МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет

Гурьева О.О., Гусева Л.П.

ЭЛЕМЕНТЫ ПРОГРАММИРОВАНИЯ НА VBA

Методические указания к лабораторным работам
по дисциплине «Информатика» для студентов специальности 1-27
01 01 «Экономика и организация производства»

Электронный учебный материал

Минск 2013

УДК 044.42 (075.8)
ББК 32.973-018я7
Э 45

А в т о ры :
О.О. Гурьева, Л.П. Гусева

Р е ц е н з е н т ы :

Валицкий С.В., доцент кафедры «Экономика и управление на предприятии» Минского института управления, к.т.н.;

Методические указания позволяют освоить основные приёмы процедурного программирования в среде Visual Basic for Applications (VBA). В нем содержатся основные сведения о VBA-программах, типы данных, описание переменных, ввод исходных данных, вывод результатов обработки исходной информации, статические и динамические массивы.

Белорусский национальный технический университет
пр-т Независимости, 65, г. Минск, Республика Беларусь
Тел.(017)292-77-52 факс (017)292-91-37
E-mail: dce@bntu.by
http://www.bntu.by/sf-es.html
Регистрационный № БНТУ/СФ71-35.2013

Содержание

Введение

VBA – процедурный язык программирования Visual Basic для приложений (for Applications), относится к языкам объектно-ориентированного программирования (ООП). VBA является стандартным макроязыком, который применяется для расширения функциональных возможностей приложения, в котором он используется. С помощью языка VBA можно создавать сложные программы, приложения, решать самостоятельно сложные задачи, не обращаясь к профессиональным программистам.

Программы в этом случае строятся из фундаментальных сущностей, называемых объектами. Каждый объект характеризуется набором свойств (property) и операций, которые он может выполнять (method). Работа программиста в рамках технологии ООП сводится к созданию объектов, описанию их свойств и реакций на внешние события. Реализация взаимодействия между объектами ложится на исполняющую среду средства разработки программ, в данном случае на VBA.

В настоящее время VBA встроен:

во все основные приложения Microsoft Office – Word, Excel, Access, Power Point, OutLook, Front Page, InfoPath;

в другие приложения Microsoft Offic, такие, кА Visio, Project;

в более 100 приложений третьих фирм таких, как CorelDRAW, CorelWordPrefect Office 2000, AutoCAD и т.п.

Для всех приложений инструментарий, синтаксис языка программирования и принципы работы VBA являются общими.

В VBA Excel имеется более 100 встроенных объектов. Примерами объектов здесь могут служить *рабочая книга* (Workbook), *рабочий лист* (Worksheet), *диапазон* (Range), *диаграмма* (Chart), *форма* (UserForm). Объекты имеют иерархическую структуру, т.е. одни объекты могут содержать ряд других. Количество и названия объектов для каждого приложения определены заранее. Всю совокупность объектов того или иного приложения можно просмотреть в справочной информации, вводя соответствующие ключевые слова, например Microsoft Word Objects, Microsoft Excel Objects, Microsoft Access Objects и т.д. *Семейство* (объект Collections) представляет собой объект, содержащий несколько других объектов одного и того же типа. Например, объект Workbooks (рабочие книги) содержит все открытые объекты Workbook (рабочая книга). Каждый элемент

семейства нумеруется и может быть идентифицирован или по номеру или по имени.

Важнейшим понятием ООП является класс. Каждый объект принадлежит некоторому классу. Класс определяет имя объекта, его свойства и действия, выполняемые над объектом. То есть, класс представляет собой шаблон, на основе которого во время выполнения программы создается объект. В свою очередь объект является экземпляром класса.

В программе на языке VBA необходимо идентифицировать объект прежде, чем применять к нему методы или изменять его свойства.

Тема 1. Макропрограммирование в приложениях Excel и Word

Приложения Word, Excel, Access пакета Microsoft Offce 2000 при создании сложных документов помимо стандартных средств позволяют автоматизировать выполнение некоторых операций с помощью макросов. Макрос представляет собой последовательность команд, сгруппированных вместе для выполнения какой-либо задачи. Макросы могут быть созданы как в режиме протоколирования, так и написаны на языке программирования Visual Basic for Application (VBA).

Макросы в приложениях Word, Excel создаются с помощью инструмента MacroRecorder — транслятора, создающего программу (макрос) на языке VBA. Макрос является результатом перевода на язык VBA действий пользователя с момента запуска MacroRecorder до окончания записи макроса. Для активизации MacroRecorder надо выбрать команду **Сервис, Макрос, Начать запись**. Появится диалоговое окно (ДО) Запись макроса.

В приложениях Word, Excel диалоговые окна Запись макроса как видно из рисунка 1 немного отличаются.

Поля Имя макроса и Описание предназначены для задания имени макроса и его описания. По умолчанию макросам присваиваются имена Макрос1, Макрос2 и т.д. Чтобы легче было распознавать макросы, лучше присваивать им уникальные имена, поясняющие их назначение.

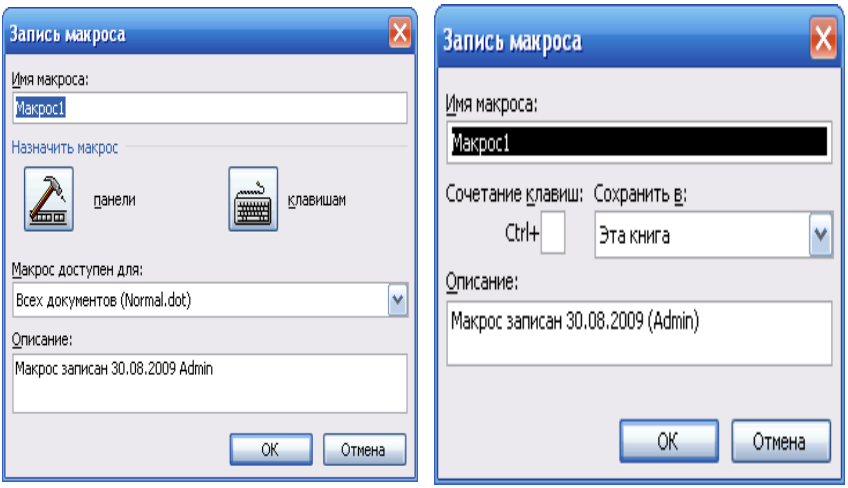


Рис.1 Окна **Запись Макроса** в текстовом редакторе и в электронной таблице

Заполнение поля Описание важно при наличии в пользовательском приложении нескольких многократно используемых макросов, чтобы по прошествии времени можно было определить, для какой цели создавался тот или иной макрос. Поле Сочетание клавиш позволяет назначить клавишу, которая в сочетании с клавишей Ctrl служит для запуска макроса на выполнение. Это поле используется только для постоянно используемых макросов. Раскрывающийся список Сохранить в: предназначен для выбора книги, в которой будет сохранен макрос. Если выбрать из списка Эта книга (по умолчанию), то макрос сохранится на новом листе модуля в активной рабочей книге. Если выбрать Новая книга, то макрос сохранится в новой рабочей книге. Если выбрать Личная книга макросов, то макрос сохраняется в специальной скрытой книге (Personal.xls). Записанные в ней макросы доступны для других рабочих книг. По команде Окно, Отобразить можно отобразить личную книгу макросов. Для просмотра текста макроса выполняется команда **Сервис, Макрос, Макросы**, выделить имя макроса, нажать кнопку **Изменить в диалоговом окне Макрос**. Это вызовет появление главного окна редактора VBA — VBE (Visual Basic Editor), представленного на рис.2.

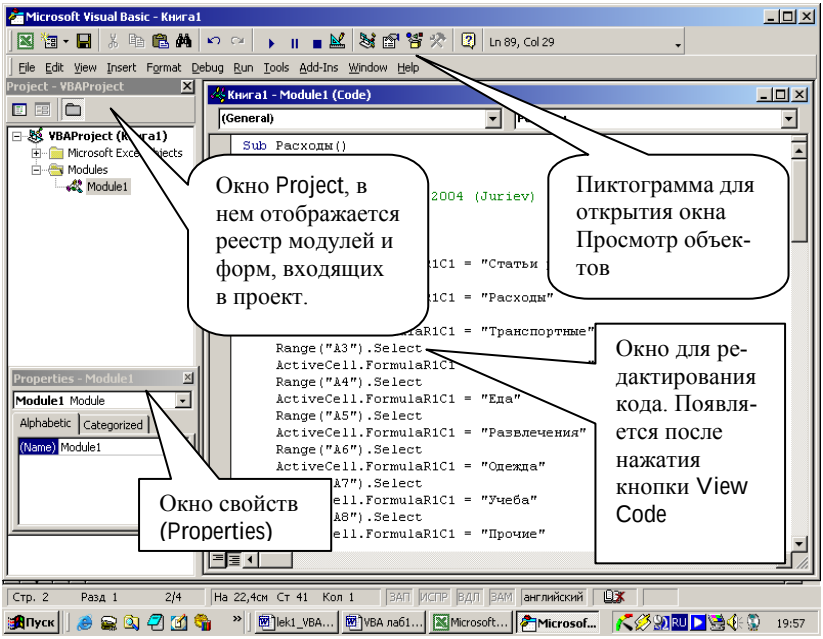


Рис 2. Структура окна VBE (Visual Basic Editor)

Задание 1. В приложении Excel создать макрос с именем Расходы. Он должен строить таблицу учета ежемесячных расходов студента, форматировать ее и по данным таблицы строить диаграмму. Таблица должна иметь 9 строк и 2 столбца.

