum ' Close file.
```

FileDateTime(pathname) - возвращает тип Date, даты последнего изменения файла.

Пример:

```
Dim MyStamp
' Предположим, что файл TESTFILE последний раз был изменён
' 12 февраля 1993 в 16:35:47
MyStamp = FileDateTime("TESTFILE") ' Возвратит "2/12/93 4:35:47 PM".
```

FileLen(pathname) - возвращает размер файла.

Пример:

```
Dim MySize
MySize = FileLen("TESTFILE") ' Возвратит длину файла в байтах
```

Fix(number) - возвратит целую часть числа.

Пример:

```
Dim MyNumber

MyNumber = Fix(99.2) ' возвратит 99.
MyNumber = Fix(-99.8) ' возвратит -99.
MyNumber = Fix(-99.2) ' возвратит -99.
```

Format(expression[, format[, firstdayofweek[, firstweekofyear]]]) - возвращает String отформатированной строки (как в C).

Пример:

```
Dim MyTime, MyDate, MyStr
MyTime = #17:04:23#
MyDate = #January 27, 1993#
```

```

' Возвратит текущее системное время в длинном формате.
MyStr = Format(Time, "Long Time")
' Возвратит текущую системную дату в длинном формате.
MyStr = Format(Date, "Long Date")
MyStr = Format(MyTime, "h:m:s") ' Возвратит "17:4:23".
MyStr = Format(MyTime, "hh:mm:ss AMPM") ' Возвратит "05:04:23 PM".
MyStr = Format(MyDate, "dddd, mmm d yyyy")
' Возвратит "Wednesday,
' Jan 27 1993".
' Если формат не задан, то возвратится просто строка.
MyStr = Format(23) ' Возвратит "23".
' Форматы определённые пользователем
MyStr = Format(5459.4, "##,##0.00") ' Возвратит "5,459.40".
MyStr = Format(334.9, "##0.00") ' Возвратит "334.90".
MyStr = Format(5, "0.00%") ' Возвратит "500.00%".
MyStr = Format("HELLO", "<") ' Возвратит "hello".
MyStr = Format("This is it", ">") ' Возвратит "THIS IS IT".

```

Если вам нужно подробное описание всех возможных форматов - обращайтесь к документации по VB.

FreeFile[(rangenumbers)] - возвратит следующий пустой файловый номер.

Пример:

```

Dim MyIndex, FileNumber
For MyIndex = 1 To 5 ' Начало цикла
FileNumber = FreeFile ' Получаем неиспользуемый номер
Open "TEST" & MyIndex For Output As #FileNumber
Write #FileNumber, "Это пример." ' Запишем текст в файл
Close #FileNumber ' Закроем файл
Next MyIndex

```

GetAllSettings(appname, section) - Возвратит список всех настроек в секции. (Которые были записаны ф-цией SaveSetting).

Пример:

```

Dim MySettings As Variant, intSettings As Integer
' Запишем кое-какие настройки
SaveSetting "MyApp", "Startup", "Left", 50
' Возвратит строку:
MySettings = GetAllSettings("MyApp", "Startup")
For intSettings = LBound(MySettings, 1) To UBound(MySettings, 1)

```



```
Debug.Print MySettings(intSettings, 0), MySettings(intSettings, 1)
Next intSettings
DeleteSetting "MyApp", "Startup" ' Удалим записанные настройки
```

GetAttr(pathname) - Возвратит атрибуты файла или папки.

Варианты возвращаемых значений:

vbNormal - 0 - Normal

vbReadOnly - 1 - Read-only

vbHidden - 2 - Hidden

vbSystem - 4 - System

vbDirectory - 16 - Directory or folder

vbArchive - 32 - File has changed since last backup

Пример:

```
Dim MyAttr
' Предположим файл TESTFILE имеет атрибут "скрытый"
MyAttr = GetAttr("TESTFILE") ' Возвратит 2
' Возвратит не ноль, если атрибут "скрытый" установлен
Debug.Print MyAttr And vbHidden
' Возвратит не ноль, если установлены "скрытый" и "только для чтения"
Debug.Print MyAttr And (vbHidden + vbReadOnly)
' Предположим что MYDIR папка.
MyAttr = GetAttr("MYDIR") ' Возвратит 16.
```

object.GetAutoServerSettings([progid], [clsid]) - с помощью этой ф-ции можно получить ProgID и CLSID, т.е. GUID компонента ActiveX.

Пример:

В этом примере мы получаем информацию о регистрации объекта Hello.

```
Dim oRegClass As New RegClass
Dim vRC As Variant
vRC = oRegClass.GetAutoServerSettings ("HelloProj.HelloClass")
If Not (IsEmpty(vRC)) Then
If vRC(1) Then
MsgBox "Hello удалённо зарегистрирован на сервере - " & vRC(1)
Else
MsgBox "Hello зарегистрирован локально."
End If
```

```
End if
```

GetObject([pathname] [, class]) - возвращает ссылку на объект ActiveX ассоциированного с определённым файлом.

Пример:

В этом примере переменная MyXL получает ссылку на объект ActiveX Excel.

```
Dim MyXL As Object
Set MyXL = GetObject("C:\Мои документы\EXCEL\Русское лото.xls")
MyXL.Application.Visible = True
MyXL.Parent.Windows(1).Visible = True
'здесь производятся необходимые манипуляции с файлом
MyXL.Application.Quit ' Заккрытие приложения Excel
Set MyXL = Nothing ' Освобождение ссылки
```

GetSetting(appname, section, key[, default]) - Возвращает настройку из реестра, записанную туда ф-цией SaveSetting. Настройки записываются по адресу:

HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings

Пример:

```
Dim znachenie As Long
znachenie = 1212 ' Это число взято от балды, для примера
SaveSetting "MyApp", "Sekcia", "Kluch", znachenie 'сохраняем настройки
Form1.Caption = GetSetting("MyApp", "Sekcia", "Kluch", 2222)
' получаем 1212 или если настроек нет, то
' получаем значение по умолчанию 2222.
```

Hex (number) - возвращает строку, содержащую шестнадцатичное значение числа параметра.

Пример:

```
Dim MyHex
MyHex = Hex(5) ' Возвратит 5.
MyHex = Hex(10) ' Возвратит A.
MyHex = Hex(459) ' Возвратит 1CB.
```

Hour (time) - возвращает количество часов из параметра типа Time.

Пример:

```
Dim MyTime, MyHour
```

```
MyTime = #4:35:17 PM# ' Присваиваем время  
MyHour = Hour(MyTime) ' MyHour содержит 16.
```

IIf (expr, truepart, falsepart) - возвращает одну из 2-х строк, которая совпадёт с выражением условия. Практически аналог оператора ? в С и Java.

Пример:

В этом примере ф-ция CheckIt возвратит строку "Large", если TestMe > 1000, в противном случае возвратит "Small".

```
Function CheckIt (TestMe As Integer)  
CheckIt = IIf(TestMe > 1000, "Large", "Small")  
End Function
```

Input (number, [#]filenumber) - используется для работы с файлами.

Считывает одну компоненту файла.

Пример:

После запуска такой программы в окошке Immediate вы увидите символы, содержащиеся в файле TESTFILE. Ф-ция EOF возвращает True если достигнут конец файла при чтении.

```
Dim MyChar  
Open "TESTFILE" For Input As #1  
Do While Not EOF(1) 'Запускаем цикл до тех пор,  
' пока не будет достигнут конец файла...  
MyChar = Input(1, #1) 'Получаем один символ  
Debug.Print MyChar ' Печатаем в окно Immediate  
Loop  
Close #1 'Закрываем файл
```

InputBox (prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context]) -

выводит окно с запросом на ввод значения. Параметры такие же, как и у функции MsgBox.

Пример:

Выводим окно с заголовком "ВНИМАНИЕ", запросом "Введите пароль", в окошке будет выделенный текст "Значение_по_умолчанию". Координаты появления окна - 100,100px.

```
Form1.Caption = InputBox("Введите пароль", _  
"ВНИМАНИЕ", "Значение_по_умолчанию", 100, 100)
```

InStr ([start,]string1, string2[, compare]) - возвращает номер позиции первого вхождения строки string2 в строку string1.

Пример:

```
Dim SearchString, SearchChar, MyPos  
SearchString = "XpXpXpXpXpXp" ' Строка, в которой будем искать  
SearchChar = "p" ' Строка для поиска  
' Текстовое сравнение начиная с 4-ой позиции. Возвратит 6.  
MyPos = Instr(4, SearchString, SearchChar, 1)  
' Бинарное сравнение начиная с 1-ой позиции. Возвратит 9.  
MyPos = Instr(1, SearchString, SearchChar, 0)  
' Бинарное по умолчанию. Последний параметр опущен. Возвратит 9.  
MyPos = Instr(SearchString, SearchChar)  
' Возвратит 0, т.е. строка не найдена.  
MyPos = Instr(1, SearchString, "W")
```

Int (number) - Если number < 0, то округляет его до ближайшего целого.
Иначе выделяет целую часть number.

Пример:

```
Dim MyNumber  
MyNumber = Int(99.8) ' Возвратит 99.  
MyNumber = Int(-99.8) ' Возвратит -100.  
MyNumber = Int(-99.2) ' Возвратит -100.
```

Следующие функции используются для проверки принадлежность значения к определённому типу данных.

IsArray(varname) - Возвратит True, если varname (имя переменной) является массивом. Иначе False. Переменная может быть и типом Variant. Пример, я думаю, здесь не нужен.

IsDate(expression) - если expression - дата, то возвратит True, иначе False.

IsEmpty(expression) - если переменная пуста (т.е. либо ей присвоено значение Empty, либо значение ещё не задано после объявления переменной) то возвратит True, иначе False.

IsError(expression) - Возвратит True, если переменная содержит ошибочные данные.

Пример:

```
Dim ReturnVal, MyCheck
ReturnVal = UserFunction()
MyCheck = IsError(ReturnVal) ' возвратит True.
```

IsMissing(argname) - Возвратит True, если аргументы не заданы.

Пример:

```
Dim ReturnValue
'В следующих строках вызывается процедура, определённая пользователем
ReturnValue = ReturnTwice() ' Возвратит Null.
ReturnValue = ReturnTwice(2) ' Возвратит 4.
' Определение функции, вызываемой выше
Function ReturnTwice(Optional A)
If IsMissing(A) Then
' Если аргументы пропущены, возвратит Null.
ReturnTwice = Null
Else
' Если аргумент задан, то возвратит квадрат этого аргумента
ReturnTwice = A * 2
End If
End Function
```

IsNull(expression) - немного отличается от IsEmpty. А именно тем, что возвратит False, если переменной ещё не присвоено значение после определения. Пример:

```
Dim MyVar, MyCheck
MyCheck = IsNull(MyVar) ' Возвратит False.
MyVar = ""
MyCheck = IsNull(MyVar) ' Возвратит False.
MyVar = Null
MyCheck = IsNull(MyVar) ' Возвратит True.
```

IsNumeric(expression) - Возвращает True, если expression является числовым значением.

Пример:

```
Dim MyVar, MyCheck
MyVar = "53" ' Задаём значение
MyCheck = IsNumeric(MyVar) ' Возвратит True.
MyVar = "459.95" ' Задаём значение
```

```
MyCheck = IsNumeric(MyVar) ' Возвратит True.  
MyVar = "45 Help" ' Задаём значение  
MyCheck = IsNumeric(MyVar) ' Возвратит False.
```

IsObject(identifier) - Возвращает True, если переменная объектного типа.

Пример:

```
Dim MyInt As Integer, YourObject, MyCheck  
Dim MyObject As Object  
Set YourObject = MyObject ' Задаём ссылку на объект  
MyCheck = IsObject(YourObject) ' Возвратит True.  
MyCheck = IsObject(MyInt) ' Возвратит False.
```

LBound(arrayname[, dimension]) - возвращает минимально возможную границу размерности массива в заданном измерении.

Пример:

```
Dim Lower  
Dim MyArray(1 To 10, 5 To 15, 10 To 20) 'Объявляем массивы  
Dim AnyArray(10)  
  
Lower = Lbound(MyArray, 1) ' Возвратит 1.  
Lower = Lbound(MyArray, 3) ' Возвратит 10.  
Lower = Lbound(AnyArray) ' Возвратит 0 или 1, в зависимости от  
' установленной настройки Option Base.
```

LCase(string) - переводит строку в нижний регистр.

Пример:

```
Dim UpperCase, LowerCase  
Uppercase = "Hello World 1234" ' Строка для конвертирования  
Lowercase = Lcase(Uppercase) ' Возвратит строку "hello world 1234".
```

Left(string, length) - Возвращает часть строки, начиная с первого символа до указанного номера.

Пример:

```
Dim AnyString, MyStr  
AnyString = "Hello World" ' Определяем строку  
MyStr = Left(AnyString, 1) ' Возвратит "H".  
MyStr = Left(AnyString, 7) ' Возвратит "Hello W".  
MyStr = Left(AnyString, 20) ' Возвратит "Hello World".
```

Len(string | varname) - Возвращает длину строки string или количество байт, занимаемых переменной varname.

Пример:

```
Type CustomerRecord ' Определяем свою запись
ID As Integer ' Кладём это определение в модуль
Name As String * 10
Address As String * 30
End Type

Dim Customer As CustomerRecord
Dim MyInt As Integer, MyCur As Currency
Dim MyString, MyLen

MyString = "Hello World" ' Инициализация переменной
MyLen = Len(MyInt) ' Возвратит 2.(кол-во байт, занимаемых переменной)
MyLen = Len(Customer) ' Возвратит 42.
MyLen = Len(MyString) ' Возвратит 11. (длину строки)
MyLen = Len(MyCur) ' Возвратит 8.
```

LoadPicture([stringexpression]) - Загружает картинку в переменную типа Picture или элементы PictureBox и Image.

Пример:

```
Private Sub Form_Click ()
Dim Msg As String
On Error Resume Next ' Устанавливаем обработку ошибки
Height = 3990
Width = 4890 ' Устанавливаем высоту и ширину в твипах
Set Picture = LoadPicture("PAPER.BMP") ' Загрузка картинки
If Err Then
Msg = "Не могу найти .BMP файл."
MsgBox Msg ' Отобразим ошибку
Exit Sub ' Выходим, если ошибка имела место
End If

Msg = "Нажмите ОК для очистки формы от картинки."
MsgBox Msg
Set Picture = LoadPicture() ' Очищаем форму от картинки.
End Sub
```

Loc(filename) - возвращает номер текущей позиции в файле, открытом binary методом.

Пример:

```
Dim MyLocation, MyLine
Open "TESTFILE" For Binary As #1 ' Открываем существующий файл
Do While MyLocation < LOF(1) ' Зацикливаемся пока не конец файла
MyLine = MyLine & Input(1, #1) ' Читаем символ в переменную
MyLocation = Loc(1) ' Получаем текущую позицию в файле
' и выводим её в окно Immediate
Debug.Print MyLine; Tab; MyLocation
Loop
Close #1 ' Закрываем файл
```

LOF(filename) - Возвратит размер файла в байтах, открытого оператором Open.

Пример:

```
Dim FileLength
Open "TESTFILE" For Input As #1 ' Открываем файл
FileLength = LOF(1) ' Получаем длину файла
Close #1 ' Закрываем файл
```

Log(number) - вычисляет натуральный логарифм числа number. (Возвращает тип Double).

Пример:

```
Dim MyAngle, MyLog
' Устанавливаем угол в радианах
MyAngle = 1.3
' Вычисляем обратный гиперболический синус
MyLog = Log(MyAngle + Sqr(MyAngle * MyAngle + 1))
```

Ф-ция Log вычисляет натуральный логарифм (т.е. по основанию e). Для того, чтобы получить логарифм по основанию n нужно произвести следующее вычисление:

$$\text{Logn}(x) = \text{Log}(x) / \text{Log}(n)$$

LTrim(string) - Возвратит строку без лидирующих пробелов.

Пример:

```
Dim MyString, TrimString
MyString = " <Trim> " ' Устанавливаем строку,
```



```
' у которой слева и справа по одному пробелу
TrimString = LTrim(MyString) ' Получили строку "<-Trim-> "
' (т.е. без пробела слева)
```

Mid(string, start[, length]) - Возвращает строку, извлечённую из строки string, начиная с символа в позиции start, и содержащую length символов.

Пример:

```
Dim MyString, FirstWord, LastWord, MidWords
MyString = "Mid Function Demo" ' Создаём текстовую строку
FirstWord = Mid(MyString, 1, 3) ' Возвратит строку "Mid".
LastWord = Mid(MyString, 14, 4) ' Возвратит строку "Demo".
MidWords = Mid(MyString, 5) ' Возвратит строку "Function Demo".
```

Minute(time) - Возвратит число минут (от 0 до 59) содержащимся в параметре Time.

Пример:

```
Dim MyTime, MyMinute
MyTime = #4:35:17 PM# ' Присваиваем время
MyMinute = Minute(MyTime) ' MyMinute содержит 35.
```

Month(date) - то же, что и выше, только возвратит число дней в параметре date.

Пример:

```
Dim MyDate, MyMonth
MyDate = #February 12, 1969# ' Присваиваем дату
MyMonth = Month(MyDate) ' MyMonth содержит 2.
```

MsgBox(prompt[, buttons] [, title] [, helpfile, context]) - Выводит на экран окно сообщения (Message Box) которое будет ждать клика на одной из кнопок. Возвращает число Integer, по которому можно определить какую кнопку нажал пользователь.

Пример:

Этот пример выводит на экран критическую ошибку с кнопками Yes и No.