Заголовок таблицы (Статьи расходов, Расходы) залить бледно–зеленым цветом, заключить в рамку, задать размер – 12, начертание – полужирный. Статьи расходов: транспорт, коммунальные платежи, еда, одежда, развлечения, учеба, прочие. Величину расходов — не задавать. Диапазон статей расходов залить бледно–зеленым цветом, диапазон величин расходов заключить в рамку, итоги таблицы (Итого, подбитая сумма расходов) залить желтым цветом, изменить ширину первого столбца так, чтобы в него полностью помещался набранный текст.

Построить диаграмму ежемесячных расходов студента. Диаграмму поместить справа от таблицы, увеличить ее размер, удалить легенду, задать размер шрифта надписей осей — 11, начертание — полужирный.

После создания макроса заполнить ячейки таблицы исходными данными.

Задание 2. Просмотреть текст макроса Расходы. Определить команды, относящиеся к построению таблицы, к построению диаграммы. Найти команду, изменяющую ширину столбца A, определить команду, в которой расчитывается сумма расходов. Определить имя рабочего листа, на котором с помощью макроса Расходы можно построить шаблон таблицы с диаграммой.

Задание 3. Выполнить макрос Расходы на новом рабочем листе, задав имя рабочего листа, определенное в предыдущем задании. После выполнения мвкроса переименовать этот рабочий лист, задав имя текущего месяца.

Задание 4. Построить макрос в приложении Excel. В выделенном диапазоне ячеек заменить отрицательные числа на 0. Назначить макросу комбинацию клавиш <Ctrl+O>. Просмотреть текст макроса.

Задание 5. Построить макрос в приложении Word.

Задать для второго абзаца активного документа следующие элементы форматирования:

тип шрифта Arial, Courier New, Tahoma, Garamond;

размер 14, 16, 11, 15, 13, 17;

цвет синий, зеленый, сине–зеленый, темно–красный, сизый;

курсив;

междустрочный интервал двойной.

Назначить макросу комбинацию клавиш <Ctrl+F>. Просмотреть текст макроса. Определить команды, выполняющие заданное форматирование.

Задание 6. Построить макрос в приложении Word.

В каждом абзаце активного документа первую букву в первом слове выделить полужирным курсивом оранжевого цвета. Назначить маколсу кнопку на панели инструментов. Просмотреть текст макроса. Определить команды, выполняющие заданное форматирование.

Тема 2. Среда программирования VBA. Структура окна редактора VBA

Чтобы войти в редактор VBA необходимо загрузить одно из приложений MS Office, в нашем случае — Excel. Затем выпол-

нить команду **Сервис/Макрос/Редактор** Visual Basic или нажать клавиши Alt-F11 или выполнить команду **Вид/Панели инструментов**/Visual Basic/**кнопка Редактор** Visual Basic.

Возвратиться в рабочую книгу Excel из редактора VBA можно нажатием кнопки Вид Microsoft Excel (View Microsoft Excel)  (рис. 3).

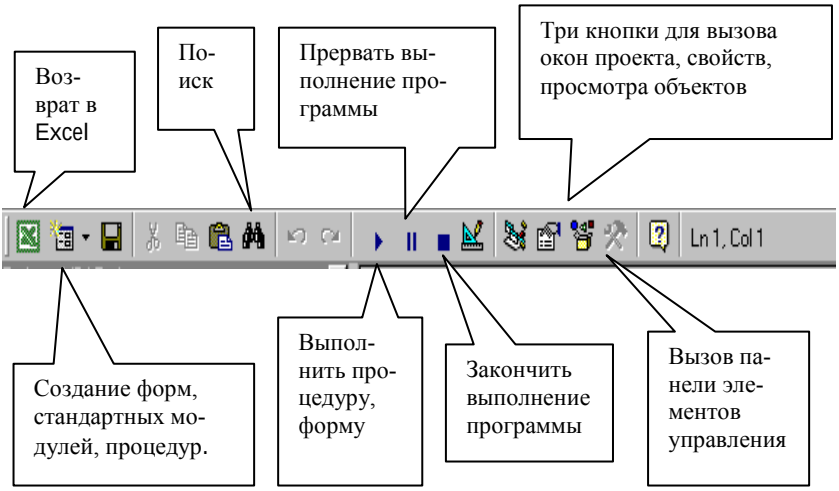


Рис. 3. Панель инструментов редактора VBA.

Интерфейс VBA состоит из следующих основных компонентов: окно проекта, окно свойств, окно редактирования кода, окна форм, меню, панели инструментов. Основные компоненты представлены на рис. 2.

Окно проекта Project Explorer

На рис.4 представлено окно проводника проекта Project Explorer. В нем представлено дерево компонентов открытого приложения. На рисунке видно, что в окне проекта отображается иерархическая структура рабочих листов, файлов форм и модулей, входящих в создаваемый проект. Это окно обычно открыто. Но если надо, то можно его открыть, используя клавиши Ctrl-R или команду меню View/Project Explorer. Самый верхний уровень – это Project. Ему соответствует рабочая книга приложения Excel, документа прило-

жения Word, презентация приложения Power Point и т.д. Если редактор VBE открыт из Excel, то в окне Project Explorer будут представлены все открытые книги этого приложения и специальная скрытая книга Personal.xls.

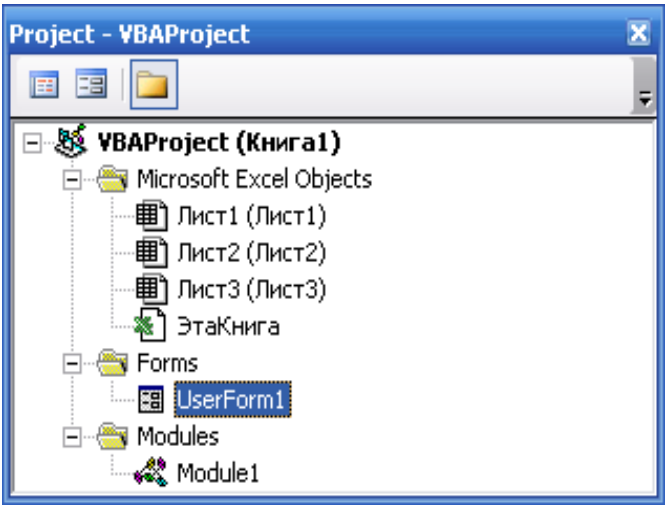


Рис 4. Окно проводника проекта (Project Explorer)

Каждый проект (Project) является одновременно и контейнером для хранения стандартных модулей, модулей классов и пользовательских форм. Добавить каждый из этих компонентов можно из меню Insert или через контекстное меню в Project Explorer.

Модуль — это текстовый файл, в котором набирается код программы. В одном модуле может храниться необходимое пользователю число программ. В проекте автоматически создается модуль для каждого рабочего листа, для всей книги, для каждой пользовательской формы, для макросов. Модули по своему предназначению делятся на модули объектов и стандартные. К модулям объектов относятся модули, связанные с рабочей книгой, рабочими листами, формами и модули класса. К стандартным модулям, относятся модули, содержащие код макросов или код программ, носящих общий характер. Стандартные модули добавляются в проект командой Insert/Module. Удаление модуля из окна проекта производится выбором значка модуля и выполнением команды File/Remove Module.

Окно для редактирования кода Code

Чтобы открыть окно редактирования кода, надо указатель мыши переместить на значок файла в окне проекта и выполнить двойной щелчок кнопкой мыши. Окно служит для ввода и редактирования кода процедур приложения (рис. 5).

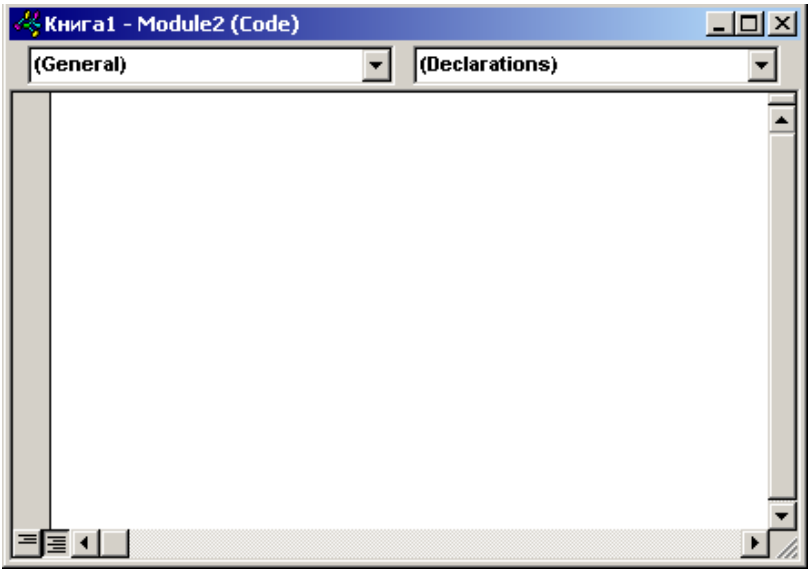


Рис. 5. Окно редактирования кода (Code)

Код внутри модуля организован в виде отдельных разделов для каждого объекта, программируемого в модуле. Доступны два режима представления кода: просмотр отдельной процедуры и всего модуля. Переключение режимов осуществляется выбором одной из двух кнопок в нижнем левом углу окна редактирования кода (Procedure View, Full Module View)  .

Редактор программного кода это обычный текстовый редактор и в нем можно вырезать и вставлять код, перетаскивать фрагменты кода, скопировать, перетаскивая фрагменты кода при нажатой клавише Ctrl.

Список General представляет собой список объектов. Из него выбирается объект, к которому будет относиться создаваемый программный код.

Список Declaration представляет собой список процедур и событий, относящихся к выбранному в списке General объекту. При выборе необходимого события автоматически будет создана процедура в окне Code. Эта процедура предназначена для обработки этого события. В процедуру нужно будет написать соответствующий код.

В редакторе кода встроено множество средств, облегчающих жизнь разработчику. Самые важные из них:

если редактор распознает ключевое слово, то первую букву этого слова делает заглавной и все слово выделяет синим цветом;

если надо закомментировать несколько строк сразу, то надо обратиться к панели инструментов Edit и воспользоваться кнопками CommentBlock и UncommentBlock;

редактор постоянно выдает предупреждение об ошибке. Отменить протесты редактора можно, сняв флажок Auto SyntaxCheck в диалоговом окне Options, но неверные строки все равно будут выделяться красным цветом.

Окно свойств

В окне свойств (рис.6) перечисляются основные установки свойств выбранной формы или элемента управления. Используя это окно, можно просматривать свойства и изменять их установки. Для просмотра свойств выбранного объекта надо щелкнуть кнопку Окно свойств (Properties Window) (рис.7) или выполнить команду Вид, Окно свойств (View, Properties Window).

Окно просмотра объектов

В этом окне (рис.8) приводится список всех объектов, которые имеются в системе и которые можно использовать при создании проекта.

Окно Просмотр объектов (Objects Browser) можно вызвать командой Вид, Просмотр объектов (View, Objects Browser) или нажатием кнопки Просмотр объектов (View, Objects Browser) (рис. 2). Окно Просмотр объектов состоит из трех основных частей: раскрывающегося списка Проект/Библиотека, списка Классы, списка Компоненты.

Раскрывающийся список Проект/Библиотека (Project/Library) находится в левом верхнем углу окна. В этом списке можно вы-

брать различные проекты и библиотеки объектов. Выбор <All Libraries> отображает список объектов всех библиотек.

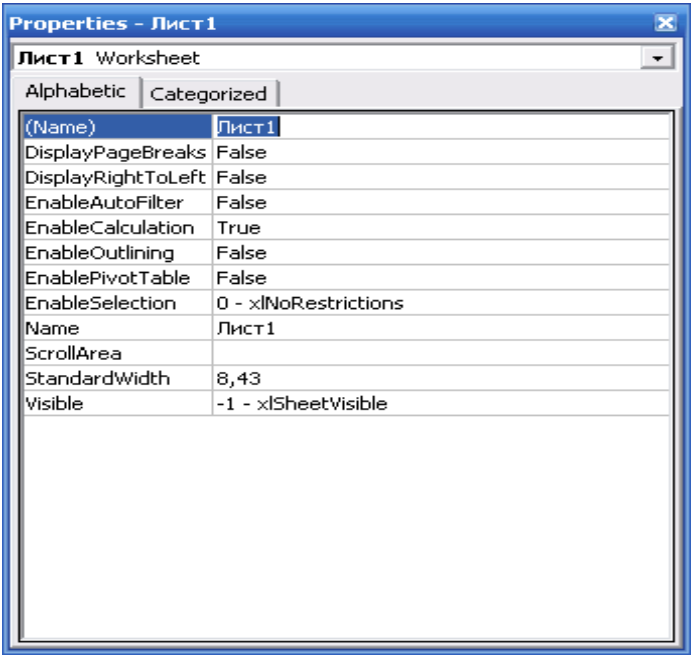


Рис. 6. Окно свойств (Properties Window)

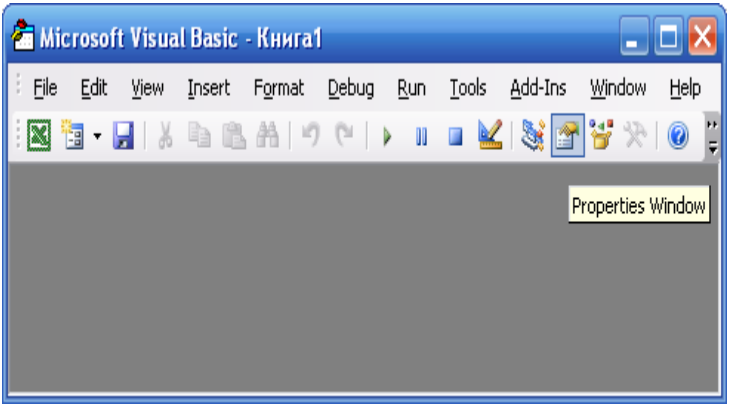


Рис. 7. Кнопка Properties Window в меню редактора VBE.

Список Классы (Classes). Все классы объектов выбранной библиотеки выводятся в списке Классы (Classes).

Список Компоненты (Members). После выбора класса из списка Классы (Classes), например Border, все компоненты выбранного класса выводятся в списке Компоненты (Members). При выделении строки в этом списке в нижней части окна Просмотр объектов (Objects Browser) приводится дополнительная информация о выбранной компоненте.

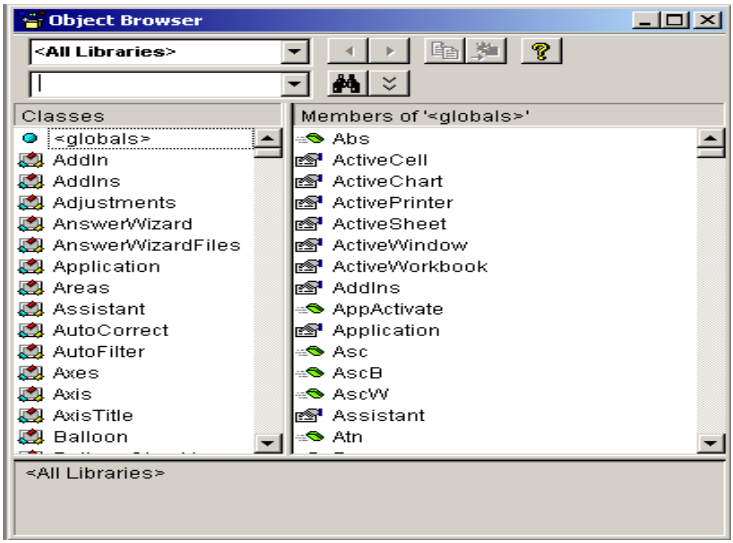


Рис. 8. Окно Просмотр объектов (Objects Browser)

Задание 1. Открыть текст макроса Расходы. Изучить структуру окна редактора VBA при просмотре текста созданного макроса.

Контрольные вопросы.

Что такое “свойство” и “методы”?

В чем состоит суть технологии программирования на языке VBA?

Привести примеры объектов в Word и в Excel.

Что такое “семейство”? Привести примеры в Word и в Excel.

Что такое “модуль”? Какие типы модулей рассматриваются в VBA?

Что такое “макрос”? Какие средства используются для создания макроса?

Назначение и структура окна проекта.

Назначение и структура окна редактирования кода.

Назначение и особенности окна просмотра объектов.

Структура панели инструментов редактора VBE.

Назначение окна свойств.

Как сохранить, запустить, отладить, редактировать макрос?

Тема 3. Создание простейших пользовательских функций

Программа — это законченная последовательность команд (инструкций) языка программирования, описывающая алгоритм решения поставленной задачи. Программы на языке VBA создаются в виде процедур в окне редактора VBE (Visual Basic Editor). Процедура представляет собой последовательность совместно выполняемых инструкций, имеющая имя. Инструкция — это синтаксически завершенная конструкция, представляющая отдельное действие, описание или определение. Инструкция VBA является полной командой и может содержать:

ключевые слова - слова или символы, распознаваемые как элемент VBA;

операторы - знаки операции в выражениях;

переменные - именованная область памяти, отведенная для временного хранения данных, которые могут изменяться во время выполнения программы;

константы - именованный элемент, сохраняющий постоянное неизменное значение в процессе выполнения программы;

выражения - комбинация ключевых слов, операторов, переменных и констант, результатом которой является строка, число или объект. Выражения можно использовать для выполнения вычислений, обработки символов или проверки данных.

Инструкции могут быть следующих типов:

инструкции описания - именуют переменные, константы или процедуры, а также могут задавать типы данных;

инструкции присвоения - присваивают значение или выражение переменной, константе или свойству объекта.

исполняемые инструкции - инициируют действие. Они могут выполнить метод Excel или функцию, а также могут организовать

повторение или ветвление блоков программы. Эти инструкции часто содержат математические или условные операторы.

Для пояснения определенной инструкции в коде программы используются комментарии. При выполнении процедуры комментарии игнорируются. Их можно вносить в любое место процедуры, начиная с апострофа (') или со слова Rem, за которым следует пробел.

Совокупность созданных пользователем и хранящихся совместно процедур составляет модуль. В документе или рабочей книге допускается любое количество модулей. VBA позволяет создавать такие типы процедур, как процедуры-подпрограммы, процедуры-функции.