```
Dim Msg, Style, Title, Response, MyString
Msg = "Do you want to continue ?" ' Устанавливаем текст сообщения
```

```

Style = vbYesNo + vbCritical + vbDefaultButton2 ' Устанавливаем стиль
Title = "MsgBox Demonstration" ' Устанавливаем заголовок (Caption).

Response = MsgBox(Msg, Style, Title)
If Response = vbYes Then
MyString = "Yes"
' Выполняем какие-то действия, если пользователь выбрал Yes
Else
MyString = "No"
' Выполняем какие-то действия, если пользователь выбрал No
End If

```

Now - Возвращает Variant значение, содержащее текущие системные дату и время.

Пример:

```

Dim Today
Today = Now ' Today содержит текущие дату и время.

```

Oct(number) - Возвращает восьмиричное значение числа number.

Пример:

```

Dim MyOct
MyOct = Oct(4) ' Возвратит 4.
MyOct = Oct(8) ' Возвратит 10.
MyOct = Oct(459) ' Возвратит 713.

```

QBColor(color) - Возвращает значение Long, содержащее цвет, заданный номером от 0 до 15. Ниже приведена таблица цветов:

Second(time) - Возвратит количество секунд, содержащихся в параметре time.

Пример:

```

Dim MyTime, MySecond
MyTime = #4:35:17 PM# ' Присваиваем время.
MySecond = Second(MyTime) ' MySecond содержит 17.

```

Seek(filenum) - Возвращает Long, содержащее текущую read/write позицию в файле, открытом оператором Open.

Пример:

```

' Определение типа должно находиться в стандартном модуле

```

```

Type Record ' Тип, определённый пользователем
ID As Integer
Name As String * 20
End Type
Dim MyRecord As Record ' Объявляем переменные
Open "TESTFILE" For Random As #1 Len = Len(MyRecord)
Do While Not EOF(1) ' Повторяем, пока не конец файла
Get #1, , MyRecord ' Читаем следующую запись
Debug.Print Seek(1) ' Выводим в окно Debug текущий номер записи.
Loop
Close #1 ' Закрываем файл

```

Sgn(number) - Возвращает знак числа number.

Пример:

```

Dim MyVar1, MyVar2, MyVar3, MySign
MyVar1 = 12
MyVar2 = -2.4
MyVar3 = 0
MySign = Sgn(MyVar1) ' Возвратит 1.
MySign = Sgn(MyVar2) ' Возвратит -1.
MySign = Sgn(MyVar3) ' Возвратит 0.

```

Shell(pathname[,windowstyle]) - Запускает программу и возвращает ID программной задачи (program task ID). Если была ошибка, возвратит 0. Параметр windowstyle определяет стиль открытия программы. Может принимать следующие значения:

vbHide Окно будет скрыто

vbNormalFocus Окно получает фокус и отображается без искажений размера

vbMinimizedFocus Окно, после запуска автоматически минимизируется и получает фокус

vbMaximizedFocus Окно разворачивается на весь экран и получает фокус

vbNormalNoFocus Тоже, что и vbNormalFocus, но окно не получает фокуса.

vbMinimizedNoFocus Тоже, что и vbMinimizedFocus, но окно не получает фокуса.

Пример:

```

Dim RetVal

```

```
RetVal = Shell("C:\WINDOWS\CALC.EXE", 1) ' Запускаем калькулятор
```

Sin(number) - Возвращает Double значение, содержащее синус числа number.

Пример:

```
Dim MyAngle, MyCosecant
MyAngle = 1.3 ' Определяем угол в радианах
MyCosecant = 1 / Sin(MyAngle) ' Вычисляем косеконс
```

Space(number) - Возвращает строку, состоящую из number пробелов.

Пример:

```
Dim MyString

' MyString содержит 10 пробелов.
MyString = Space(10)
' Вставляем 10 пробелов между двумя строками
MyString = "Hello" & Space(10) & "World"
```

Spc(n) - Используется вместе с оператором Print # или Print для задания отступа.

Пример:

```
Open "TESTFILE" For Output As #1 ' Открываем файл для записи
Print #1, "10 пробелов между этим местом"; Spc(10); "и этим."
Close #1 ' Закрываем файл
```

ИЛИ

```
Debug.Print Spc(30); "Перед этой строкой 30 пробелов..."
```

Sqr(number) - Возвращает корень числа number.

Пример:

```
Dim MySqr
MySqr = Sqr(4) 'Возвратит 2.
MySqr = Sqr(23) 'Возвратит 4.79583152331272.
MySqr = Sqr(0) 'Возвратит 0.
MySqr = Sqr(-4) 'Генерирует ошибку (корень из отрицательного числа).
```

Str(number) - Возвращает строку, представляющую число.

Пример:

```
Dim MyString
MyString = Str(459) ' Возвратит "459".
MyString = Str(-459.65) ' Возвратит "-459.65".
MyString = Str(459.001) ' Возвратит "459.001".
```

StrComp(string1, string2[, compare]) - Сравнивает две строки. Возвращает:

-1, если string1 < string2

0, если string1 = string2

1, если string > string2

Null, если string1 или string2 содержит Null.

Параметр compare определяет тип сравнения. Может быть одним из следующих значений:

vbBinaryCompare - по умолчанию, бинарное сравнение

vbTextCompare - сравнивает строки без учёта регистра

vbDatabaseCompare - используется в базах данных Microsoft Access.

Пример:

```
Dim MyStr1, MyStr2, MyComp
MyStr1 = "ABCD"
MyStr2 = "abcd"
MyComp = StrComp(MyStr1, MyStr2, 1) ' Возвратит 0.
MyComp = StrComp(MyStr1, MyStr2, 0) ' Возвратит -1.
MyComp = StrComp(MyStr2, MyStr1) ' Возвратит 1.
```

StrConv(string, conversion) - Конвертирует строку в формат, заданный

параметром conversion. Этот параметр может принимать одно из следующих значений:

vbUpperCase - Конвертирует строку в верхний регистр

vbLowerCase - Конвертирует строку в нижний регистр

vbProperCase - Конвертирует первую букву каждого слова в верхний регистр

vbUnicode - Конвертирует строку в формат Unicode, используя кодовую страницу, заданную в системе.

vbFromUnicode - Конвертирует строку из формата Unicode в нормальный

формат, используя кодовую страницу, заданную в системе.

Пример:

```
MyForm.Caption = StrConv("visual BASIC", vbUpperCase)
' сконвертирует строку в верхний регистр
```

String(number, character) - Возвращает строку, состоящую из number числа символов character.

Пример:

```
Dim MyString
MyString = String(5, "*") ' Возвратит "*****".
MyString = String(5, 42) ' Возвратит "*****".
MyString = String(10, "ABC") ' Возвратит "AAAAAAAAAA".
```

Switch(expr-1, value-1[, expr-2, value-2 ... [, expr-n, value-n]]) - Возвращает первое значение, условие которого выполняется.

expr-1, expr-2, ... - условия.

value-1, value-2, ... - значения.

Пример:

Функция MatchUp возвратит строку "Italian", если CityName будет равно "Rome",

или строку "English", если CityName будет равно "London" и т.д.

```
Function MatchUp (CityName As String)
Matchup = Switch(CityName = "London", "English", CityName _
= "Rome", "Italian", CityName = "Paris", "French")
End Function
```

Tab[(n)] - Используется вместе с оператором Print # или Print для задания отступа.

Пример:

```
Debug.Print Tab(10); "10 columns from start."
```

Tan(number) - Возвращает тангенс числа number.

Пример:

```
Dim MyAngle, MyCotangent
MyAngle = 1.3 ' Устанавливаем угол в радианах
```

```
MyCotangent = 1 / Tan(MyAngle) ' Вычисляем котангенс
```

Time - Возвращает Variant значение, содержащее текущее системное время.

Пример:

```
Dim MyTime  
MyTime = Time ' Возвращает текущее системное время.
```

Timer - Возвращает Single, содержащее количество секунд, прошедших после полуночи.

Пример:

```
Dim PauseTime, Start, Finish, TotalTime  
  
If (MsgBox("Нажмите Yes для паузы в 5 сек.", 4)) = vbYes Then  
    PauseTime = 5 ' Устанавливаем длину паузы  
    Start = Timer ' Устанавливаем начальное время  
    Do While Timer < Start + PauseTime  
        DoEvents ' Обрабатываем другие процессы  
    Loop  
    Finish = Timer ' Устанавливаем конечное время  
    TotalTime = Finish - Start ' Вычисляем общее время  
    MsgBox "Пауза в " & TotalTime & " секунд."  
Else  
    End  
End If
```

TimeSerial(hour, minute, second) - Возвратит время, указанное по частям.

Пример:

```
Dim MyTime  
MyTime = TimeSerial(16, 35, 17) 'MyTime содержит время 4:35:17 PM.
```

TimeValue(time) - Возвращает время, заданное строкой.

Пример:

```
Dim MyTime  
MyTime = TimeValue("4:35:17 PM") ' MyTime содержит время 16:35:17
```

Trim(string) - Удаляет лидирующие и замыкающие пробелы в строке string.

Т.е. одновременно выполняет 2 ф-ции: RTrim и LTrim.

Пример:

```
Dim MyString, TrimString
MyString = " <-Trim-> " ' Строка с обеих сторон содержит 1 пробел

TrimString = Trim(MyString) ' TrimString = "<-Trim->"
```

TypeName(varname) - Предоставляет информацию о переменной varname.

Пример:

```
Dim NullVar, MyType, StrVar As String, _
IntVar As Integer, CurVar As Currency
Dim ArrayVar (1 To 5) As Integer

NullVar = Null ' Присваиваем значение Null.
MyType = TypeName(StrVar) ' Возвратит "String".
MyType = TypeName(IntVar) ' Возвратит "Integer".
MyType = TypeName(CurVar) ' Возвратит "Currency".
MyType = TypeName(NullVar) ' Возвратит "Null".
MyType = TypeName(ArrayVar) ' Возвратит "Integer()".
MyType = TypeName(Label1) ' Возвратит "Label"
```

UBound(arrayname[, dimension]) - Возвращает число Long, содержащее максимально возможный индекс массива arrayname указанной размерности.

Пример:

```
Dim Upper
Dim MyArray(1 To 10, 5 To 15, 10 To 20) ' Объявляем массивы
Dim AnyArray(10)

Upper = UBound(MyArray, 1) ' Возвратит 10.
Upper = UBound(MyArray, 3) ' Возвратит 20.
Upper = UBound(AnyArray) ' Возвратит 10.
```

UCase(string) - Переводит строку в верхний регистр.

Пример:

```
Dim LowerCase, UpperCase

LowerCase = "Hello World 1234" ' Строка для конвертирования
UpperCase = UCase(LowerCase) ' Возвратит "HELLO WORLD 1234".
```

Val(string) - Превращает строку в число.

Пример:

```
Dim MyValue
MyValue = Val("2457") ' Возвратит 2457.
MyValue = Val(" 2 45 7") ' Возвратит 2457.
MyValue = Val("24 and 57") ' Возвратит 24.
MyValue = Val("") ' Возвратит 0.
MyValue = Val("laja") ' Возвратит 0.
```

VarType(varname) - Возвращает тип переменной varname. Возможные значения функции:

vbEmpty - Empty (Не инициализирована)

vbNull - Null (нет правильного значения)

vbInteger - Integer

vbLong - Long integer

vbSingle - Single

vbDouble - Double

vbCurrency - Currency

vbDate - Date значение

vbString - String

vbObject - Object

vbError - значение Error

vbBoolean - Boolean значение

vbVariant - Variant (используется только для Variant массивов)

vbDataObject - Объект доступа к данным

vbDecimal - десятичное число

vbByte - Byte

vbArray - массив

Пример:

```
Dim IntVar, StrVar, DateVar, MyCheck
IntVar = 459
StrVar = "Hello World"
DateVar = #2/12/69#
MyCheck = VarType(IntVar) ' Возвратит 2.
```

```
MyCheck = VarType(DateVar) ' Возвратит 7.  
MyCheck = VarType(StrVar) ' Возвратит 8.
```

Weekday(date, [firstdayofweek]) - Возвращает день недели, которому соответствует указанная дата. Возможные значения функции:

vbUseSystem = 0 Использует в качестве первого дня недели день, используемый в системе. (В Америке неделя начинается с Воскресенья, поэтому по умолчанию firstdayofweek = 1).

vbSunday = 1 Воскресенье

vbMonday = 2 Понедельник

vbTuesday = 3 Вторник

vbWednesday = 4 Среда

vbThursday = 5 Четверг

vbFriday = 6 Пятница

vbSaturday = 7 Суббота

Пример:

```
Dim MyDate, MyWeekDay  
MyDate = Date ' Присваиваем текущую дату  
MyWeekDay = Weekday(MyDate, 0)  
' MyWeekDay содержит сегодняшний день недели
```

Year(date) - Возвращает количество годов, содержащихся в параметре date.

Пример:

```
Dim MyDate, MyYear  
MyDate = #February 12, 1969# ' Присваиваем дату  
MyYear = Year(MyDate) ' MyYear содержит 1969.
```

13 Приложение: Дополнительные материалы по вычислению некоторых функций

В данном приложении приводится раздел справки VBA, в котором приводятся выражения, позволяющие вычислить значения некоторых функций.

Справочные выражения
$\text{Sec}(X) = 1 / \text{Cos}(X)$
$\text{Cosec}(X) = 1 / \text{Sin}(X)$
$\text{Cotan}(X) = 1 / \text{Tan}(X)$
$\text{Arcsin}(X) = \text{Atn}(X / \text{Sqr}(-X * X + 1))$
$\text{Arccos}(X) = \text{Atn}(-X / \text{Sqr}(-X * X + 1)) + 2 * \text{Atn}(1)$
$\text{Arcsec}(X) = \text{Atn}(X / \text{Sqr}(X * X - 1)) + \text{Sgn}((X) - 1) * (2 * \text{Atn}(1))$
$\text{Arccosec}(X) = \text{Atn}(X / \text{Sqr}(X * X - 1)) + (\text{Sgn}(X) - 1) * (2 * \text{Atn}(1))$
$\text{Arccotan}(X) = \text{Atn}(X) + 2 * \text{Atn}(1)$
$\text{HSin}(X) = (\text{Exp}(X) - \text{Exp}(-X)) / 2$
$\text{HCos}(X) = (\text{Exp}(X) + \text{Exp}(-X)) / 2$
$\text{HTan}(X) = (\text{Exp}(X) - \text{Exp}(-X)) / (\text{Exp}(X) + \text{Exp}(-X))$
$\text{HSec}(X) = 2 / (\text{Exp}(X) + \text{Exp}(-X))$
$\text{HCosec}(X) = 2 / (\text{Exp}(X) - \text{Exp}(-X))$
$\text{HCotan}(X) = (\text{Exp}(X) + \text{Exp}(-X)) / (\text{Exp}(X) - \text{Exp}(-X))$
$\text{HArcsin}(X) = \text{Log}(X + \text{Sqr}(X * X + 1))$
$\text{HArccos}(X) = \text{Log}(X + \text{Sqr}(X * X - 1))$
$\text{HArctan}(X) = \text{Log}((1 + X) / (1 - X)) / 2$
$\text{HArcsec}(X) = \text{Log}((\text{Sqr}(-X * X + 1) + 1) / X)$
$\text{HArccosec}(X) = \text{Log}((\text{Sgn}(X) * \text{Sqr}(X * X + 1) + 1) / X)$
$\text{HArccotan}(X) = \text{Log}((X + 1) / (X - 1)) / 2$
$\text{LogN}(X) = \text{Log}(X) / \text{Log}(N)$

14 Приложение: Операторы VB

AppActivate title[, wait] - Активизирует окно приложения с заголовком title. Вместо заголовка может быть использовано ID задачи, полученное при помощи функции Shell.

Пример:

```
Dim MyAppID, ReturnValue
AppActivate "Microsoft Word" 'Активизируется Microsoft Word
MyAppID = Shell("C:\WORD\WINWORD.EXE", 1) ' Запускаем Microsoft Word.
AppActivate MyAppID 'Активизируется Microsoft Word
```

Beep - Проигрывает звук, установленный в системе как "Стандартный звук" (в панели управления->Звук). Если такой не установлен, то играетя PC спикер.

Пример:

```
Beep
```

[Call] name [argumentlist] - Вызывает процедуру или функцию. Оператор Call может быть опущен.

Пример:

```
Call PrintToDebugWindow ("Hello World")
'Вызываем процедуру. Скобки обязательны!
PrintToDebugWindow "Hello World"
'To же самое. Скобки можно опустить.
Sub PrintToDebugWindow (AnyString)
Debug.Print AnyString ' Печатаем в окно Debug.
End Sub
```

ChDir path - Смена текущего каталога.

Пример:

```
' Изменяем текущую директорию на "MYDIR".
ChDir "MYDIR"
' Пусть диск "C:" является текущим диском. Следующий оператор
' изменяет текущий каталог на "D:\WINDOWS\SYSTEM", но
' текущим диском по-прежнему остаётся "C:".
ChDir "D:\WINDOWS\SYSTEM"
```

ChDrive drive - Смена текущего диска.

Пример:

```
ChDrive "D"
```

Close [number] - Закрывает файл, открытый оператором Open под номером number.

Пример:

```
Open "TESTFILE" For Output As #3
Print #3, "Hello World"
Close #3
```

[Public | Private] **Const** constname [As type] = expression - Объявляет константу.

Пример:

```
' Константа типа Private (используется по умолчанию).
Const MyVar = 459
' Константа типа Public.
Public Const MyString = "HELP"
' Константа Private, типа Integer.
Private Const MyInt As Integer = 5
```

Date = date - изменяет системную дату.

Пример:

```
Dim MyDate
MyDate = #February 12, 1985# ' Присваиваем дату переменной
Date = MyDate ' Изменяем системную дату
```

[Public | Private] **Declare** Sub name Lib "libname" [Alias "aliasname"] [(arglist)]

или

[Public | Private] **Declare** Function name Lib "libname" _

[Alias "aliasname"] [(arglist)] [As type]

Этот оператор имеет 2 синтаксиса. Первый объявляет процедуру, находящуюся в DLL библиотеке (которая объявлена в ней как export). Второй

объявляет функцию. Замечание: если вы объявляете функцию в разделе формы или компонента ActiveX, то она должна быть объявлена как Private, если в модуле, то Public.

Пример:

```
' Это находится в модуле:
Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA" _
    (ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) _
    As Long
' А это в разделе формы:
GetWindowText Form1.Hwnd, strCapt, 255
' Теперь переменная strCapt содержит заголовок окна Form1
```

DefBool letterrange[, letterrange] ...

DefByte letterrange[, letterrange] ...

DefInt letterrange[, letterrange] ...

DefLng letterrange[, letterrange] ...

DefCur letterrange[, letterrange] ...

DefSng letterrange[, letterrange] ...

DefDbl letterrange[, letterrange] ...

DefDec letterrange[, letterrange] ...

DefDate letterrange[, letterrange] ...

DefStr letterrange[, letterrange] ...

DefObj letterrange[, letterrange] ...

DefVar letterrange[, letterrange] ...

С помощью этих операторов можно установить тип переменных по умолчанию. letterrange - диапазон символов, с которых должны начинаться имена переменных. Операторы должны находиться в разделе модуля.

Пример:

```
DefInt A-K
' Переменные, имена которых начинаются с букв L до K будут
' по умолчанию объявлены как Integer. А от L до Z - как String.
' Напомню, что пример работает при выключенном Option Explicit.
DefStr L-Z
CalcVar = 4 ' Объявляется как Integer.
```

```
StringVar = "Hello there" ' Объявляется как String.
```

DeleteSetting appname, section[, key] - удаляет ранее записанную настройку из реестра. (SaveSetting сохраняет настройку, а функция GetSetting загружает).

Настройки записываются по адресу:
HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings
В этом ключе создаётся ещё один ключ с названием appname.

Пример:

```
' Записываем что-нибудь, в нашем случае - это положение окна
SaveSetting "MyApp", "Startup", "Top", Form1.Top
SaveSetting "MyApp", "Startup", "Left", Form1.Left
' Удаляем только что записанные настройки
DeleteSetting "MyApp", "Startup"
```

Dim [WithEvents] varname([[subscripts]]) [As [New] type] [, [WithEvents] _
varname([[subscripts]]) [As [New] type]] ...

Этот оператор используется для объявления переменных.

Пример:

```
' AnyValue и MyValue объявляются как Variant по умолчанию
' и им присваивается начальное значение Empty.
Dim AnyValue, MyValue
' Объявляем переменную типа Integer
Dim Number As Integer
' Здесь объявление происходит в одной строчке. Переменная AnotherVar
' объявляется как Variant, т.к. её тип опущен.
Dim AnotherVar, Choice As Boolean, BirthDate As Date
' DayArray - массив, состоящий из 51-го элемента (от 0 до 50).
' Если в модуле написать Option Base 1, то индексы всех массивов будут
' начинаться с единицы. По умолчанию Option Base установлен в 0
Dim DayArray(50)
' Матрица 4 на 5.
Dim Matrix(3, 4) As Integer
' BirthDay - массив, с индексами от 1 до 10.
Dim BirthDay(1 To 10) As Date
' MyArray - динамический массив типа Variant.
Dim MyArray()
```

Do...Loop – конструкция организации циклов

End

End Function

End If

End Property

End Select

End Sub

End Type

End With

Заканчивает процедуру или блок.

End - немедленно завершает выполнение программы. Закрывает все открытые файлы и очищает все переменные.

End Function - необходим для завершения функции.

End If - необходим для завершения проверки условия оператором If.

End Property - необходим для завершения процедур Property Let, Property Get или Property Set.

End Select - необходим для завершения оператора Select Case.

End Sub - необходим для завершения процедур.

End Type - необходим для завершения определения пользовательского типа (оператор Type)

End With - необходим для завершения оператора With.

Пример:

```
Sub Form_Load
Dim Password, Pword
PassWord = "Swordfish"
Pword = InputBox("Type in your password")
If Pword <> PassWord Then
MsgBox "Sorry, incorrect password"
End
End If
End Sub
```

[Public | Private] **Enum** name

membername [= constantexpression]

membername [= constantexpression]

...

End Enum

Объявляет перечисляемый тип.

Пример:

```
Public Enum InterfaceColors
    icMistyRose = &HE1E4FF&
    icSlateGray = &H908070&
    icDodgerBlue = &HFF901E&
    icDeepSkyBlue = &HFFBF00&
    icSpringGreen = &H7FFF00&
    icForestGreen = &H228B22&
    icGoldenrod = &H20A5DA&
    icFirebrick = &H2222B2&
End Enum
```

Erase arraylist - сбрасывает все значения массива фикс. размера и освобождает неиспользуемую память динамического массива.

Пример:

```
Dim a(5) As Integer
Dim Dyn()
a(2) = 4
ReDim Dyn(10) ' Резируем память для 11-ти элементов
Erase a ' Сбрасываем все значения массива a
Erase Dyn ' Освобождаем память
Form1.Caption = a(2) ' Выведет 0
Form1.Caption = Dyn(2) ' Выдаст ошибку Subscript Out Of Range.
```

Error errornumber - Симулирует наступление ошибки под номером errornumber. Полное описание всех кодов ошибок вы найдёте в п. 4.

Пример:

```
On Error Resume Next ' Устанавливаем обработку ошибки
Error 11 ' Симулируем ошибку "Деление на ноль".
```

[Public] Event procedurename [(arglist)] - Используется для объявления события в компоненте или модуле класса.

Пример:

```
Public Event UpdateTime (ByVal dblJump As Double)
```

Exit Do

Exit For

Exit Function

Exit Property

Exit Sub

Используется для досрочного выхода из циклов, функций, свойств и процедур.

Пример:

```
Sub ExitStatementDemo()  
Dim I, MyNum  
Do ' Бесконечный цикл  
For I = 1 To 1000 ' Повторяем 1000 раз  
MyNum = Int(Rnd * 1000) ' Генерируем случайное число  
Select Case MyNum ' Проверяем сгенерированное значение  
Case 7: Exit For ' Если 7, выходим из For...Next.  
Case 29: Exit Do ' Если 29, выходим из Do...Loop.  
Case 54: Exit Sub ' Если 54, выходим из процедуры.  
End Select  
Next I  
Loop  
End Sub
```

FileCopy source, destination - копирует файл source в файл destination.

Пример:

```
FileCopy "C:\Windows\Win.ini", "C:\Backups\Win.bak"
```

For Each...Next – конструкция организации цикла «для каждого»

For...Next – конструкция организации цикла со счетчиком

```
[Public | Private | Friend] [Static] Function name [(arglist)] [As type]
[statements]
[name = expression]
[Exit Function]
[statements]
[name = expression]
End Function
```

Этот оператор объявляет функцию с именем name и параметрами arglist.

Пример:

```
Function CalculateSquareRoot (NumberArg As Double) As Double
If NumberArg < 0 Then ' проверяем аргумент
Exit Function ' Выходим из функции (возвращаясь на то место,
' откуда она была вызвана)
Else
CalculateSquareRoot = Sqr(NumberArg)
' Возвращает квадратный корень аргумента
End If
End Function
```

Get [#]filename, [resnumber], varname - Читает данные из открытого файла с номером filename в переменную varname. resnumber - задаёт позицию начала чтения.

Пример:

```
' Тип, определяемый пользователем (должен находиться в модуле) .
Type Record
ID As Integer
Name As String * 20
End Type
Dim MyRecord As Record, Position ' Объявляем переменные
' Открываем файл с произвольным доступом (Random)
Open "TESTFILE" For Random As #1 Len = Len(MyRecord)
Position = 3 ' Устанавливаем позицию начала чтения
Get #1, Position, MyRecord ' Читаем одну запись
Close #1 ' Закрываем файл
```

GoSub line

...

line

...

Return

Это устаревший оператор перехода в псевдо-подпрограмму с возвращением по Return.

GoTo line - оператор для перехода на определённую метку. Пользуйтесь им только в крайних случаях.

Пример:

```
Dim a, b As Integer
b = 5
a = 1
If a = 1 Then GoTo МЕТКА
b = 10 ' Этот участок никогда не выполнится
МЕТКА:
Debug.Print b ' Выведет цифру 5
```

If condition Then [statements] [Else elsestatements] – конструкция организации ветвления

Implements [InterfaceName | Class] - Определяет интерфейс или класс, который будет включён в модуль класса. В качестве параметра задаётся либо имя интерфейса, либо имя класса, методы которого будут включены в класс, в котором определён оператор.

Input #filenumber, varlist - Последовательно читает данные из файла и записывает их в переменную(ые) varlist.

Пример:

```
Dim MyString, MyNumber
Open "TESTFILE" For Input As #1 ' Открываем файл для чтения
Do While Not EOF(1) ' Цикл, пока не конец файла
```

```
Input #1, MyString, MyNumber ' Читаем данные в две переменные
Debug.Print MyString, MyNumber ' Печатаем полученные данные в Dubug
Loop
Close #1 ' Закрываем файл
```

Kill pathname - Оператор удаляет файл с диска. В качестве pathname можно использовать маску.

Пример:

```
' Удаляем файл
Kill "TestFile"
' Удаляем все файлы с расширением txt, находящиеся в текущей директории
Kill "*.TXT"
```

[Let] varname = expression - Это необязательный оператор присваивания, который перекачал со старинного бейсика.

Пример:

```
Dim MyStr As String
Let MyStr = "Hello World" ' Присваиваем с использованием оператора Let
MyStr = "Hello World" ' То же самое, но без оногo.
```

String Like Pattern- оператор для проверки строки String на маску Pattern. Это очень мощный оператор, почти аналог регулярных выражений в Perl.

Пример:

```
Dim MyCheck
MyCheck = "aBBBa" Like "a*a" ' Возвратит True.
MyCheck = "F" Like "[A-Z]" ' Возвратит True.
MyCheck = "F" Like "[!A-Z]" ' Возвратит False.
MyCheck = "a2a" Like "a#a" ' Возвратит True.
MyCheck = "aM5b" Like "a[L-P]#[!c-e]" ' Возвратит True.
MyCheck = "BAT123khg" Like "B?T*" ' Возвратит True.
MyCheck = "CAT123khg" Like "B?T*" ' Возвратит False.
myString = "312T-87GD-8922"
If myString Like "###[A-Z]-##[A-Z][A-Z]-####" Then ...
```

Спец-символы в маске:

? Любой одиночный символ

* Ноль или более символов

Любая одиночная цифра (0–9).

[charlist] Любой одиночный символ в классе символов (списке)

[!charlist] Любой одиночный символ не принадлежащий классу символов

Line Input #filename, varname - Последовательно читает одну строку из открытого файла в переменную String.

Пример:

```
Dim TextLine
Open "TESTFILE" For Input As #1 ' Открываем файл
Do While Not EOF(1) ' Зацикливаемся, пока не конец файла
Line Input #1, TextLine ' Читаем в переменную
Debug.Print TextLine ' Печатаем в окно Debug
Loop
Close #1 ' Закрываем файл
```

Load object - загружает форму или компонент в память. Пример, хоть и большой, но очень простой и понятный.

Пример:

```
Private Sub Form_Click ()
Dim Answer, Msg As String ' Объявляем переменные
Unload Form1 ' Выгружаем форму
Msg = _
"Form1 только что была выгружена. Нажмите Да, чтобы загрузить её и "
Msg = Msg & "показать. Нажмите Нет чтобы загрузить её и "
Msg = Msg & "оставить невидимой."
Answer = MsgBox(Msg, vbYesNo) ' Выводим окно сообщения
If Answer = vbYes Then ' Проверяем выбор пользователя
Show ' если Да, то показываем форму
Else
Load Form1 ' Если Нет, то только загружаем
Msg = "Теперь форма загружена. Нажмите ОК, чтобы показать её."
MsgBox Msg ' Выводим сообщение
Show ' Показываем форму
End If
End Sub
```

Lock [#]filename[, recordrange]

...

Unlock [#]filename[, recordrange]

Эти операторы управляют доступом к файлу для других процессов, чтобы те не могли что-либо сделать с компонентом этого файла. В приведённом ниже примере после оператора Lock доступ к записи закрывается, и другие процессы (программы) не смогут изменить эту запись. Запись отпирается оператором Unlock.

Пример:

```
Type Record ' Тип, определённый пользователем (находится в модуле)
ID As Integer
Name As String * 20
End Type
Dim MyRecord As Record, RecordNumber ' Объявляем переменные
' Открываем файл-пример для произвольного доступа
Open "TESTFILE" For Random Shared As #1 Len = Len(MyRecord)
RecordNumber = 4 ' Устанавливаем позицию записи
Lock #1, RecordNumber ' Запираем эту запись
Get #1, RecordNumber, MyRecord ' Читаем запись
MyRecord.ID = 234 ' Модифицируем её
MyRecord.Name = "John Smith"
Put #1, RecordNumber, MyRecord ' Записываем изменённую запись
Unlock #1, RecordNumber ' Отпираем запись
Close #1 ' Закрываем файл
```

LSet stringvar = string - Выравнивает строку по левой стороне при присваивании ей значения, меньшего по размеру, чем исходная строка. Вместо строк могут использоваться определённые пользователем типы.

Пример:

```
Dim MyString
MyString = "0123456789" ' Инициализируем строку
Lset MyString = "<-Left" ' MyString содержит "<-Left"
```

Mid(stringvar, start[, length]) = string - заменяет символы в строке stringvar начиная с позиции start символами, содержащимися в строке string.

Пример:

```
Dim MyString
MyString = "The dog jumps" ' Инициализируем строку
Mid(MyString, 5, 3) = "fox" ' MyString = "The fox jumps".
Mid(MyString, 5) = "cow" ' MyString = "The cow jumps".
Mid(MyString, 5) = "cow jumped over" ' MyString = "The cow jumpe".
Mid(MyString, 5, 3) = "duck" ' MyString = "The duc jumpe".
```

MkDir path - создаёт каталог path.

Пример:

```
MkDir "MYDIR" ' Создаёт новую папку в текущем каталоге.
MkDir "C:\MYDIR" ' Создаёт новую папку в корневом каталоге диска C.
```

Name oldpathname As newpathname - переименовывает файл или каталог с именем oldpathname в файл или каталог с именем newpathname.

Пример:

```
Dim OldName, NewName
OldName = "OLDFILE": NewName = "NEWFILE" ' Устанавливаем имена
Name OldName As NewName ' Переименовываем
OldName = "C:\MYDIR\OLDFILE": NewName = "C:\YOURDIR\NEWFILE"
Name OldName As NewName ' Перемещаем и переименовываем файл
```

On Error GoTo line

On Error Resume Next

On Error GoTo 0

Устанавливает обработчик ошибок на процедуру. Первый оператор указывает метку, на которую будет передано управление при возникновении ошибки. С помощью второго оператора можно сделать так, чтобы при возникновении ошибки программа продолжала работать. И, наконец, третий позволяет отключить все ранее установленные обработчики ошибок.

Пример:

```
Dim strVar As String
' Устанавливаем обработчик ошибки
On Error GoTo OSHIBKA
'Открываем несуществующий файл.Возникает ошибка №53 "Файл не найден" и
```



```

' программа передаёт управление на метку OSHIBKA
Open "C:\Windows\MyIni.ini" For Input As #3
Input #3, strVar
Close #3
Exit Sub ' Нужно выйти из процедуры, иначе появится ненужный MsgBox

OSHIBKA:
If Err.Number = 53 Then ' Объект Err - содержит свойства ошибки
MsgBox "Файл не найден!", vbCritical, "error"
Else
MsgBox "Какая-то другая ошибка!", vbCritical, "error"
End If

```

Open pathname For mode [Access access] [lock] As [#]filenumber [Len=reclength] - открывает файл для чтения, записи или для произвольного доступа.

Пример:

```

' Открываем файл для последовательного чтения
Open "TESTFILE" For Input As #1
' Закрываем файл перед тем, как открыть его снова другим методом
Close #1
' Открываем для двоичного доступа и только для записи
Open "TESTFILE" For Binary Access Write As #1
' Закрываем файл перед тем, как открыть его снова другим методом
Close #1
' Этот пример открывает файл для произвольного доступа.
' Файл содержит записи определённого пользователем типа.
Type Record
ID As Integer
Name As String * 20
End Type
Dim MyRecord As Record ' Объявляем переменную типа запись
Open "TESTFILE" For Random As #1 Len = Len(MyRecord)
' Закрываем файл перед тем, как открыть его снова другим методом
Close #1
' Открываем файл для последовательной записи.
' Причём другие процессы также могут записывать или читать этот файл.
Open "TESTFILE" For Output Shared As #1
' Закрываем файл перед тем, как открыть его снова другим методом
Close #1
'Этот оператор открывает файл для двоичного доступа только для чтения,

```