Процедура-функция - это набор команд, которые должны быть выполнены. Принципиальное отличие функции одно: функция возвращает результирующее значение вызвавшей ее программе. В программе это значение будет затем использовано. Функция всегда возвращает единственное значение. Ее можно запустить из другой VBA-процедуры или использовать в формулах на рабочих листах. Синтаксис записи процедур-функций имеет вид:

```
Function имя([аргументы])
Инструкции языка VBA
имя=выражение
End Function
```

} Тело функции

При создании процедуры-функции после написания ключевого слова Function, имени функции, ее аргументов редактор автоматически дописывает оператор End Function.

Имя (идентификатор) процедуры-функции должно отражать суть функции. Это распространяется и на имена переменных, констант, процедур и других объектов. В VBA имеются следующие ограничения на имена:

длина имени должна быть не более 255 символов;

имя не должно содержать точку, пробел, символа процента, !, &, #, @, $;

имя может состоять из любой комбинации букв, цифр, символов и начинаться с буквы;

не следует использовать имена, совпадающие с ключевыми словами VBA и именами встроенных функций и процедур;

имена должны быть уникальны внутри области, в которой они определены.

Приложение Excel включает в себя большое количество встроенных функций разных категорий: математические, финансовые, даты и времени, статистические и т.д., а так же - определенные пользователем.

Если при работе в приложении Excel требуется многократно выполнять одни и те же вычисления, то рационально воспользоваться возможностью создания новых функций с помощью языка VBA. Такие функции называются пользовательскими функциями и могут использоваться в формулах рабочего листа Excel или в других программах, написанных в VBA. Для создания и работы с пользовательской функцией необходимо выполнить следующие действия:

создать процедуру–функцию в стандартном модуле VBE;

активизировать рабочий лист приложения Excel;

выполнить команду Вставка/Функции/список Категория/Определенные пользователем. В списке функций выбрать по имени созданную функцию.

Задание 1. Создать пользовательскую функцию для решения следующей задачи. Задана себестоимость товара, процент налога на добавленную стоимость и процент возможной прибыли от продажи товара. Требуется вычислить размер цены за товар.

Для решения задачи на рабочем листе Excel должна быть оформлена следующая таблица (рис. 9). Разработанную пользовательскую функцию надо будет записать в соответствующие ячейки столбца E.

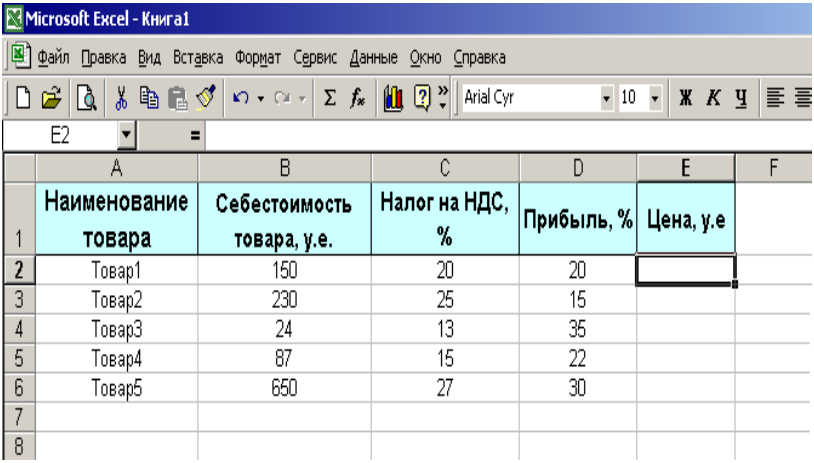
Для создания пользовательской функции необходимо выполнить следующие действия:

войти в редактор VBE (нажать клавиши Alt-F11);

вставить стандартный модуль (команда меню Insert/Module);

набрать код программы (см. рис. 10).

Тело функции в данном задании составляют три инструкции присвоения, последовательно идущие друг за другом и расположенные в отдельных строках. В общем случае инструкция присвоения предписывает выполнить выражение, заданное в правой части инструкции, и присвоить полученный результат переменной, константе или свойству объекта, чье имя указано в левой части выражения. В конце тела процедуры необходимо указать инструкцию имя = выражение для определения возвращаемого значения.



	A	B	C	D	E	F
1	Наименование товара	Себестоимость товара, у.е.	Налог на НДС, %	Прибыль, %	Цена, у.е	
2	Товар1	150	20	20		
3	Товар2	230	25	15		
4	Товар3	24	13	35		
5	Товар4	87	15	22		
6	Товар5	650	27	30		
7						
8						

Рис. 9. Таблица расчета цены товара.

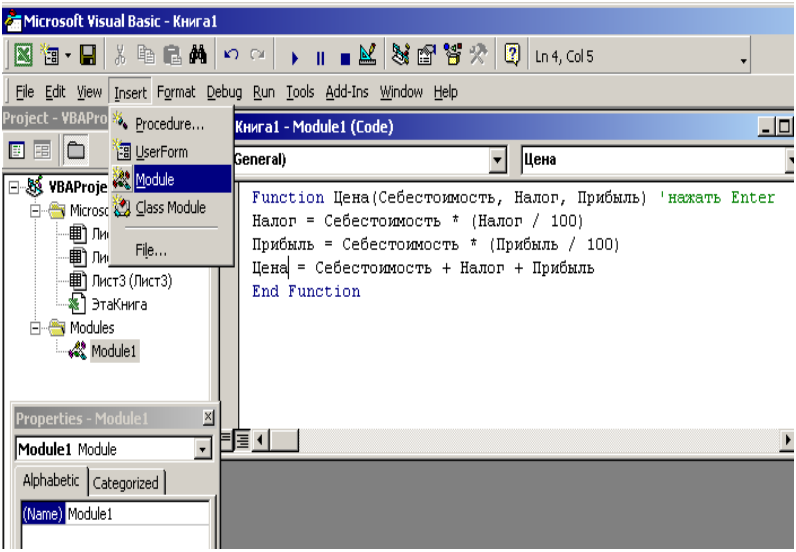


Рис. 10. Окно редактора VBE с кодом пользовательской функции

Далее выполнить следующие действия:

перейти на рабочий лист приложения Excel;

для первого товара в столбце Цена вставить созданную функцию;

скопировать функцию в последующие строки.

Задание 2. Создать программный код для вычисления функций f(x) и f1(x) в соответствии с заданным вариантом.

В VBA не существует функции, вычисляющей значение π. Поэтому для расчета ее значения используется формула ПИ = арктангенс единицы умножить на 4. Встроенная функция VBA Atn(число) вычисляет арктангенс задаваемого числа. Встроенные математические функции Cos(число), Sin(число) вычисляют соответственно значения синуса и косинуса заданного выражения.

Вариант 1	$f(x) = \cos^2(\pi x)$	$f_1(x) = \cos(\pi x)^2$
Вариант 2	$f(x) = \sin^2(\pi x)$	$f_1(x) = \sin(\pi x)^2$
Вариант 3	$f(x) = 2\cos^2(\pi x)$	$f_1(x) = 2\cos(\pi x)^2$
Вариант 4	$f(x) = 2\sin^2(\pi x)$	$f_1(x) = 2\sin(\pi x)^2$
Вариант 5	$f(x) = \cos^2(2\pi x)$	$f_1(x) = \cos(2\pi x)^2$
Вариант 6	$f(x) = \sin^2(2\pi x)$	$f_1(x) = \sin(2\pi x)^2$
Вариант 7	$f(x) = 3\cos^2(\pi x/3)$	$f_1(x) = 3\cos(\pi x/3)^2$
Вариант 8	$f(x) = 3\sin^2(2/3\pi x)$	$f_1(x) = \sin(2\pi x)^2$
Вариант 9	$f(x) = 1/5\cos^2(5\pi x)$	$f_1(x) = 1/5\cos(5\pi x)^2$

Задание 3.

Вариант 1. Правительство гарантирует, что инфляция в новом году составит i % в месяц. Какого роста цен за год можно ожидать?

Вариант 2. За первый год производительность труда на предприятии возросла на p1 %, за второй и третий — соответственно на p2 % и p3 %. Найти среднегодовой прирост производительности (в процентах).

Вариант 3. Продавец в начале каждой зимы повышает отпускную цену на молоко на p %, а каждым летом — снижает на столько же процентов. Изменится ли цена товара и если да, то в какую сторону и насколько через n лет?

Контрольные вопросы.

Технология создания пользовательской функции.

Что такое программа? Что такое инструкция? Какие типы инструкций в VBA существуют?

Назначение и использование комментариев?

Синтаксис процедуры–функции.

Правила задания имен переменным.

Синтаксис оператора объявления переменной.

Синтаксис оператора присвоения значений переменной.

Тема 4. Вычисления математических выражений по алгоритму линейной структуры

Для получения требуемых результатов при решении любой задачи необходимо построить алгоритм ее решения. Алгоритм — конечная последовательность точно определенных действий, приводящая к решению поставленной задачи. Его составление должно начинаться с изучения поставленной задачи, выявления исходных данных. Значения исходных данных определяются их вводом. Действия алгоритма осуществляются только над данными, значения которых введены или вычислены на предыдущих шагах алгоритма. Значения результатов расчета выводятся. Алгоритм линейной структуры — алгоритм, действия в котором выполняются строго последовательно друг за другом. Алгоритм должен быть описан на языке VBA. Описание алгоритма на языке программирования, предназначенное для последующего автоматического выполнения является программой для компьютера. Программы в МИФ реализуются через разные типы процедур.