```
' Причём другие процессы не могут обращаться к этому файлу!  
Open "TESTFILE" For Binary Access Read Lock Read As #1
```

Option Base {0 | 1} - задаёт нижний предел для массивов. Оператор действует на уровне модуля. По умолчанию все массивы начинают свои индексы с нуля.

Пример:

```
Option Base 1 ' Устанавливаем границу для массивов равной единице  
Dim Lower  
Dim MyArray(20), TwoDArray(3, 4) ' Объявляем массивы  
Dim ZeroArray(0 To 5) ' Этот массив всё равно начнётся с нуля  
' Используем ф-цию LBound для выявления нижней границы массива  
Lower = LBound(MyArray) ' Возвратит 1.  
Lower = LBound(TwoDArray, 2) ' Возвратит 1.  
Lower = LBound(ZeroArray) ' Возвратит 0.
```

Option Compare {Binary | Text | Database} - То же, что и выше, только задаёт тип сравнения, который будет использоваться по умолчанию. По умолчанию используется бинарный тип сравнения.

Пример:

```
' Устанавливаем бинарный тип сравнения.  
Option Compare Binary ' Теперь, строка "AAA" меньше, чем "aaa".  
' Устанавливаем текстовый тип сравнения.  
Option Compare Text ' Теперь, строка "AAA" равна строке "aaa".
```

Option Explicit - Если задать этот оператор, то нельзя будет использовать необъявленные ранее переменные, например, как в паскале.

Пример:

```
Option Explicit  
Dim MyVar ' Объявляем переменную  
MyInt = 10  
' MyInt - необъявленная переменная, поэтому произойдёт ошибка  
MyVar = 10 ' Здесь всё ОК
```

Option Private Module - делает модуль приватным. Т.е. его методы и функции не могут быть использованы в других подключённых проектах.

Пример:

```
Option Private Module ' Теперь модуль Module - приватный
```

Print #filenumber, [outputlist] - записывает переменную(ые) outputlist в файл последовательного доступа.

Пример:

```
Open "TESTFILE" For Output As #1 ' Открыть файл для записи
Print #1, "This is a test"
' Записать в файл строку текста
Print #1,
' Записать в файл пустую строку
Print #1, "Zone 1"; Tab ; "Zone 2"
' Записать информацию в две колонки (зоны)
Print #1, "Hello" ; " " ; "World"
' Разделить строки пробелом
Print #1, Spc(5) ; "5 leading spaces "
' Записать 5 лидирующих пробелов
Print #1, Tab(10) ; "Hello"
' Записать слово в десятый столбец

' Присвоим Boolean, Date, Null и Error значения.
Dim MyBool, MyDate, MyNull, MyError
MyBool = False
MyDate = #February 12, 1969#
MyNull = Null
MyError = CVErr(32767)
' True, False, Null и Error будут преобразованы, основываясь
' на локальных настройках вашей системы.
' Дата запишется, используя стандартный короткий формат
Print #1, MyBool ; " это Boolean значение"
Print #1, MyDate ; " это Дата"
Print #1, MyNull ; " это значение Null"
Print #1, MyError ; " это значение переменной Error"
Close #1 ' Закрываем файл
```

Private [WithEvents] varname[([subscripts])] [As [New] type] [, [WithEvents] varname[([subscripts])] [As [New] type]] . . .

Объявляет приватные переменные и выделяет под них место в памяти.

Пояснение: приватные отличаются от обычных, объявленных оператором Dim тем, что не может объявлять переменные внутри процедуры или функции.

При объявлении в разделе глобальных объявлений они равнозначны.

Пример:

```
Private Number As Integer ' Объявляем приватную переменную Number
Private NameArray(1 To 5) As String ' Приватный массив
' Несколько объявлений, 2 типа Variant и 1 Integer. Все приватные.
Private MyVar, YourVar, ThisVar As Integer
```

```
[Public | Private | Friend] [Static] Property Get name [(arglist)] [As type]
[statements]
[name = expression]
[Exit Property]
[statements]
[name = expression]
End Property
```

Объявляет процедуру получения значения свойства. Эта процедура срабатывает, когда пользователь обращается к свойству.

Пример:

```
Dim CurrentColor As Integer
Const BLACK = 0, RED = 1, GREEN = 2, BLUE = 3
' Возвращает текущий цвет карандаша (Pen) в виде строки
Property Get PenColor() As String
Select Case CurrentColor
Case RED
PenColor = "Красный"
Case GREEN
PenColor = "Зелёный"
Case BLUE
PenColor = "Синий"
End Select
End Property
' Этот код обращается к свойству и вызывает процедуру Property Get
ColorName = PenColor
```

```
[Public | Private | Friend] [Static] Property Let name ([arglist,] value)
[statements]
[Exit Property]
```

[statements]

End Property

Объявляет процедуру присваивания значения свойству. Эта процедура срабатывает, когда пользователь присваивает значение свойству.

Пример:

```
Dim CurrentColor As Integer
Const BLACK = 0, RED = 1, GREEN = 2, BLUE = 3
' Устанавливает цвет карандаша (Pen) по цвету, указанному в виде строки
' Переменной уровня модуля CurrentColor присваивается
' текущий цвет карандаша
Property Let PenColor(ColorName As String)
Select Case ColorName ' Проверяем строку, определяющую цвет
Case "Красный"
CurrentColor = RED ' Красный цвет
Case "Зелёный"
CurrentColor = GREEN ' Зелёный
Case "Синий"
CurrentColor = BLUE ' Синий
Case Else
CurrentColor = BLACK ' Присваиваем значение по умолчанию
End Select
End Property
' Этот код вызывает процедуру Property Let
PenColor = "Красный"
```

[Public | Private | Friend] [Static] **Property Set** name ([arglist,] reference)

[statements]

[Exit Property]

[statements]

End Property

Объявляет процедуру присваивания свойству ссылки на объект. Эта процедура срабатывает, когда пользователь присваивает свойству ссылку на объект. Например, если тип свойства - Picture.

Пример:

```
' Свойство DownPicture может иметь различные картинки
' CurrentPic - хранит рисунок
```

```
Property Set DownPicture(P As Picture)
Set CurrentPic = P ' Присваиваем новый рисунок
End Property
```

Public [WithEvents] varname([[subscripts]]) [As [New] type] [, [WithEvents] varname([[subscripts]]) [As [New] type]] . . .

Объявляет глобальную переменную. Объявленная таким способом переменная становится доступной из всех модулей и форм проекта. Если переменная объявлена в разделе глобальных объявлений, то доступ к ней осуществляется просто по её имени. Если же она объявлена в коде формы, то доступ к ней из других форм и модулей осуществляется так: `ИмяФормы.ИмяПеременной`.

Пример:

```
Public Number As Integer ' Глобальная переменная типа Integer
Public NameArray(1 To 5) As String ' Глобальный массив
' Несколько объявлений, 2 типа Variant и 1 Integer. Все глобальные.
Public MyVar, YourVar, ThisVar As Integer
```

Put [#]filenumber, [recnumber], varname - Записывает данные в файл с номером filenumber из переменной varname. recnumber - задаёт позицию начала записи.

Пример:

```
Type Record ' Определённый пользователем тип
ID As Integer
Name As String * 20
End Type
Dim MyRecord As Record, RecordNumber ' Объявляем переменные
' Открываем файл для произвольного доступа
Open "TESTFILE" For Random As #1 Len = Len(MyRecord)
For RecordNumber = 1 To 5 ' Циклимся 5 раз
MyRecord.ID = RecordNumber ' Устанавливаем компоненту ID
MyRecord.Name = "My Name" & RecordNumber ' Создаём строку
Put #1, RecordNumber, MyRecord ' Записываем запись в файл
Next RecordNumber
Close #1 ' Закрываем файл
```

RaiseEvent eventname [(argumentlist)] - Запускает событие, определённое в компоненте, форме или документе.

Пример:

```
' В этом примере при нажатии на UserControl будет
' генерироваться нажатие клавиши с кодом 101
Event Click()
Event KeyPress(KeyAscii As Integer)
Private Sub UserControl_Click()
RaiseEvent KeyPress(101) ' Вместо щелчка будет происходить KeyPress
End Sub
Private Sub UserControl_KeyPress(KeyAscii As Integer)
RaiseEvent KeyPress(KeyAscii) ' Обычный KeyPress
End Sub
' Теперь поместите этот код в форму, на которой расположен UserControl
Private Sub UserControl1_KeyPress(KeyAscii As Integer)
MsgBox "Hello World " & KeyAscii
End Sub
' Запустите проект и щёлкните по UserControl'у.
```

Randomize [number] - Инициализирует генератор случайных чисел. Если этот оператор не поместить перед функцией Rnd, то при каждом запуске приложения будут генерироваться одни и те же случайные числа.

Пример:

```
Dim MyValue
Randomize ' Инициализирует генератор случайных чисел
MyValue = Int((6 * Rnd) + 1) ' Генерируем случайное число от 1 до 6
```

ReDim [Preserve] varname(subscripts) [As type] [, varname(subscripts) [As type]]

...

Изменяет размер динамического массива. Замечание: Для этого оператора Option Base установлен в единицу!

Пример:

```
Dim MyArray() As Integer ' Объявляем динамический массив
Redim MyArray(5) ' Выделяем место для ПЯТИ элементов
For I = 1 To 5 ' Циклимся 5 раз
MyArray(I) = I ' Присваиваем значения компонентам массива
Next I
```

```
' Следующий оператор изменяет размер массива и очищает его
Redim MyArray(10) ' Теперь размер массива = 10
For I = 1 To 10 ' Циклимся 10 раз
MyArray(I) = I ' Присваиваем значения компонентам массива
Next I
' А этот оператор изменяет размер, но не очищает его
Redim Preserve MyArray(15) ' Теперь размер = 15
```

Rem comment - старый оператор для создания комментариев в тексте программы. Лучше использовать новый аналог - ' (апостроф).

Пример:

```
Dim MyStr1, MyStr2
MyStr1 = "Hello": Rem Здесь ваш комментарий
MyStr2 = "Goodbye" ' Это тоже комментарий, только без двоеточия
```

Reset - закрывает все файлы, открытые оператором Open.

Пример:

```
Dim FileNumber
For FileNumber = 1 To 5 ' Цикл выполнится 5 раз
'Откроем файл для записи с номером FileNumber и к имени прибавим номер
Open "TEST" & FileNumber For Output As #FileNumber
Write #FileNumber, "Hello World" ' Записываем данные в файл
Next FileNumber
' Закрываем все открытые файлы и записываем их содержимое на диск
Reset
```

Resume [0]

Resume Next

Resume line

Возобновляет выполнение программы после обработки ошибки.

Пример:

```
Sub ResumeStatementDemo ()
On Error GoTo ErrorHandler ' Включаем обработчик ошибок
Open "TESTFILE" For Output As #1 ' Открываем файл
Kill "TESTFILE" ' Пытаемся удалить открытый файл
Exit Sub ' Выходим из процедуры, чтобы обойти обработчик ошибки
ErrorHandler: ' Начало обработчика
```



```
Select Case Err.Number ' Проверяем номер ошибки
Case 55 ' Ошибка "Файл уже открыт"
Close #1 ' Закрываем открытый файл
Case Else
' Здесь обрабатываем другие ситуации...
End Select
Resume ' Продолжаем выполнение программы с той же строки,
' в которой произошла ошибка
End Sub
```

Rmdir path - удаляет директорию с диска.

Пример:

```
Rmdir "C:\Temp" ' Удаляет директорию Temp с корневого диска
```

RSet stringvar = string - Выравнивает строку по правой стороне при присваивании ей значения, меньшего по размеру, чем исходная строка.

Вместо строк могут использоваться определённые пользователем типы.

Пример:

```
Dim MyString
MyString = "0123456789" ' Инициализируем строку
Rset MyString = "Right->" ' MyString содержит " Right->"
```

SavePicture picture, stringexpression - сохраняет изображение, находящееся в данный момент в свойствах Picture или Image объекта X в файл.

Пример:

```
' Поместите на форму PictureBox и загрузите туда любую картинку
Private Sub Form_Click()
AutoRedraw = True ' Включаем перерисовку окна
Picture1.AutoRedraw = True
Form1.ScaleMode = vbPixels ' Ставим вид масштаба в пиксели
Picture1.ScaleMode = vbPixels
Form1.PaintPicture Picture1, 0, 0, 100, 100, 0, 0, 100, 100, vbSrcCopy
Form1.Refresh
MsgBox "Нажмите ОК для сохранения"
SavePicture Image, "C:\TEST.BMP" ' Сохраняем картинку
End Sub
```

SaveSetting appname, section, key, setting - Записывает настройку в реестр. Напомню, что функция DeleteSetting удаляет настройку, а функция GetSetting загружает. Настройки записываются по адресу: HKEY_CURRENT_USER\SOFTWARE\VB and VBA Program Settings В этом ключе создаётся ещё один ключ с названием appname.

Пример:

```
' Сохраняем положение левой координаты формы в реестр
SaveSetting "MyApp", "Startup", "Left", Form1.Left
```

Seek [#]filenumber, position - Устанавливает позицию для чтения/записи в файле, открытым оператором Open.

Пример:

```
Type Record ' Определённый пользователем тип
ID As Integer
Name As String * 20
End Type

' Для файлов, открытых для произвольного доступа,
' Seek устанавливает следующую запись
Dim MyRecord As Record, MaxSize, RecordNumber ' Объявляем переменные
' Открываем файл для произвольного доступа
Open "TESTFILE" For Random As #1 Len = Len(MyRecord)
MaxSize = LOF(1) \ Len(MyRecord) ' Получаем количество записей в файле
' Цикл читает записи с конца до начала
For RecordNumber = MaxSize To 1 Step - 1
Seek #1, RecordNumber ' Устанавливаем позицию записи
Get #1, , MyRecord ' Читаем запись
Next RecordNumber
Close #1 ' Закрываем файл

' Для файлов, открытых не для произвольного доступа,
' Seek устанавливает позицию с точностью до байта.
Dim MaxSize, NextChar, MyChar
Open "TESTFILE" For Input As #1 ' Открываем файл
MaxSize = LOF(1) ' Получаем размер файла в байтах
' Цикл читает по букве с конца файла до начала
For NextChar = MaxSize To 1 Step -1
Seek #1, NextChar ' Устанавливаем позицию
MyChar = Input(1, #1) ' Читаем символ
Next NextChar
```

```
Close #1 ' Закрываем файл
```

Select Case – конструкция служит для реализации структуры выбор

SendKeys string[, wait] - посылает одно или более нажатий на клавишу в активное окно. Всё происходит точно так же, как будто всё было введено с клавиатуры.

Пример:

```
Dim ReturnValue, I
ReturnValue = Shell("CALC.EXE", 1) ' Запускаем калькулятор
AppActivate ReturnValue ' Активизируем калькулятор
For I = 1 To 100 ' Циклимся 100 раз
SendKeys I & "{+}", True ' Посылаем нажатие на клавишу в калькулятор
Next I ' Добавляем к значению в калькуляторе переменную I
SendKeys "=", True ' Нажимаем на знак равенства
SendKeys "%{F4}", True ' Посылаем Alt+F4 для закрытия калькулятора
```

Set objectvar = {[New] objectexpression | Nothing} - присваивает переменной ссылку на объект.

Пример:

```
Dim YourObject, MyObject, MyStr
Set MyObject = YourObject ' Присваиваем ссылку
' MyObject и YourObject ссылаются на один и тот же объект
YourObject.Text = "Hello World" ' Инициализируем свойство
MyStr = MyObject.Text ' Возвратит "Hello World".
' Удаляем ссылку на объект
Set MyObject = Nothing ' Освобождаем объект
```

SetAttr pathname, attributes - устанавливает атрибуты файла pathname.

Пример:

```
SetAttr "TESTFILE", vbHidden
' Теперь файл скрытый
SetAttr "TESTFILE", vbHidden + vbReadOnly
' Теперь он и скрытый и только для чтения
```

Static varname[([subscripts])] [As [New] type] [, varname[([subscripts])] [As

[New] type]]...

Объявляет переменную. Переменные Static объявляются внутри процедур и функций и вне их недоступны, но в отличие от обычных локальных переменных они не инициализируются при входе в процедуру или функцию, где они объявлены.

Пример:

```
' Определение функции
Function KeepTotal (Number)
' Только переменная Accumulate сохраняет своё значение между вызовами
Static Accumulate
Accumulate = Accumulate + Number
KeepTotal = Accumulate
End Function

' Определение статической функции
Static Function MyFunction (Arg1, Arg2, Arg3)
' Все локальные переменные сохраняют свои значения между вызовами
Accumulate = Arg1 + Arg2 + Arg3
Half = Accumulate / 2
MyFunction = Half
End Function
```

Stop - С помощью этого оператора можно поставить брекпоинт в программе. Если же программу скомпилировать с этим оператором и запустить, то при выполнении этого оператора программа выдаст ошибку. В отличие от оператора End, оператор Stop не закрывает открытые файлы и не очищает переменные.

Пример:

```
Dim I
For I = 1 To 10 ' Запускаем цикл
Debug.Print I ' Печатаем I в Debug
Stop ' Останавливаемся на каждой итерации
Next I
```

[Private | Public | Friend] [Static] **Sub** name [(arglist)]

[statements]

[Exit Sub]

[statements]

End Sub

Этот оператор объявляет процедуру с именем name и параметрами arglist.

Пример:

```
' Объявляем процедуру
' Процедура имеет 2 аргумента
Sub SubComputeArea (Length, TheWidth)
Dim Area As Double ' Объявляем локальную переменную
If Length = 0 Or TheWidth = 0 Then
' Если любой аргумент равен нулю
Exit Sub ' Сразу выходим из процедуры
End If
Area = Length * TheWidth ' Вычисляем площадь прямоугольника
Debug.Print Area ' Печатаем её в окно Debug
End Sub
```

Time = time - Изменяет системное время на time.

Пример:

```
Dim MyTime
MyTime = #4:35:17 PM# ' Присваиваем время переменной
Time = MyTime ' Изменяем системное время
```

[Private | Public] **Type** varname

elementname [[[subscripts]]] As type

[elementname [[[subscripts]]] As type]

...

End Type

Объявляет новый тип (тип, определённый пользователем), содержащий один или несколько элементов.

Пример:

```
Type EmployeeRecord ' Создаём новый тип
ID As Integer ' Определяем типы значений, входящие в новый тип
Name As String * 20
Address As String * 30
Phone As Long
HireDate As Date
```

```

End Type
Sub CreateRecord()
Dim MyRecord As EmployeeRecord ' Объявляем переменную
' Присваивание значения переменной типа EmployeeRecord
' должно произойти в процедуре
MyRecord.ID = 12003 ' Присваиваем значение элементу переменной
End Sub

```

Unload object - Выгружает объект (форму или компонент). Эта функция является обратной функции Load.

Пример:

```

Private Sub Form_Click ()
Dim Answer, Msg As String ' Объявляем переменные
Unload Form1 ' Выгружаем форму
Msg = _
"Form1 только что была выгружена. Нажмите Да, чтобы загрузить её и "
Msg = Msg & "показать. Нажмите Нет чтобы загрузить её и "
Msg = Msg & "оставить невидимой."
Answer = MsgBox(Msg, vbYesNo) ' Выводим окно сообщения
If Answer = vbYes Then ' Проверяем выбор юзера
Show ' если Да, то показываем форму
Else
Load Form1 ' Если Нет, то только загружаем
Msg = "Теперь форма загружена. Нажмите ОК, чтобы показать её."
MsgBox Msg ' Выводим сообщение
Show ' Показываем форму
End If
End Sub

```

Lock [#]filenumber[, recordrange]

...

Unlock [#]filenumber[, recordrange]

Эти операторы управляют доступом к файлу для других процессов, чтобы те не могли что-либо сделать с компонентом этого файла. В приведённом ниже примере после оператора Lock доступ к записи закрывается, и другие процессы (программы) не смогут изменить эту запись. Запись отпирается оператором Unlock.

Пример:

```
Type Record ' Тип, определённый пользователем (находится в модуле)
ID As Integer
Name As String * 20
End Type

Dim MyRecord As Record, RecordNumber ' Объявляем переменные
' Открываем файл-пример для произвольного доступа
Open "TESTFILE" For Random Shared As #1 Len = Len(MyRecord)
RecordNumber = 4 ' Устанавливаем позицию записи
Lock #1, RecordNumber ' Запираем эту запись
Get #1, RecordNumber, MyRecord ' Читаем запись
MyRecord.ID = 234 ' Модифицируем её
MyRecord.Name = "John Smith"
Put #1, RecordNumber, MyRecord ' Записываем изменённую запись
Unlock #1, RecordNumber ' Отпираем запись
Close #1 ' Закрываем файл
```

While...Wend - Оператор цикла. Отличие от оператора Do While ... Loop заключается в том, что здесь не используется слово Do, и вместо слова Loop используется Wend. Является устаревшей конструкцией.

Пример:

```
Dim Counter
Counter = 0 ' Инициализируем переменные
While Counter < 20 ' Проверяем значение счётчика
Counter = Counter + 1 ' Увеличиваем счётчик
Wend ' Заканчиваем цикл, если Counter > 19
Debug.Print Counter ' Печатаем 20 в окно Debug
' Этот же пример, только с циклом Do While:
Dim Counter
Counter = 0 ' Инициализируем переменные
Do While Counter < 20 ' Проверяем значение счётчика
Counter = Counter + 1 ' Увеличиваем счётчик
Loop ' Заканчиваем цикл, если Counter > 19
Debug.Print Counter ' Печатаем 20 в окно Debug
```

Width #filenumber, width - Устанавливает длину строки при выводе в файл, открытый оператором Open.

Пример:

```

Dim I
Open "TESTFILE" For Output As #1 ' Открываем файл для записи
Width #1, 5 ' Устанавливаем длину вывода 5
For I = 0 To 9 ' Циклимся 10 раз
Print #1, Chr(48 + I); ' Выводим по 5 символов в строке
Next I
Close #1 ' Закрываем файл

```

With object

[statements]

End With

С помощью этого оператора можно произвести серию обращений к объекту или переменной типа запись, код становится короче и понятней.

Пример:

```

With MyObject
.Height = 100
' Аналогично выражению MyObject.Height = 100.
.Caption = "Hello World"
' Аналогично выражению MyObject.Caption = "Hello World".
With .Font
.Color = Red ' Аналогично выражению MyObject.Font.Color = Red.
.Bold = True ' Аналогично выражению MyObject.Font.Bold = True.
End With
End With

```

Write #filenumber, [outputlist] - последовательно записывает данные в файл.

Пример:

```

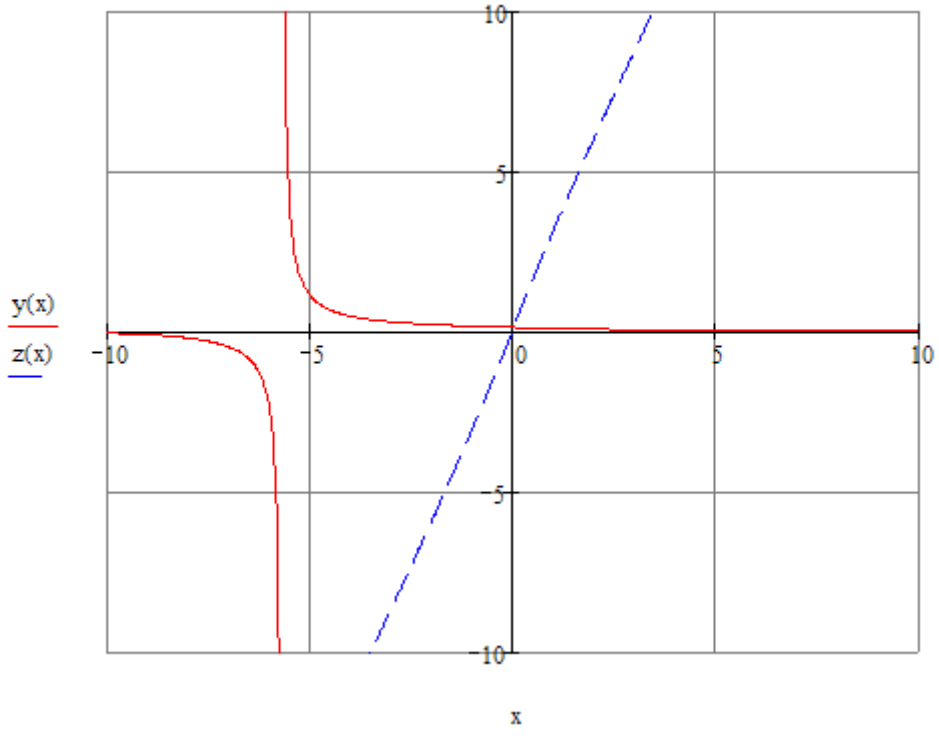
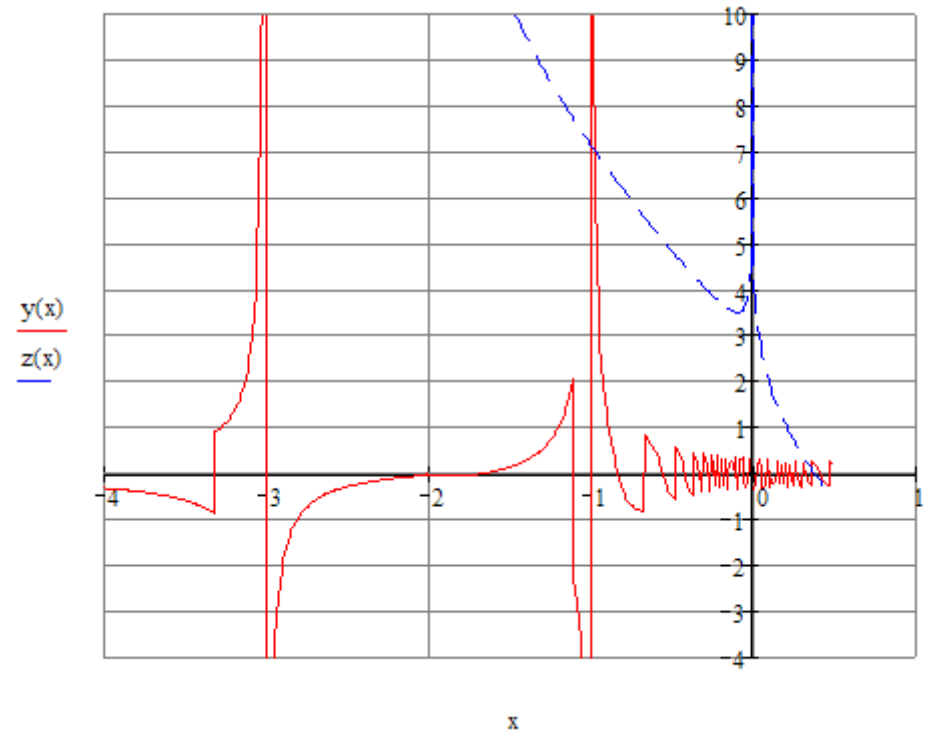
Open "TESTFILE" For Output As #1 ' Открываем файл для записи
Write #1, "Hello World", 234 ' Записываем строку, разделённую запятой
Write #1, ' Записываем пустую линию
Dim MyBool, MyDate, MyNull, MyError
' Присваиваем Boolean, Date, Null и Error значения
MyBool = False
MyDate = #February 12, 1969#
MyNull = Null
MyError = CVErr(32767)
' Boolean запишется как #TRUE# или #FALSE#. Дата запишется
' как #1969-01-12#. Null запишется как #NULL#.

```

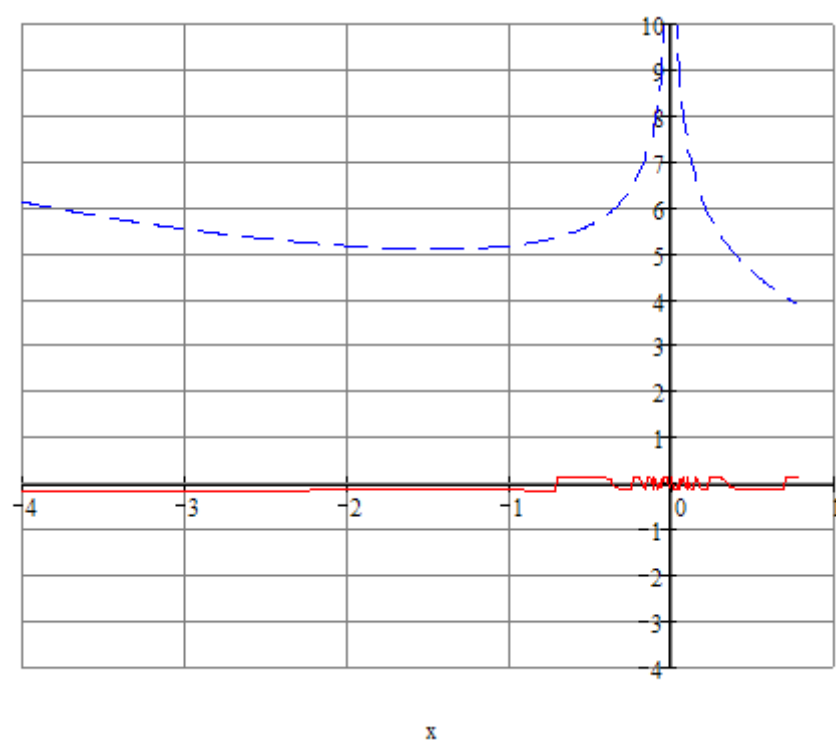