Процедуры – это самые важные функциональные блоки языка VBA. В VBA кроме процедуры типа Function предусмотрена процедура типа Sub – универсальная процедура для выполнения каких-либо действий:

```
Sub имя ()
Инструкции языка VBA
End Sub
```

Тело функции

Редактор кода автоматически добавляет строку End Sub и линию разделитель между текстами процедур. Можно объявить процедуру используя команды меню Insert/Procedure.

Практически не одна программа не обходится без переменных. Переменные это контейнеры для хранения изменяемых данных. Перед работой с переменной ее надо объявить. Для этого используется инструкция Dim со следующим синтаксисом:

Dim имя_переменной1 As тип переменной, имя_переменной2 As тип переменной, …

Для хранения различных типов данных в VBA используются соответствующие типы переменных, например:

Byte — целые числовые значения из диапазона от 0 до 255;

Integer — целые числовые значения из диапазона от –32 768 до 32 767;

Long — целые значения чисел из диапазона от –-2 147 483 648 до 2 147 483 647;

Single и Double —значения чисел с плавающей точкой одинарной (Single)точности, т.е. с мантиссой, округленной до 10 значащих цифр, и двойной точности (Double);

Currency — большое десятичное число с 19 позициями, включая 4 позиции после запятой;

Decimal — большое десятичное число с 29 позициями, после запятой можно использовать от 0 до 28 позиций;

String — используется для хранения символьных (строковых) значений. Длина строки символов от 1 до 65400 байтов для строк с фиксированной длинной;

Date — используется для хранения значений даты и времени (от 01.01.100 до 31.12.9999);

Boolean — используется для хранения значений True и False;

Object — хранит ссылку на любой объект в памяти;

Variant — может использоваться для хранения любых данных. При использовании этого типа данных не самым экономным способом расходуется память, и переменные этого типа требуют дополнительного времени на обработку.

В одном операторе Dim допускается определять различные типы переменных. Например, Dim k as Byte, m, i as integer.

По доступности к своим значениям в программном коде переменные разделяются на локальные, модульные и глобальные.

Если объявление переменной оператором Dim производится внутри процедуры, то такая переменная действует только внутри данной процедуры и является локальной. Переменные уровня модуля используются только в модуле, в котором они описаны при помощи оператора Dim, размещенной в области описания модуля – – перед описанием процедур. Общие переменные, используемые во всех модулях данного проекта, описываются при помощи инструкции Public, размещаемой в области описания модуля.

В программах на VBA можно использовать стандартный набор операций над данными. Имеются следующие основные типы операций: математические, отношения, логические, строковые.

Математические операции на языке VBA выполняются над числами и их результатом являются числа. Числа могут быть результатом математических выражений. В математических выражениях записываются константы, переменные, встроенные функции, соединенные знаками математических операций. Математические операции задаются следующими символами: + (сложение), - (вычитание), * (умножение), / (деление), ^ (возведение в степень), \ (целочисленное деление), Mod (остаток от деления по модулю). Например, 9\5 возвращает 1, а 9 mod 5 возвращает 4. Сами выражения записываются в одну строку. Перенос строки задается символами пробел и _ (подчеркивание). Строка не должна быть более 1024 символов и допустимо не более семи продолжений строки.

Остальные типы операций VBA будут рассмотрены в последующих темах.

В VBA имеется большой список математических функций, позволяющих произвести любые вычисления. В таблице 1 приведен синтаксис математических функций и возвращаемые ими значения.

Таблица 1.

Функция	**Возвращаемое значение**
Abs(число)	модуль числа
Atn(число)	арктангенс числа (угол в радианах)
Cos(число)	косинус числа (угол в радианах)
Exp(число)	экспонента, т.е. результат возведения основания натурального логарифма в указанную степень
Log(число)	натуральный логарифм числа
Rnd(число)	случайное число из интервала [0,1). Если *число* < 0, то функция возвращает одно и то же число; если *число* > 0 или аргумент опущен, то следующее случайное число в последовательности; если *число* = 0, то случайное число, возвращенное при предыдущем вызове функции. Перед вызовом функции Rnd должна использоваться инструкция *Randomize*.
Sin(число)	синус числа (угол в радианах)
Sqr(число)	квадратный корень из числа
Tan(число)	Тангенс числа (угол в радианах)

Диалоговый ввод значений переменных может осуществляться с помощью функции InputBox, которая при выполнении в программе выводит на экран свое собственное окно (рис. 11).

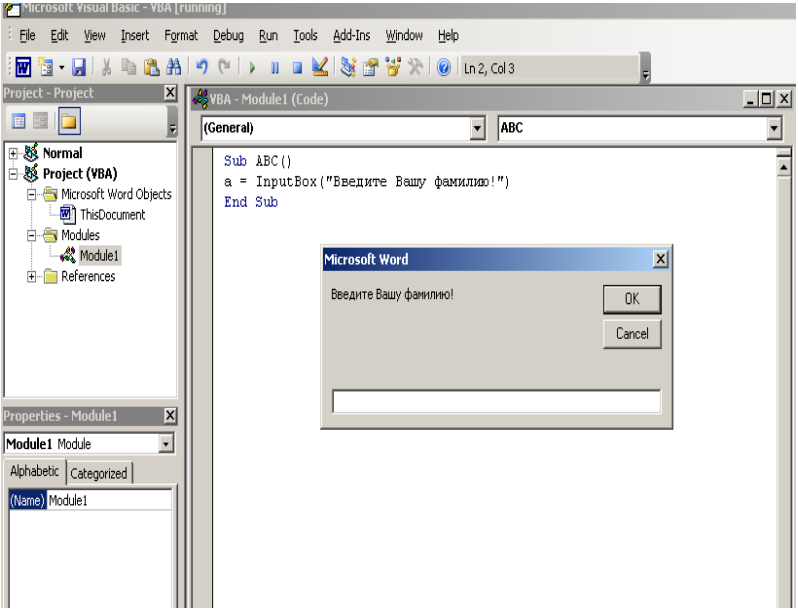


Рис. 11. Диалоговое окно ввода значений функции InputBox.

Синтаксис функции:

InputBox(сообщение[, заголовок] [, по умолчанию] [, позицияX] [, позицияY] _[, файл_справки] [, номер_раздела]),

где

сообщение — сообщение в диалоговом окне, представленное строкой символов;

заголовок — заголовок диалогового окна, представленный строковым выражением;

по умолчанию — строковое выражение, отображаемое в поле ввода по умолчанию;

позицияX — числовое выражение, которое задает расстояние по горизонтали между левой границей диалогового окна и левым краем экрана;

позицияY — числовое выражение, которое задает расстояние по вертикали между верхней границей диалогового окна и верхним краем экрана;

файл_справки — имя файла справки, содержащего справочные сведения о данном диалоговом окне;

номер_раздела — номер соответствующего раздела справочной системы.

Например, при выполнении следующей строки программного кода: a=InputBox("Введите фамилию")

на экране появится диалоговое окно, в котором будет записан текст, заключенный в кавычки с курсором в поле ввода значения.

После чего необходимо ввести запрашиваемое в окне значение и нажать клавишу ввода или щелкнуть мышью по кнопке OK. При этом возвращаемое функцией InputBox значение имеет тип string. Поэтому при вводе числового значения переменной в программе необходимо предусмотреть преобразованиеь получаемого значения в числовой тип. Для этого следует воспользоваться одной из функций преобразования типов данных от строкового к числовому. В VBA имеются функции преобразования типов выражений. Некоторые из них представленны в таблице 2.

Таблица 2.

Функции	**Тип, в который преобразуется выражение**
CByte(выражение)	byte
CSng(выражение)	single
CDate(выражение)	date
CInt(выражение)	integer
CLng(выражение)	long
CStr(выражение)	string

Для вывода результатов в VBA существуют различные способы. Одним из операторов вывода значений переменных в языке VBA является оператор Debug.Print, Этот опнртор в терминах объектно-ориентированного программирования рассматривается как метод Print, действующий на объект Debug. Оператор Debug.Print позволяет выводить промежуточные значения или результаты вычислений в отладочное окно Immediate. Это окно отображается на экране после выполнения команды меню View/Immediate Window или по-

сле нажатия клавиш Ctrl-G. Например, после выполнения инструкции Debug.Print "Сумма массива S= ", s вид окна Immediate будет как на рис. 12.