```
' Error копируется как #ERROR код_ошибки#.
Write #1, MyBool ; " это Boolean значение
Write #1, MyDate ; " это Дата"
Write #1, MyNull ; " это Null"
Write #1, MyError ; " а это Error"
Close #1 ' Закрываем файл
```

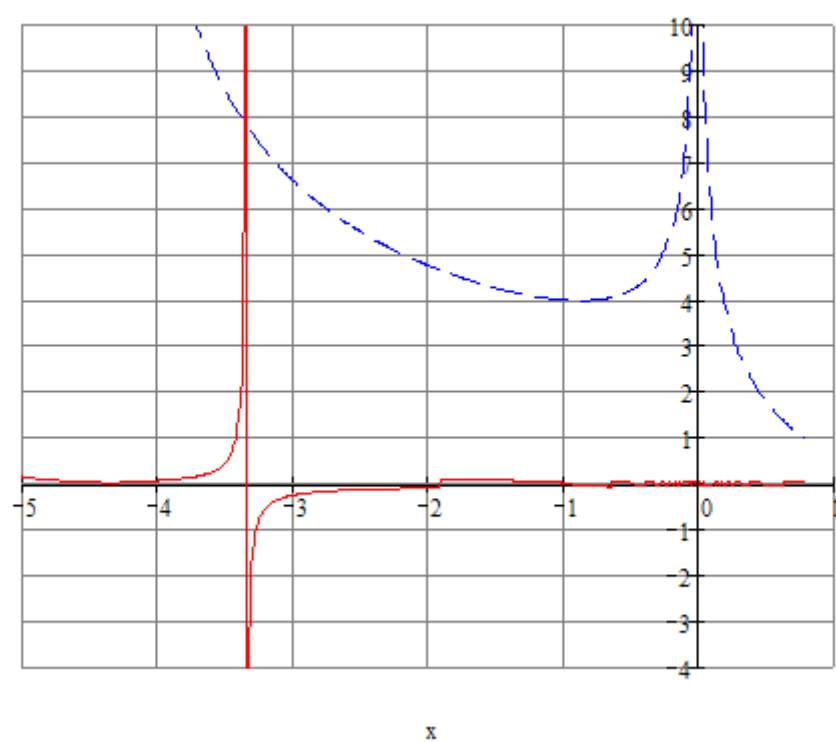
15 Приложение: Графики функций к самостоятельным заданиям п. 4.6

Вариант	График функций
1	 <p>Graph of functions $y(x)$ (red solid line) and $z(x)$ (blue dashed line) for Variant 1. The x-axis ranges from -10 to 10, and the y-axis ranges from -10 to 10. The red curve $y(x)$ has a vertical asymptote at $x = -5$ and a horizontal asymptote at $y = 0$. The blue dashed line $z(x)$ is a straight line passing through the origin with a slope of 1.</p>