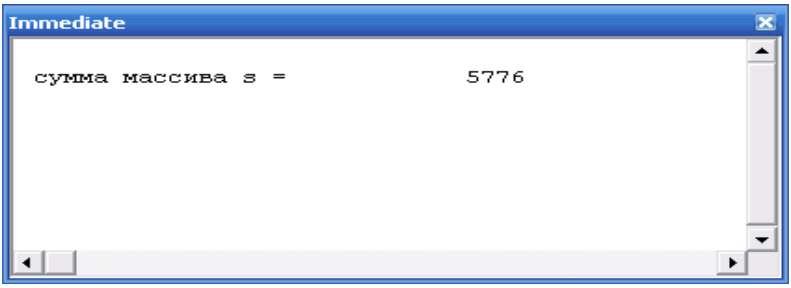


Рис. 12. Отладочное окно Immediate

Для форматирования выводимого значения в языке VBA используется функция Format, имеющая два аргумента: первый — имя переменной или выражение, значение которых форматируется, второй — в кавычках указывается тип формата для выводимого значения. Например, для округления выводимого значения переменной z = 12,34507 до двух значащих цифр после десятичной запятой необходимо записать оператор в следующем виде: Debug.Print Format (z,"#.#0"). После выполнения в программе в окно Immediate будет выведено z = 12,35. Символ "#" используется для задания пользовательского формата (количества цифр в выводимом числе до и после запятой, а 0 служит для вывода незначащего нуля). Кроме того, для резервирования места при выводе числа можно использовать символы пробела до и после символов "#". Это удобно при выводе списков. Например, при выводе значений списка переменных y, z следующей строкой кода:

```
Debug.Print Format(y,"   ##.##"), Format(z,"   ##.##")
```

перед каждым числом будет сделан отступ, в соответствии с количеством пробелов, введенных перед первым символом "#". Для каждой переменной в выводимом списке для форматирования ее вывода необходимо использовать функцию Format отдельно. При построении пользовательских форматов можно использовать следующие символы: 0 (резервирует позицию цифрового разряда), % (резервирует процентное отображение числа),’ (разделитель разряда сотен от тысяч), E+,E- (разделитель мантиссы и порядка в экспоненциальном формате).

Кроме пользовательских существуют именованные форматы. К ним относятся: General Number (число без разделителя тысяч), Currency (использует денежные установки страны в Панели управления.), Fixed (отображает, по крайней мере, одну цифру слева и две справа от десятичной точки), Percent (отображает числа в виде процентов и выводит две цифры справа от десятичной точки), Scientific (использует формат с плавающей десятичной точкой).

В случае обнаружения редактором VBE синтаксических ошибок, они выделяются красным цветом при переходе к следующей строке кода. В этом случае необходимо вернуться курсором к строке, где обнаружена ошибка, и исправить ее.

При выполнении программы в ней могут быть обнаружены ошибки выполнения программ, такие как деление на ноль, неправильное использование оператора. В этом случае происходит прерывание в выполнении программы, появляется диалоговое окно с указанием кода ошибки. Каждая ошибка имеет свой код. Например, 6 — переполнение, 7 — не хватает памяти, 11 — деление на ноль, 13 — несоответствие типа. В этом же диалоговом окне можно выбрать режим работы: остановка программы End, переход в режим отладки программы Debug, получение справки Help по замеченной ошибке.

Задание 1. Создать программный код, который в окне Immediate показывает, как работают различные пользовательские форматы.

Задание 2. Создать программный код, который в окне Immediate показывает, как работают различные именованные форматы.

Задание 3. Создать программный код, в котором ввод данных осуществляется функцией InputBox. Использовать различные аргументы функции InputBox. Вывод данных осуществлять в окно Immediate, форматируя выводимые значения. Задание выполнять в соответствии с заданным вариантом.

Таблица 3.

№ варианта	**Формула для вычислений**	**Исходные данные**
1	$z = 1/(x\text{-}1) + \sin^2 x - \sqrt{nb+n}$ $y = (e^{-b}+1)/2n$ $g = \ln(n/4)/(1+b)$	Вводить n=2; b=0.125; Задать x=1.25*10^{-4}

2	$d = \text{arctg}(-x^2 * i) / \sqrt{x - z}$ $f = \sin(2d)/d$ $g = \ln(d/2) + e^{-x}$	Вводить $i = -6$; $x = 0.195$; Задать $z = 1.25*10^{-6}$
3	$p = 2.6*t + \cos(y/(0.003x+y))$ $q = \sin t/\cos t$ $s = 10e^{-0.1p} * \sqrt{t - 0.1*t}$	Вводить $t = 6$; $y = -1.254$; Задать $x = 0.15*10^{6}$
4	$w = (a + b)^2 + 10^{-6} * \text{tg}\, x/(x+1)$ $v = 1/2b - \sqrt{w - ab}$ $s = 100*e^{-0.2b}/(0.02*v^2 - 2.1x)$	Вводить $b = 40$; $x = 1.055$; Задать $a = 5*10^{-6}$
5	$t = 1/\sqrt{y + 14}$ $u = (t+1)/(t^3+1)$ $s = \ln((2a/3) + e^{-a}/(1.05 + u)$	Вводить $y = 0.956$; Задать $a = 5*10^{-6}$
6	$y = \sqrt{1 + x^2}$ - $\cos(2/m)/\sin(0.9m)$ $w = 0.4y - 2e^{-0.2ym} + z$	Вводить $x = 1.24$; $m = 6$; Задать $z = 0.05*10^{-5}$
7	$s = 0.4*x^3 - 1/(jx + j^2)*\text{tg}\, y$ $t = s^5 - arctg\, s$ $u = e^{-t}/(1 - \sqrt{x})$	Вводить $x = 0.911$; $j = 12$; Задать $y = 5*10^{-6}$
8	$z = e^{-cx}/y$ $u = \sqrt{z^3 - 0.1*z}$ $w = 1/\ln((0.1u) + y)/1+e^{y}$	Вводить $y = 0.555$; $c = 1.44$; Задать $x = 2*10^{-4}$
9	$s = \sqrt{ta/(t+1)} + 4*e^{2b}$ $w = sa/(1+0.1a)$ $v = s + (j + 0.5)*\sqrt{a^2 + b^2}$	Вводить $a = 1.75$; $b = -8.1$; Задать $j = 4$; $t = 45*10^{-8}$

Контрольные вопросы.
Для чего используются переменные в программе?
Основные типы данных в VBA.

Что значит объявить переменную в программе?
Математические операции, операции отношения.
Логические операции, операция конкатенации.
Назначение и режимы использования функции InputBox.
Назначение и использование объекта Debug и метода Print.
Использование функции Format. Именованные форматы.
Использование функции Format. Пользовательские форматы.

Тема 5. Условные операторы. Алгоритмы разветвляющейся структуры.

Если какие-либо действия должны происходить только при выполнении некоторого условия, то говорят, что такой процесс имеет разветвление. Выбор варианта действий обеспечивается условным оператором, который таким образом осуществляет управление в программе. Алгоритм, порядок выполнения действий в котором зависит от итогов проверки условия, называется алгоритмом разветвляющейся структуры.

Для формулировки условия в VBA используются следующие логические операции: > (больше), >= (больше или равно), < (меньше), <= (меньше или равно), = (равно), <> (не равно). Например, x>y, a<=sin(b+1) и т.д. Для формирования сложного условия, когда необходимо проверить сразу несколько условий, используются операции их объединения. Это логические операции: And (должны выполняться оба условия), Or (должно выполняться хотя бы одно из условий), Not (отрицание условия). Следующее сложное условие x>y And a>=5 будет выполнено только тогда, когда выполняются одновременно оба условия x>y и a>=5. Сложное условие x>y Or x=a будет выполнено тогда, когда выполняется условие x>y либо a>=5. Комбинируя условия с операциями And, Or и Not можно составить и более сложные условия.

При проверке любое условие может принимать только одно из двух значений: либо True (истина), когда условие выполняется, либо False (ложь), когда оно не выполняется. Проверка выполнимости условий в программах используется довольно часто и производится многими операторами, в которых само условие является их частью.

В системе программирования VBA имеется специальный условный оператор If, который имеет два варианта синтаксиса. Если при

проверке условия должен выполняться только один оператор, запись производится в одну строку в виде

If условие Then оператор1 [Else оператор2]

Логический смысл этой конструкции следующий: если условие выполняется, то следует выполнить оператор1, иначе следует выполнить оператор2. Квадратные скобки говорят о том, что ветвь [Else оператор2] является необязательной В этом случае условный оператор называется строчным If. Например, если скидка применяется только к суммам больше 150 у.е., то можно записать:

If сумма>150 Then скидка=0.03 Else скидка=0

Если при проверке условия должны выполняться несколько инструкций, то условный оператор записывается в блочной форме:

```
If условие Then
    блок операторов1
[Else
    блок операторов2]
End If]
```

В этом случае условный оператор называется блочным If.

Условный оператор If работает следующим образом. Если условие после слова If является истинным, выполняется группа операторов после слова Then, а если условие является ложным, то выполняется группа операторов после слова Else. В случае, когда ветвление в задаче отсутствует, то в операторе If отсутствует ветвь Else и происходит переход к следующему по тексту программы оператору.

Например, предыдущий пример можно записать в блочной структуре следующим образом:

```
If сумма>150 Then
    скидка=0.03
Else
    скидка=0
End If
цена = цена – скидка*цена
```

Для случаев, когда необходимо проверить более одного условия, можно использовать вложение операторов If друг в друга или оператор If…Then…ElseIf, который позволяет проверять множественные условия. Имеется два варианта синтаксиса этого оператора: в одну строку и в форме блока. В первом случае он имеет вид:

If условие Then оператор1 [ElseIf оператор2] [ElseIf оператор3]

Во втором случае оператор расположен на нескольких строках:

```
If условие1 Then
    операторы1
ElseIf условие2 Then
    операторы2
    . . .
ElseIf условиеj Then
    операторыj
    . . .
Else
    операторыN
End If
```

Здесь условие обязательно в обоих вариантах. Если условие истинно, выполняются операторы1, нет - отыскивается первое истинное условие-j, и выполняются операторы-j. Если все эти условия ложны, выполняются Else операторы-N. После выполнения одной последовательности операторов управление передается оператору, следующему за End If. Например,

```
If x<0.2 Then
    Z=1+log(1+x)
ElseIf x<=0.8 Then
    Z=(1+x^2(1/2))/(1+x)
Else
    Z=2*exp(-2*x)
End If
```

Инструкция Select Case в зависимости от выражения выбирает и исполняет одну из последовательностей инструкций.

```
Select Case Выражение
[Case СписокВыражений1
    инструкции1]
    . . .
[Case СписокВыраженийj
    инструкцииj]
[Case Else
    инструкции_else]
End Select
```

Выражение должно присутствовать обязательно. Оно может быть произвольным выражением с числовым или строковым значе-

нием. СписокВыраженийj должен присутствовать в строке, начинающейся ключевым словом Case (Случай). Выражения в этом списке отделяются запятыми и могут иметь одну из форм:

выражение,

выражение-нижняя-граница To выражение-верхняя-граница,

Is оператор-сравнения выражение.