2	 <p>Graph of functions $y(x)$ (red solid line) and $z(x)$ (blue dashed line) for Variant 2. The x-axis ranges from -4 to 1, and the y-axis ranges from -4 to 10. The red curve $y(x)$ has vertical asymptotes at $x = -3$ and $x = -1$, and a horizontal asymptote at $y = 0$. The blue dashed line $z(x)$ is a curve that decreases from a high value at $x = -3.5$ towards $y = 0$ as x approaches 0.</p>

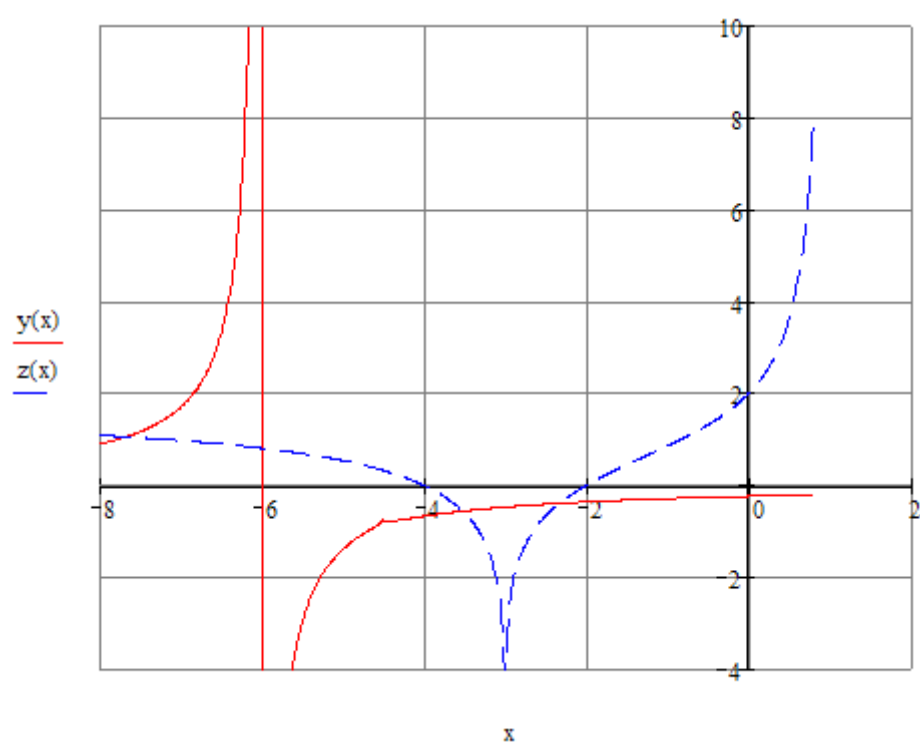
3

 $\frac{y(x)}{z(x)}$


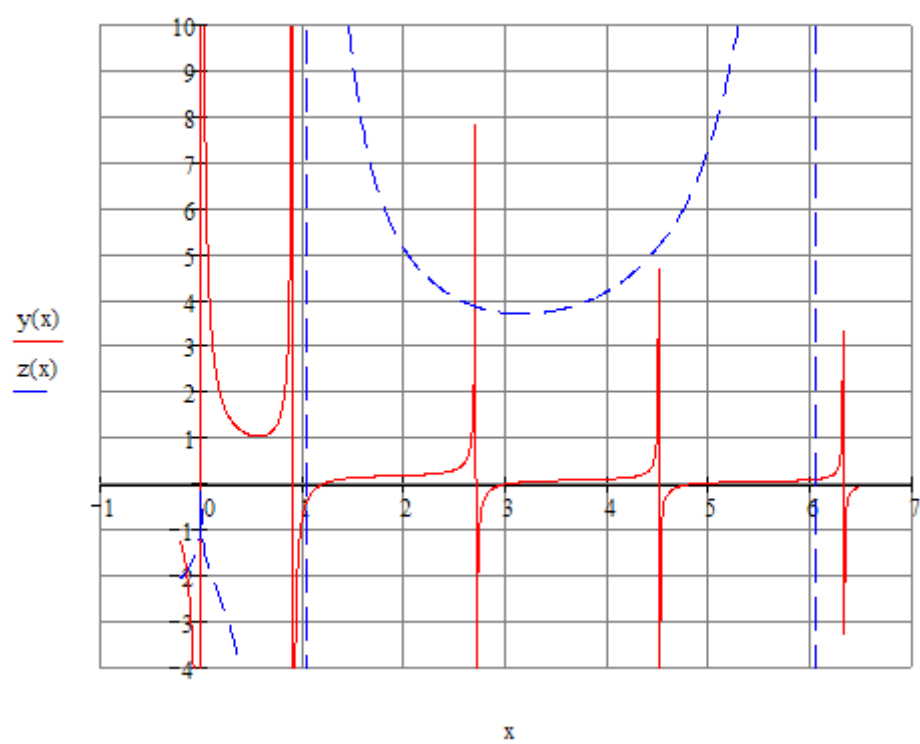
4

 $\frac{y(x)}{z(x)}$


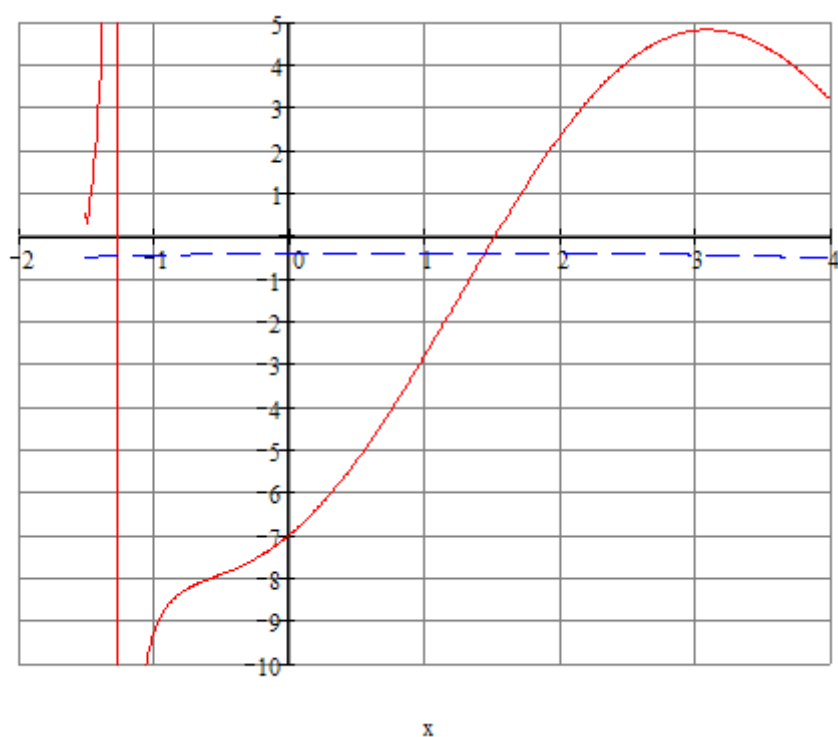
5



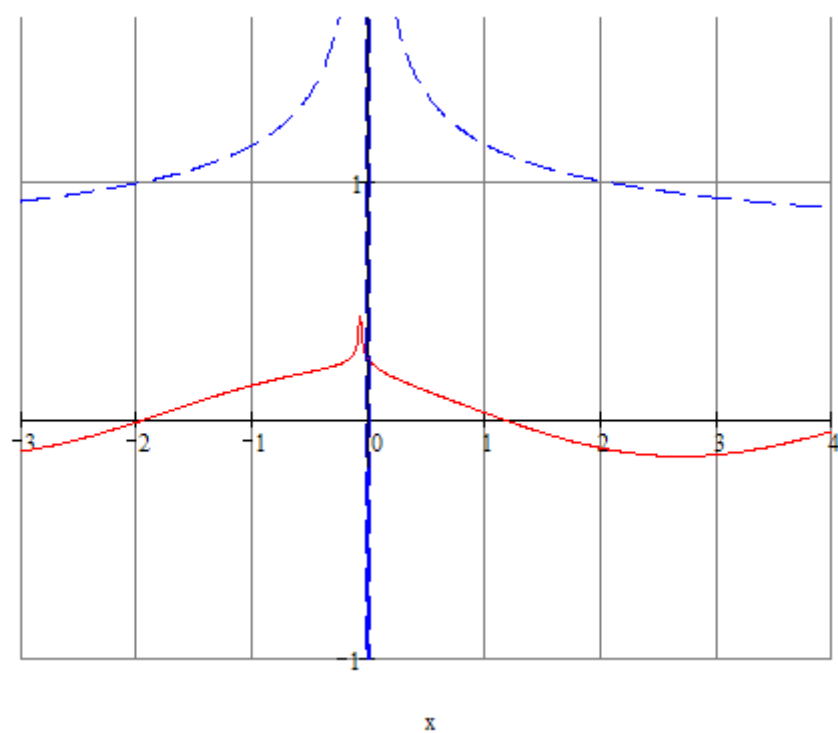
6



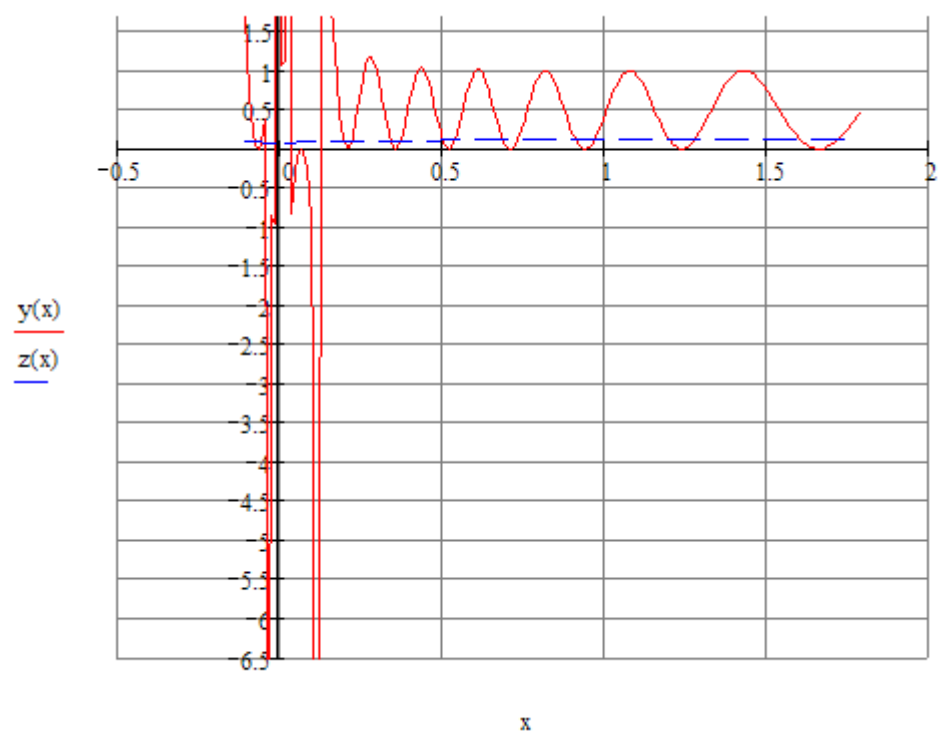
7

 $\frac{y(x)}{z(x)}$


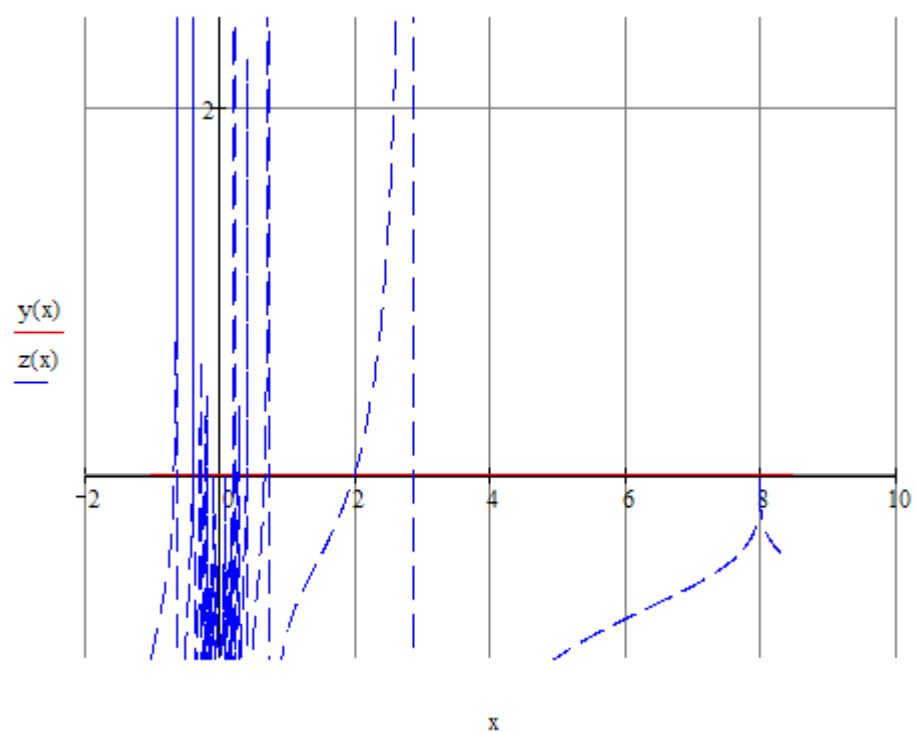
8

 $\frac{y(x)}{z(x)}$


9

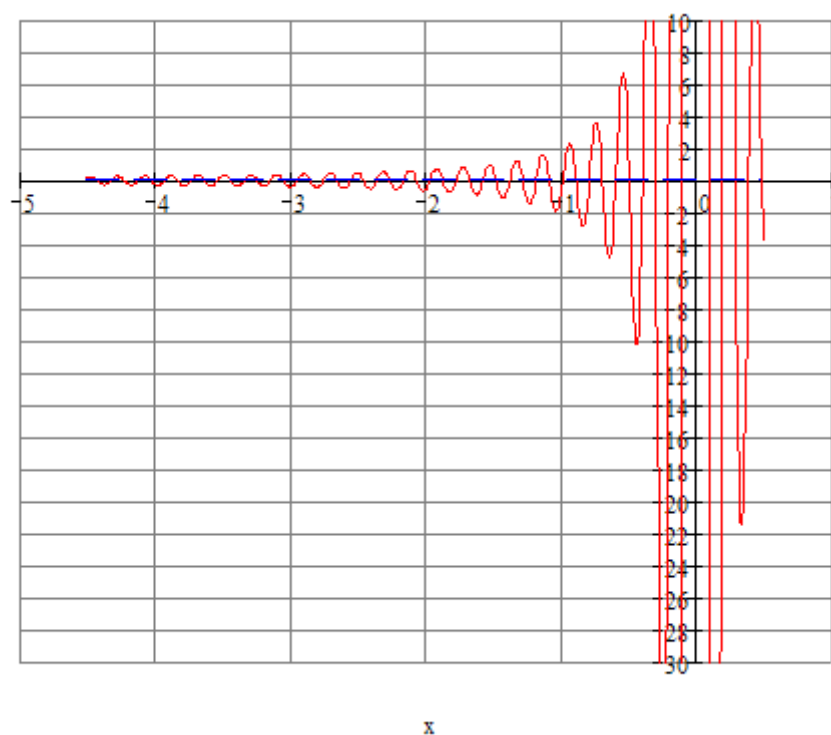


10



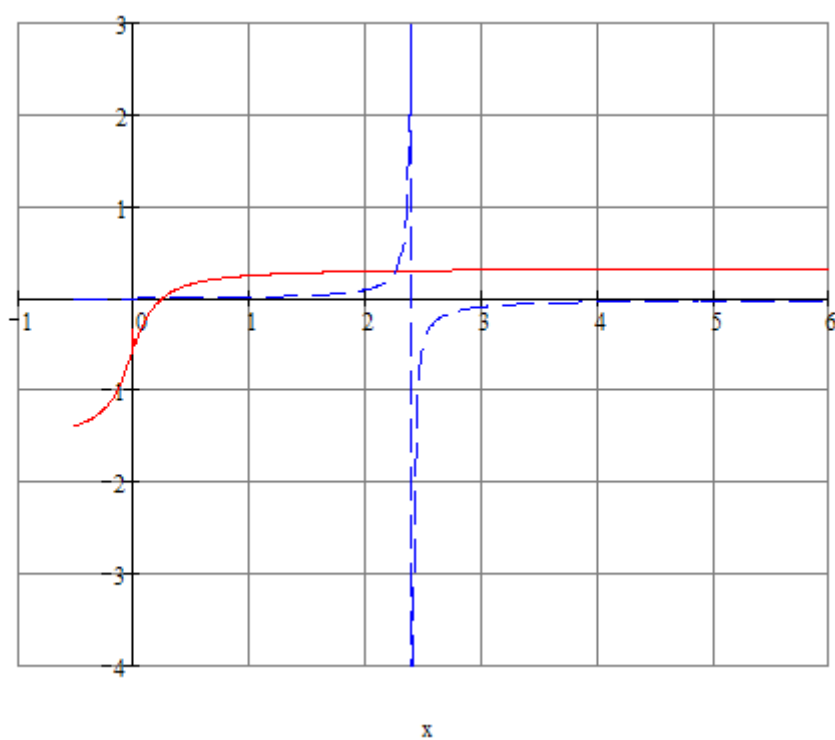
11

y(x)
z(x)

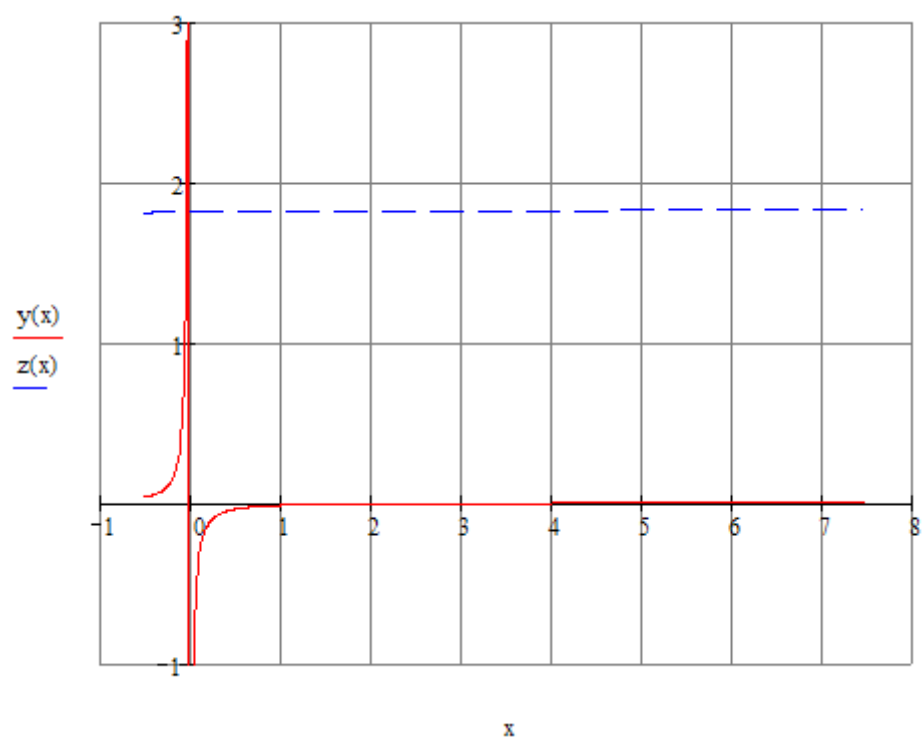


12