Первая форма задает отдельные значения, вторая и третья позволяют задавать сразу целые диапазоны значений. Последовательность инструкцииj необязательна. Она будет исполнена, если соответствующий СписокВыраженийj является первым списком и сопоставимым с текущим значением Выражение. После исполнения инструкцииj проверка на соответствие другим спискам выражений не производится, и управление передается на оператор, следующий за End Select. Необязательная последовательность Case Else выполняется, если ни один из списков выражений несопоставим со значением Выражение. Пример использования оператора выбора Select Case:

```
Целое=InputBox("Введите целое число")
Select Case Целое
Case 1
    Debug.Print "Число равно 1"
Case 2, 3
    Debug.Print "Число равно 2 или 3"
Case 4 to 6
    Debug.Print "Число от 4 до 6"
Case Is>6
    Debug.Print "Число не менее 7"
Case Is<=0
    Debug.Print "Отрицательное число"
End Select
```

Вывод результата пользователю

Вывести вычисленное значение можно с помощью встроенной процедуры вывода MsgBox. При выполнении эта процедура активизирует свое собственное окно сообщений на экране. Вид окна MsgBox представлен на рис. 13.

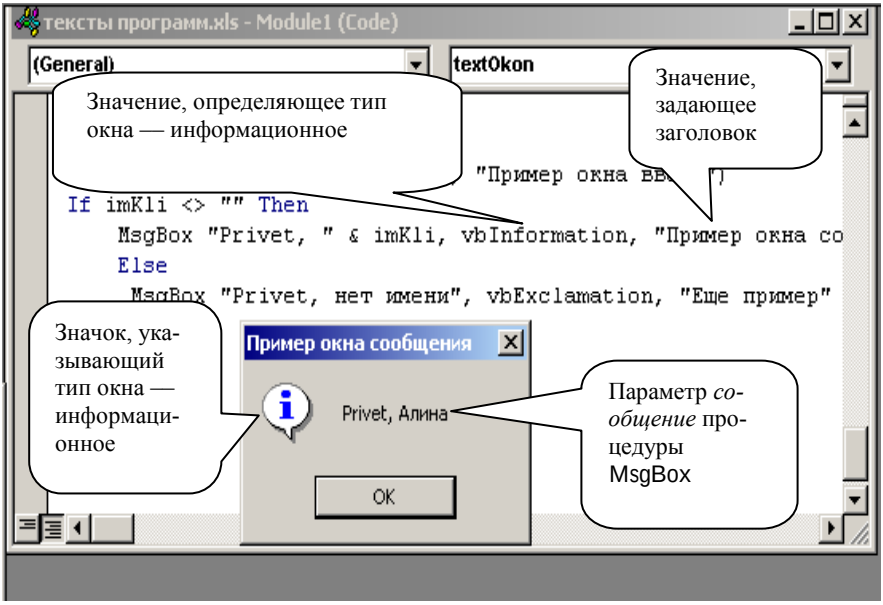


Рис. 13. Окно сообщений процедуры MsgBox

Синтаксис процедуры MsgBox:

MsgBox сообщение, кнопка, заголовок, имя файла справки, раздел справки,

где сообщение — это строковое выражение, которое отображается как сообщение в диалоговом окне.

кнопка — числовое выражение, представляющую сумму значений. Эти значения указывают: 1. сколько и каких кнопок будет отражено в окне сообщений; 2. тип отображаемого информационного значка; 3. какая из отображаемых кнопок будет основной. По умолчанию этот аргумент равен 0.

заголовок — строковое выражение, которое отображается в строке заголовка диалогового окна. По умолчанию в строке заголовка помещается имя приложения.

имя файла справки — имя файла справки, содержащего справочные сведения о данном диалоговом окне;

раздел справки — номер соответствующего раздела справочной системы.

Задание 1. Создать программные коды, с помощью которых продемонстрировать вывод разных типов данных и сообщений в диалоговых окнах MsgBox разного типа. Оформить выход из программ с выдачей соответствующего сообщения при нажатии кнопки Cancel.

Диалоговые окна MsgBox могут иметь различные кнопки (vbOKOnly, vbOKCancel, vbAbortRetryIgnore, vbYesNoCancel, vbYesNo,vbRetryCancel), различные информационные значки (vbCritical, vbQuestion, vbExclamation, vbInformation) и различные основные кнопки (vbDefaultButton1, vbDefaultButton2, vbDefaultButton3, vbDefaultButton4). В зависимости от того, какая кнопка диалогового окна будет нажата, соответственно будут возвращены определенные значения.

Если нажата кнопка OK, то константой vbOK возвращается значение 1. Если нажата кнопка Cancel (Отмена), то константой vbCancel возвращается значение 2. Если нажата кнопка Abort (Прервать), то константой vbAbort возвращается значение 3.

Если нажата кнопка Retry (Повторить), то константой vbRetry возвращается значение 4. Если нажата кнопка Ignore (Пропустить), то константой vbIgnore возвращается значение 5. Если нажата кнопка Yes (Да), то константой vbYes возвращается значение 6. Если нажата кнопка No (Нет), то константой vbNo возвращается значение 7.

Задание 2. Составить программный код для вычисления формулы (вариант задания представлен в таблице 4) следующей задачи. Ввод исходных данных организовать с помощью встроенной функции InputBox. Вывод результата расчета организовать с помощью встроенной процедуры MsgBox и в окно Immediate с использованием функции Format.

Таблица 4

№ п/п	Формулы для вычислений	Исходные данные

1.	$g=\begin{cases}\dfrac{1+x^2}{\sqrt{1+x^4}} \\ 2x+\dfrac{\sin^2(x)}{2+x}\end{cases}$	$x \le 0$ $x > 0$
2.	$g=\begin{cases}3\sin(x)-\cos^2(x) \\ 3\sqrt{1+x^2}\end{cases}$	$x \le 0$ $x > 0$
3.	$y = e^{-2x}+1;$ $\quad z = \dfrac{\ln x}{x+1};$ $w = \begin{cases}\sqrt{xy}, \\ nx+2\end{cases}$	 $x < z^2$ $x \ge z^2$
4.	$x = tg(a^2-1)/(d+1),$ $y = \begin{cases}ak+d, 3x < ac \\ \cos(ak)*e^{a+1}, 3x \ge ac\end{cases}$	a = 2; c = 3.7; d = $51.9*10^{-5}$; k = 4
5.	$g = \begin{cases}\dfrac{\sqrt{1+\lvert x\rvert}}{2+\lvert x\rvert}, x \le 0 \\ \dfrac{1+x}{2+\cos^3(x)}, x > 0\end{cases}$	
6.	$p = e^{\sin(y/x)} * \ln(x/y) * x;$ $q = \begin{cases}\sqrt{p/m}, p \le y^2 \\ \sqrt{2x}/(y+p), p > y^2\end{cases},$	x = 1.1; y = $1.4*10^{-3}$; m = 4

7.	$g = \begin{cases} \sin^2(x) + \dfrac{1+x}{1+\cos^2(x)}, x > 0 \\ \sqrt[3]{1+x^2}, x \le 0 \end{cases}$	
8.	$g = \begin{cases} \dfrac{3x^3}{1+x^2}, x \le 0 \\ \sqrt{1+\dfrac{2x}{1+x^2}}, x > 0 \end{cases}$	
9.	$b = 12a\text{-}e^{-s/2} * (x\text{-}j/a);$ $z = \begin{cases} \sqrt{-2xj} + b, a < 1.5 \\ \lvert 13xj \rvert + b, a \ge 1.5 \end{cases}$	$x = -4*10^{-3}$; $s = 1.1$; $a = 15$; $j = 4$
10.	$y = \sqrt{a + a^2 * x^2}\ /(a+x) * m;$ $z = \begin{cases} y+1, \lvert y \rvert < 1 \\ \sin^2 y, \lvert y \rvert \ge 1 \end{cases}$	$a = 1.774$; $x = -27$; $m = 5$

Задание 3. Составить программный код следующей задачи. Ввод исходных данных организовать с помощью встроенной функции InputBox. Вывод результата расчета организовать с помощью встроенной процедуры MsgBox.

За одну смену первый маляр покрасил a м2 поверхности, а второй b м2, причем b < a. Если разность a - b ≤ c, то оба маляра заканчивают работу, а если a – b > c, то второй маляр должен продолжать работу до тех пор, пока площадь окрашенной им поверхности не станет такой же, как у первого маляра, т.е. a м2. Рассчитать площадь поверхности, окрашенной двумя малярами.