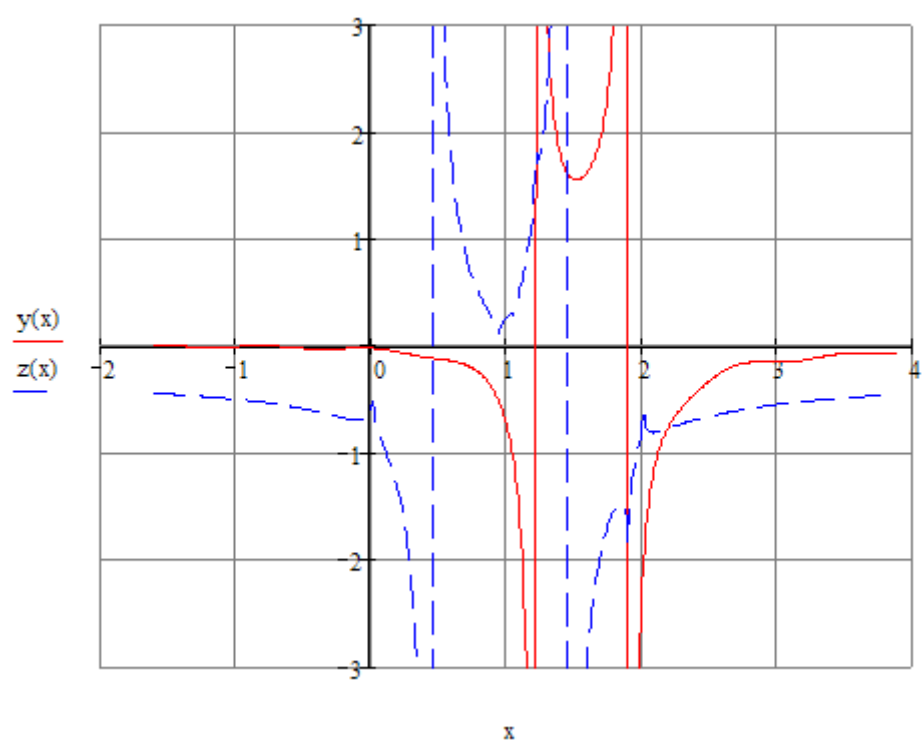
y(x)
z(x)



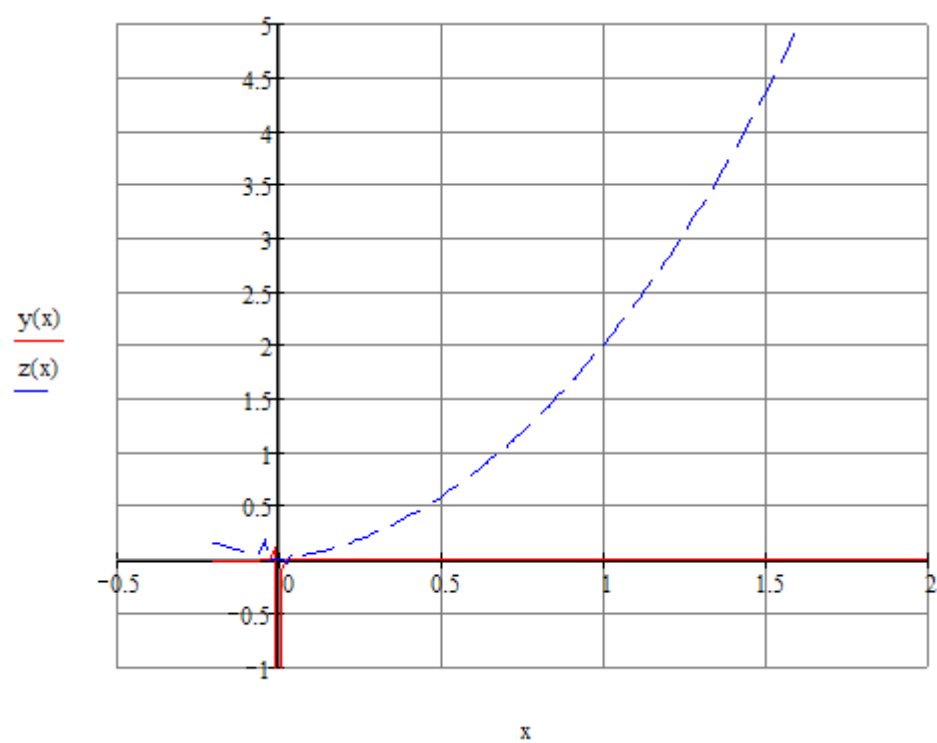
13



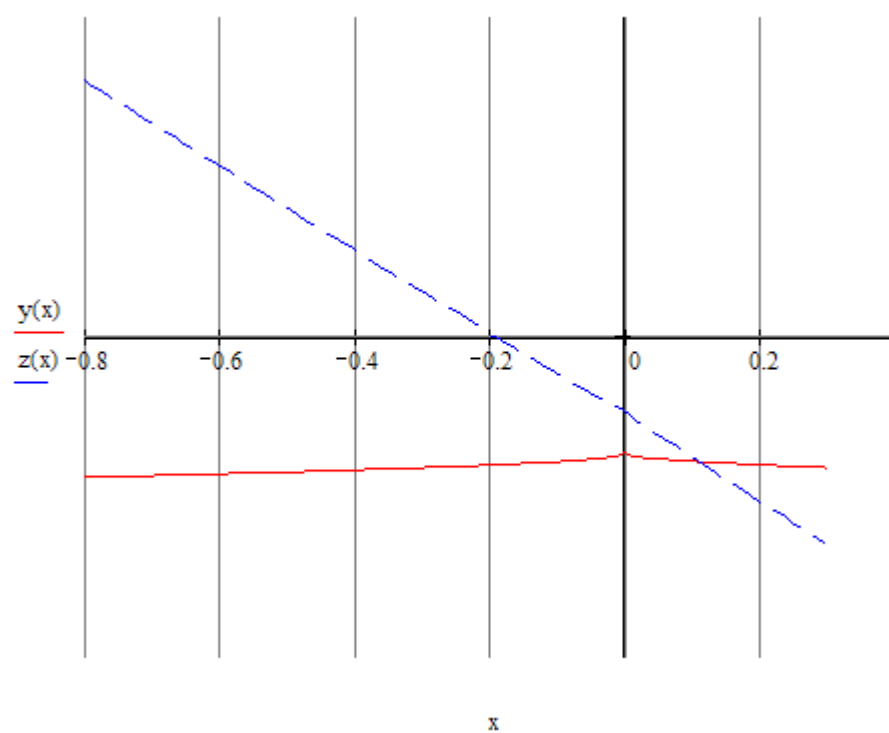
14



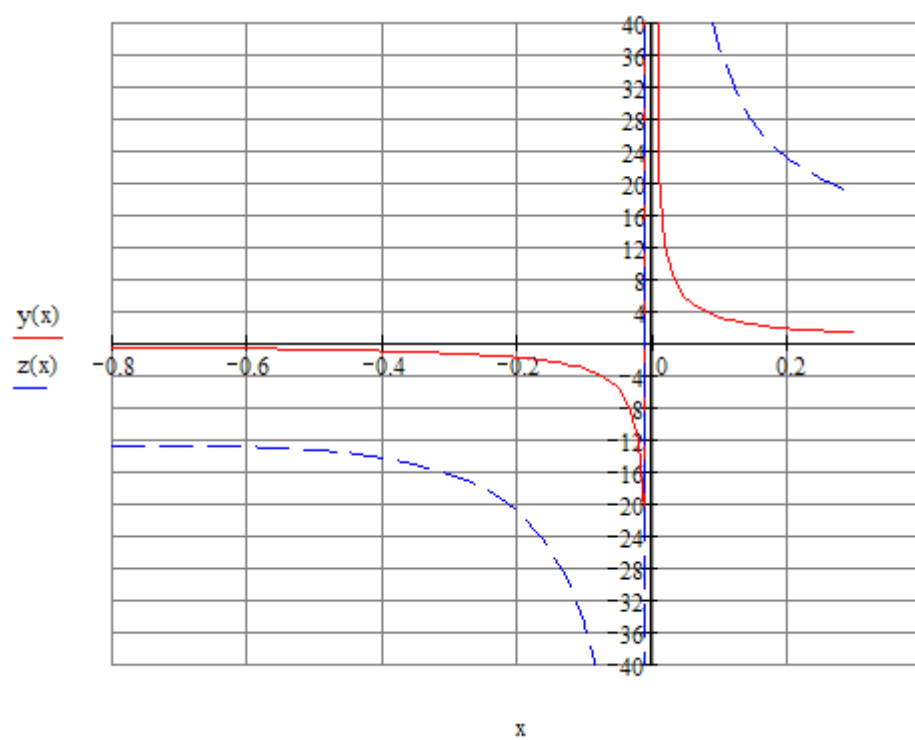
15



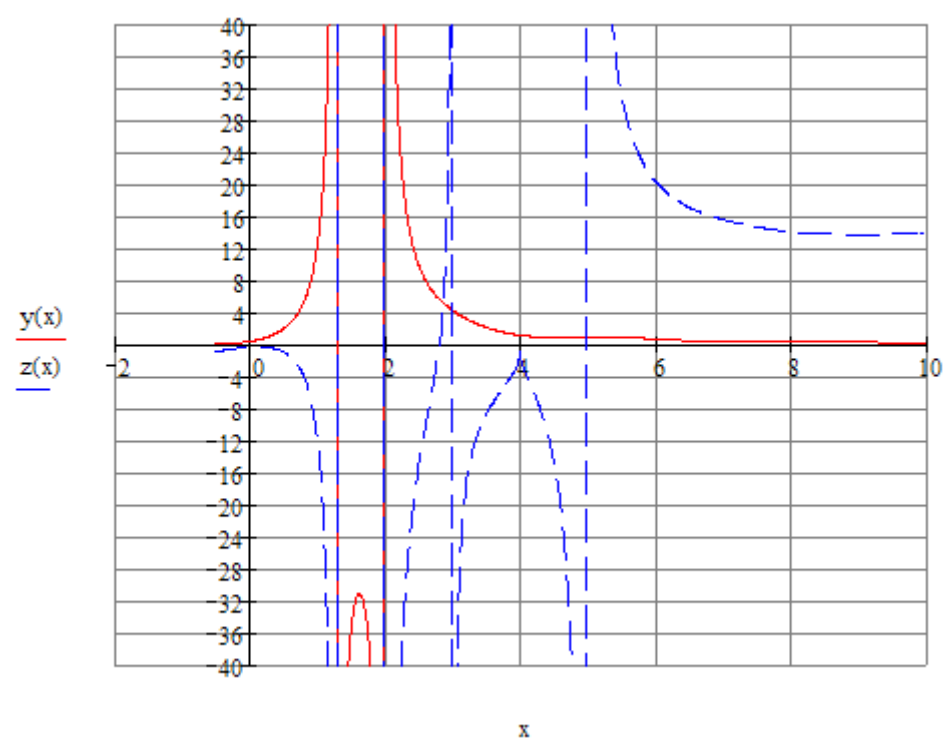
16



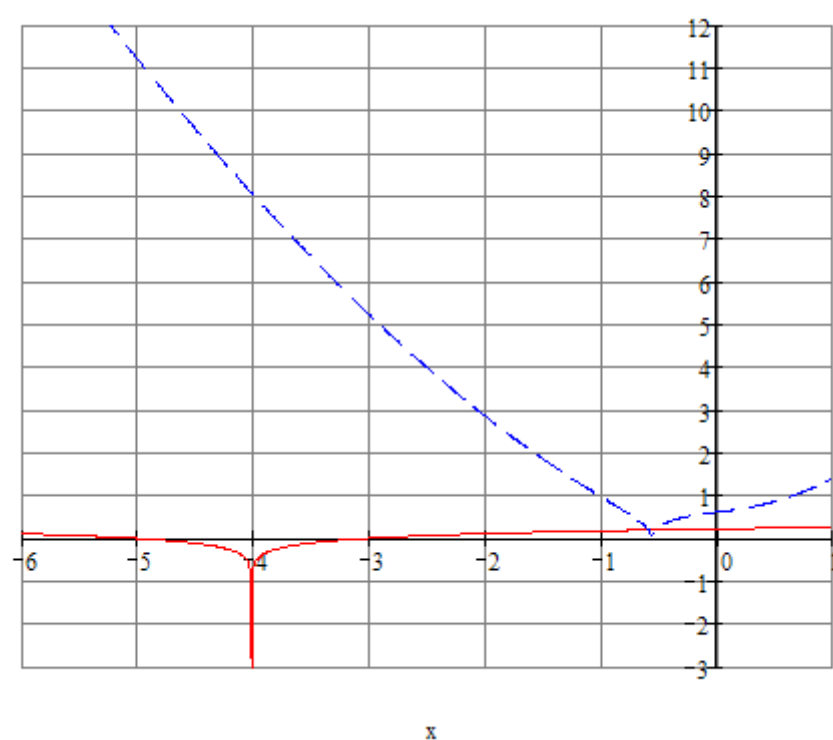
17



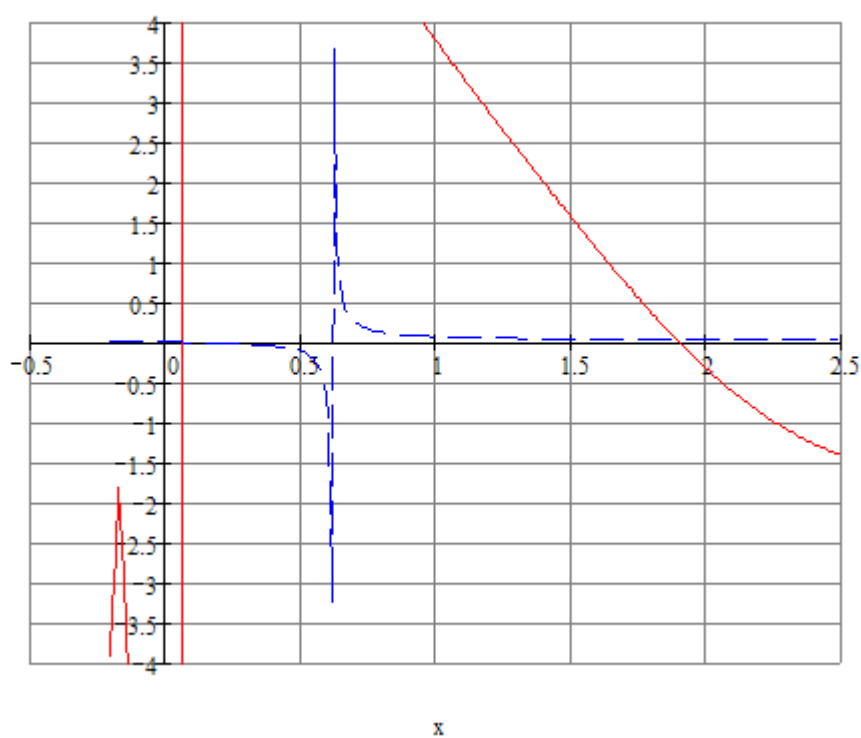
18



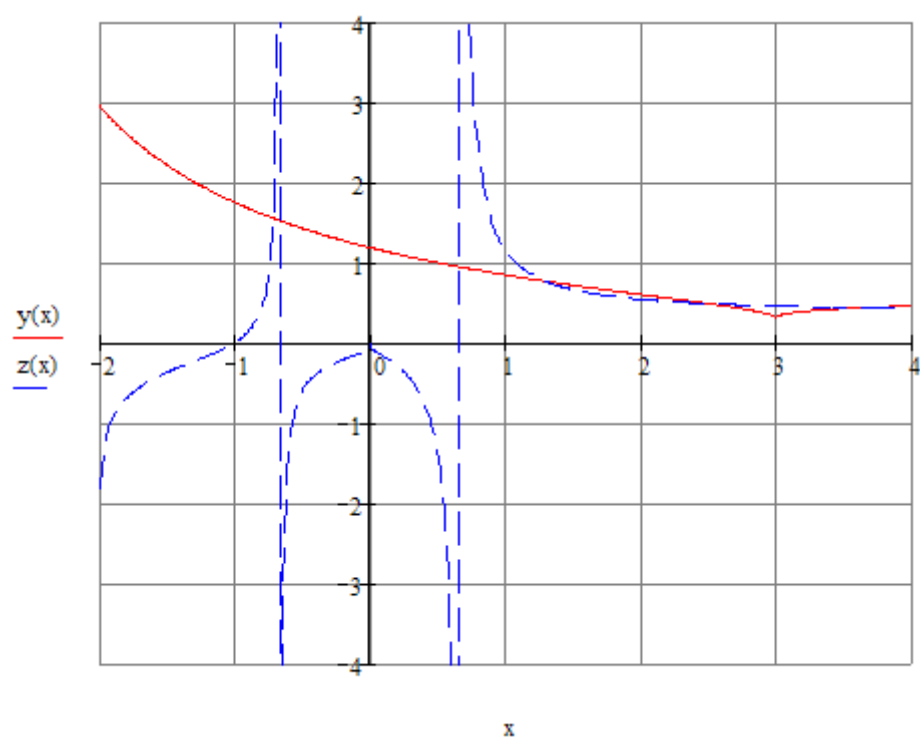
19

 $\frac{y(x)}{z(x)}$


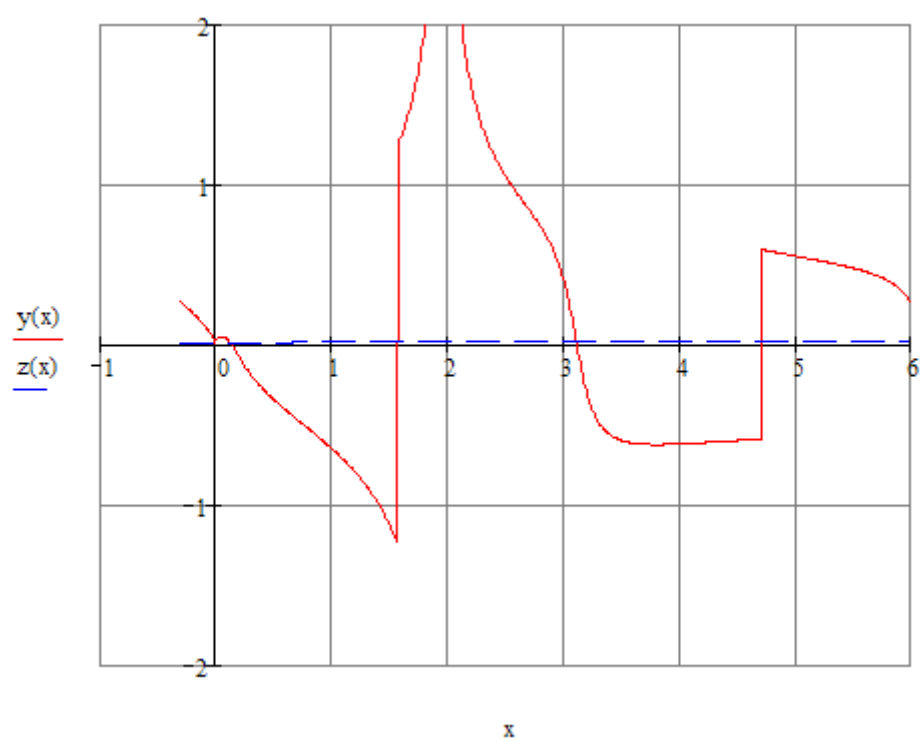
20

 $\frac{y(x)}{z(x)}$


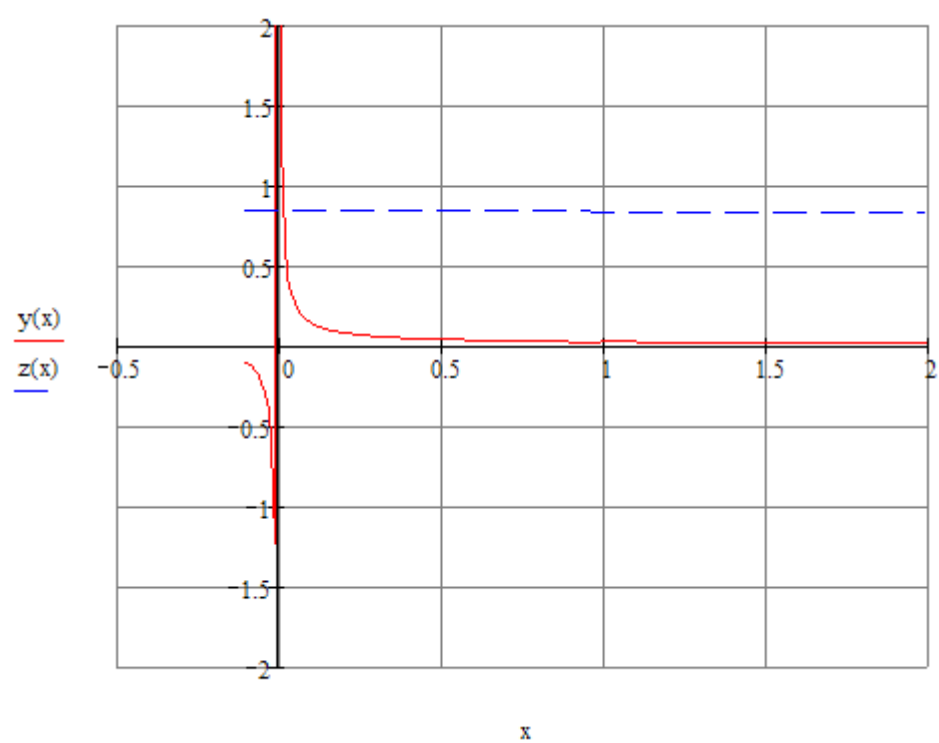
21



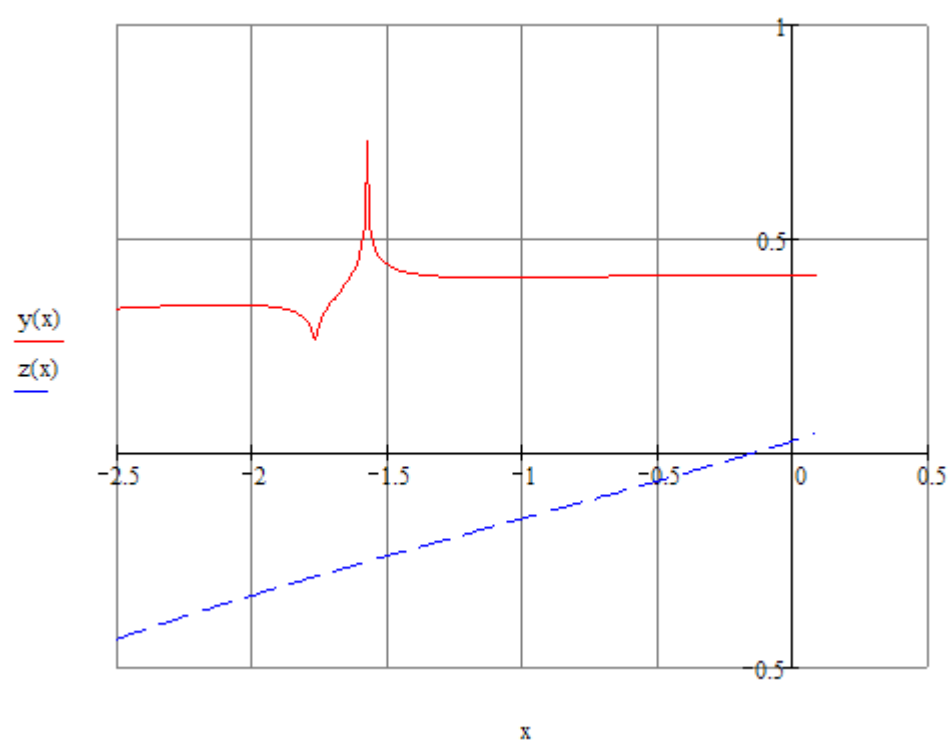
22



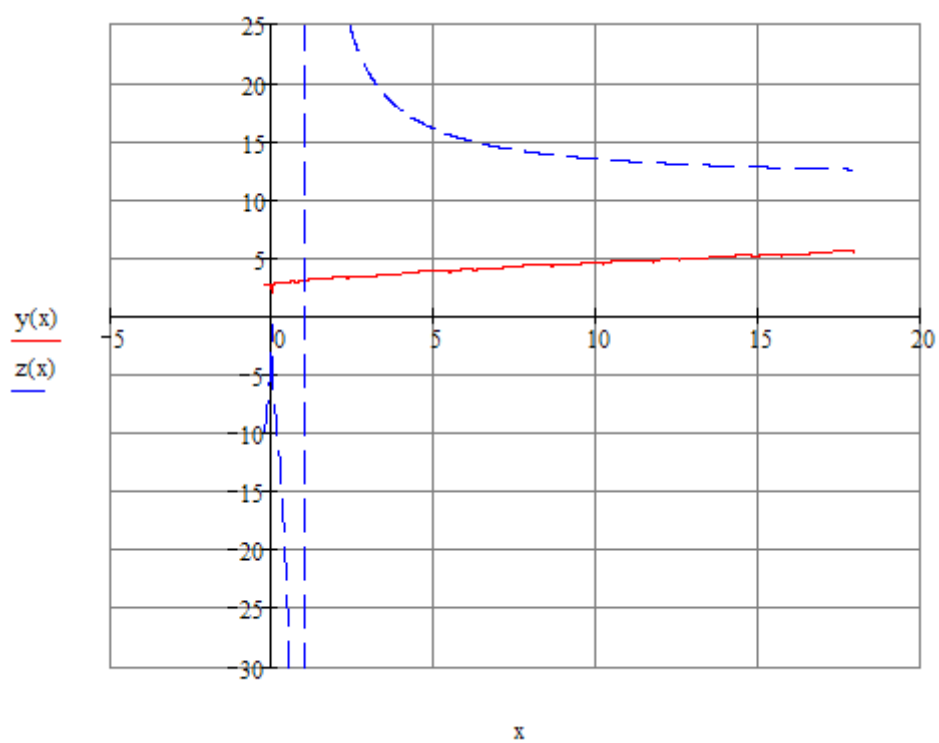
23



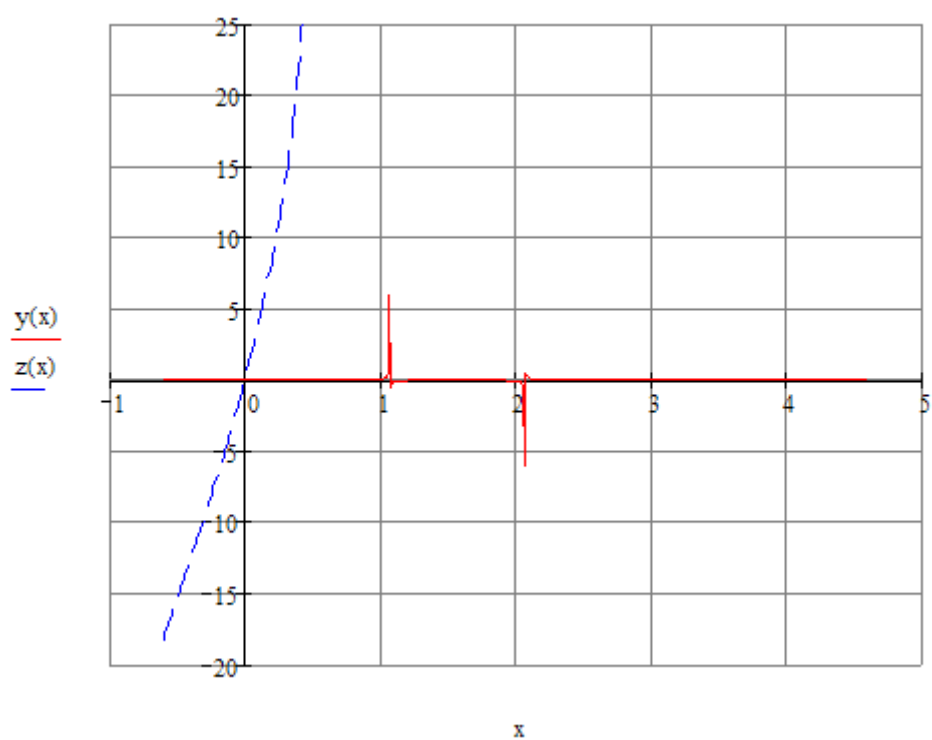
24



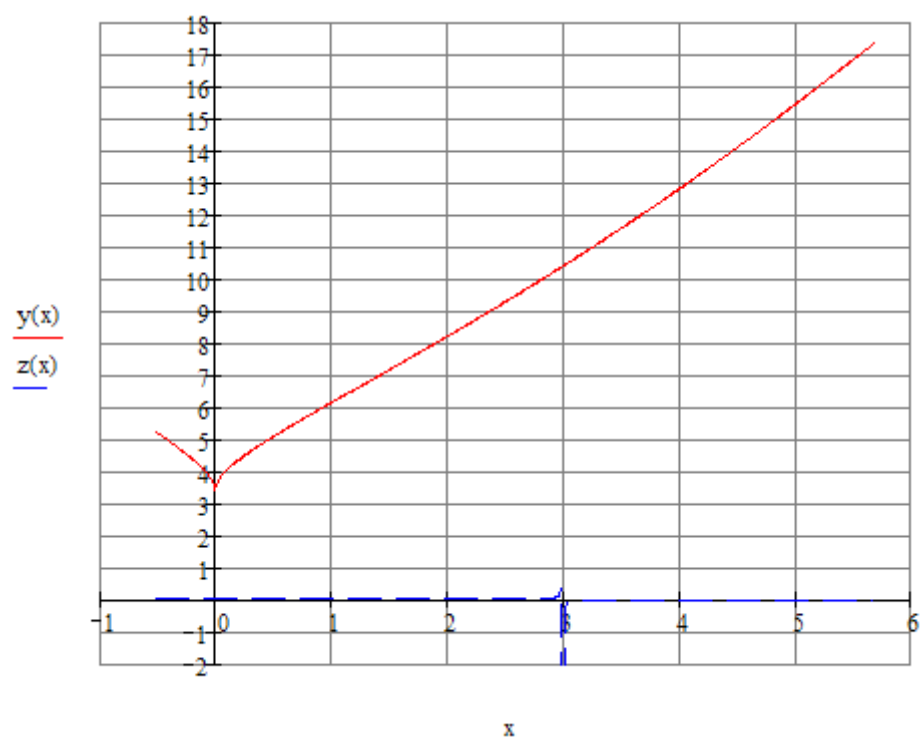
25



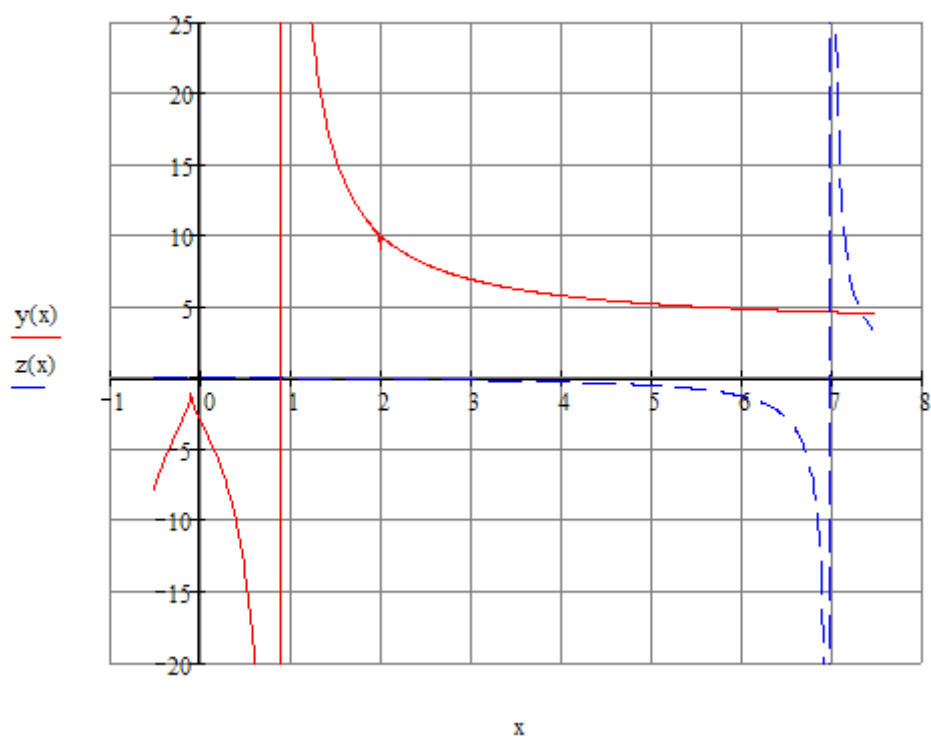
26



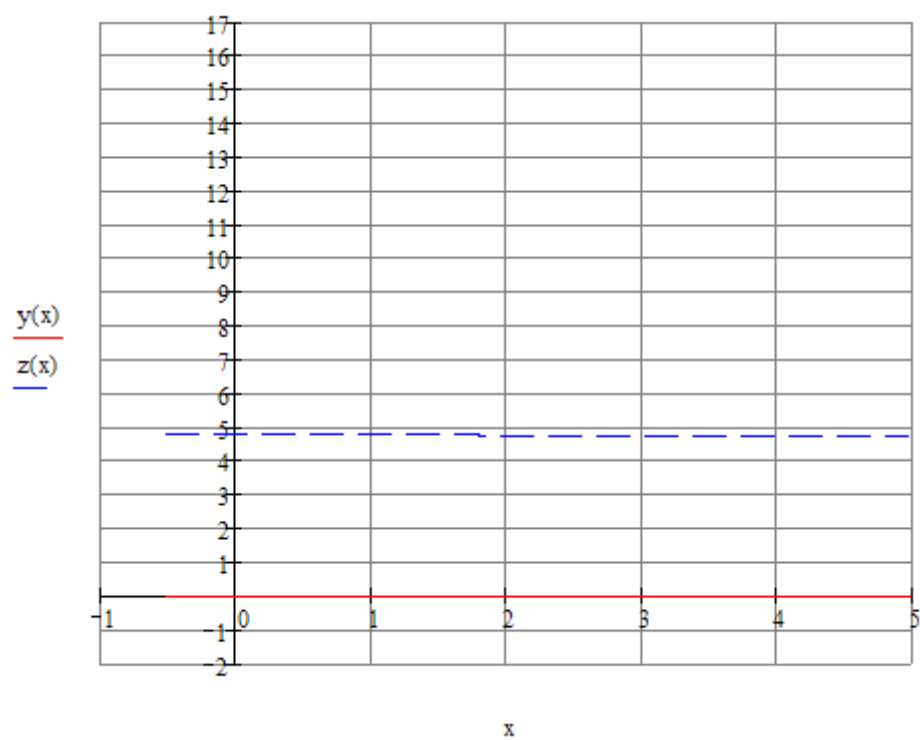
27



28



29



30