Задание 4. Деление на 3. Как известно, число делится на 3 тогда и только тогда, когда сумма его цифр делится на 3. Проверить этот признак на примере заданного трехзначного числа.

Замечание. Теоретическое утверждение о признаке делимости предлагается проверить на примере любого вводимого числа.

Задание 5.. Путник двигался t1 часов со скоростью v1 км/ч, затем t2 часов — со скоростью v2 км/ч и t3 часов — со скоростью v3

км/ч. За какое время он одолел первую половину пути, после чего запланировал привал?

Задание 6.. Можно ехать на такси со скоростью v1км/ч и оплатой p1 р./км либо идти пешком со скоростью v2 км/ч бесплатно. Как с наименьшими затратами преодолеть путь s за время t, если это возможно? Каковы эти затраты?

Надо рассмотреть «запредельные» случаи: когда времени слишком мало, чтобы успеть даже на такси, либо слишком много, так что и пешком можно с запасом успеть до отхода поезда.

Задание 7.. Составить программный код для решения следующей задачи. Предусмотреть контроль значения стипендиального фонда. Факультету выделен стипендиальный фонд в размере f р./мес. Результаты сессии таковы: n1 «отличников», n2 «хорошистов», n3 «троечников». Повышенная стипендия (для отличников) составляет s1 р., обычная — s2р., задолжники стипендии лишаются. Сколько студентов каждой категории могут получать стипендию? Каков будет остаток фонда на материальную помощь малоимущим?

Контрольные вопросы.

Какие логические операции используются в VBA?

Как задать условие для выполнения условного оператора?

Как задать сложное условие в условном операторе?

Назначение ключевых слов Else и ElseIf в условных операторах.

Правила использования ключевых слов and, or в сложном условии.

Типы условных операторов в VBA.

Назначение и синтаксис встроенной процедуры MsgBox.

Тема 6. Операторы циклов. Алгоритмы разветвляющейся структуры.

В тех случаях, когда необходимо повторить последовательность операций несколько раз, используются *операторы циклов.* В системе программирования VBA имеется большой выбор средств организации циклов, такие как:

- **For ... Next**
- **For Each…Next**
- **Do ...Loop**
- **While…Wend**

Оператор цикла **For ... Next** повторяет выполнение группы инструкций указанное число раз. Наиболее часто используется следующая конструкция цикла **For ... Next**:

For *счетчик=начальное значение* **To** *конечное значение* [**Step** *приращение*]

[*операторы*]

[**Exit For**]

Next [*счетчик*]

Здесь счетчик - это числовая переменная. В начале выполнения цикла она принимает значение, задаваемое числовым выражением начальное значение. Числовое выражение конечное значение задает заключительное значение счетчика цикла. Оно вычисляется до начала исполнения тела цикла и не меняется, даже если входящие в него переменные изменяют в теле цикла свои значения. Числовое выражение приращение необязательно. Его значение также вычисляется в начале цикла и прибавляется к счетчику цикла всякий раз, когда завершается выполнение тела цикла и вычисление достигает строки Next. Приращение (шаг изменения значения счетчика) может быть как положительным, так и отрицательным числом. Если приращение равно единице, то конструкция Step может быть опущена. Если приращение отрицательно, то начальное значение, должно быть больше конечного. В операнде Next счетчик может быть опущен, однако это делать не рекомендуется. Тело цикла - это последовательность операторов, которая будет выполнена заданное число раз Альтернативный выход из цикла предоставляет инструкция Exit For. Если шаг положителен, цикл завершится, когда впервые выполнится условие: счетчик больше конечного значения. Если шаг цикла отрицателен, условие его завершения: счетчик меньше конечного значения. Это условие проверяется перед началом выполнения цикла, а затем - после каждого прибавления шага к счетчику цикла в операторе Next. Если оно выполнено, управление передается на оператор, следующий за Next, нет - выполняются операторы тела цикла. Нельзя менять значение счетчика в теле цикла. Такой цикл возможно никогда не завершится. Например:

```
Sub CiklFor()
Dim A As Integer
Dim i As Integer
For i = 5 To 1 Step -1
```

```
A=A + i
i = i + 1
Next i
End Sub
```

В представленном далее примере при помощи оператора цикла наращивается значение переменной S:

```
S = 0
For i = 1 To 20
      S = S + cos(( i )^3)
Next i
Debug.Print "s="; S
```

Обычно попытка изменить значение счетчика цикла в его теле означает, что вместо цикла For…Next следовало бы применить цикл другого вида, например Do…Loop.

Пример 1. Программа вычисления таблицы значений функции $y = ae^x$, где x = 1.6 (0.2) 2, т.е. x изменяется от 1,6 до 2 с шагом 0,2, n = – 1,25

```
Dim x as Single, y as Single, a as Single
a = -1.25
For x = 1.6  To 2  Step 0.2
      Y = a*Exp(- x)
      Debug.Print x, y
Next x
```

Цикл Do…Loop. повторяет блок операторов, пока заданное условие является истинным или пока оно не станет истинным.

Имеется четыре варианта синтаксиса этого цикла. В двух первых вариантах условие проверяется в начале цикла:

```
Do {While | Until} условие
тело цикла
[Exit Do]
Loop
```

В других двух вариантах условие проверяется в конце цикла:

```
Do
тело цикла
[Exit Do]
Loop {While | Until} условие
```

Циклы могут быть вложенными, когда внутри одного циклического процесса должен выполняться другой.

Условие является числовым или строковым выражением со значениями True или False. Вообще оно необязательно. Значение Null условия трактуется как False. Тело цикла - это последовательность операторов, которая будет выполняться, пока условие остается истинным, если перед ним стоит ключевое слово While или пока оно остается ложным - в вариантах цикла с ключевым словом Until. Таким образом, циклы вида While условие эквивалентны циклам вида Until Not условие. Кроме того, в тело цикла может входить оператор Exit Do, выполнение которого сразу прекращает цикл и передает управление оператору, непосредственно следующему за Loop. В случае нескольких вложенных циклов Do … Loop оператор Exit Do завершает лишь самый внутренний цикл, в теле которого он расположен.

Пример 2. Программа вычисления таблицы значений функции $y = ae^x$, где x = 1.6 (0.2) 2, т.е. x изменяется от 1,6 до 2 с шагом 0,2, n = – 1,25 с использованием оператора цикла с условием While. Перед началом цикла переменной x присваивается начальное значение 1,6. Затем в начале цикла проверяется условие x < 2, где 2 – конечное значение x. Если это условие верно, то вычисляется значение x, увеличенное на величину шага, и выполняются все операторы, входящие в тело цикла. Как только условие x < 2 станет неверным, цикл завершается. Завершается и выполнение программы.

```
Dim x as Single, y as Single, a as Single
a = -1.25
x = 1.6
Do While x < 2
      y = a*Exp(-x)
      Debug.Print x, y
      x = x + 0.2
Loop
```

Пример 3. Программа вычисления суммы ряда $y = \sum_{n=1}^{\infty} \frac{n}{n^2 + 1}$, включая все члены ряда, пока они не станут меньше 0,001.

```
Dim n as Integer
Dim x as Single, y as Single
n = 1
```

```
y = 0
Do
    x = n/(n^2 + 1)
    y = y + x
    n = n + 1
Loop While x > 0.001
Debug.Print “Сумма = ”, y
```

Пример 4. Программа вычисления таблицы значений функции y = aex, где x = 1.6 (0.2) 2, т.е. x изменяется от 1,6 до 2 с шагом 0,2, n = – 1,25 с использованием оператора цикла с помощью оператора Until и проверкой условия в конце цикла.

```
    Dim x as Single, y as Single, a as Single
a = -1.25
    x = 1.6
    Do
        y = a*Exp(-x)
        Debug.Print x, y
        x = x + 0.2
    Loop Until x > 1.8
```

Пример 5.

Ожидание правильного ввода пароля может быть задано следующим программным кодом:

```
Dim пароль As String
Do
пароль = InputBox(“Введите правильный пароль!”)
Loop Until пароль = “Алгоритм”
```

Пример 6.

Программа вычисления произведения $y = \prod_{k=0}^{\infty}(k^2+1)$ до тех пор, пока произведение y из сомножителей (k^2+1) не достигнет значения больше 1000.

```
Dim k as Integer
Dim x as Single, y as Single
a = -1.25
y = 1
```

```
k = 0
Do Until y > 1000
    x = k^2 + 1
    y = y*x
    k = k + 1
Loop
Debug.Print “Произведение = ”; y, “вошло”; k; “ сомножителей”
```

Оператор **For Each…Next** будет рассмотрен в следующих лабораторных работах.

Оператор **While…Wend** выполняет последовательность инструкций, пока заданное условие имеет значение True. Оператор While…Wend повторяет выполнение последовательности операторов, пока заданное условие не станет ложным. Синтаксис команды:

While условие
    [инструкции]
Wend

Условие является обязательным параметром. Может принимать числовое или строковое выражение, которое имеет значение True или False. Если условие имеет значение Null, условие рассматривается как имеющее значение False.

Если условие имеет значение True, выполняются все инструкции до ключевого слова Wend. Затем управление возвращается инструкции While и вновь проверяется условие. Если условие попрежнему имеет значение True, процесс повторяется. Если оно не имеет значение True, выполнение возобновляется с инструкции, следующей за ключевым словом Wend.

Циклы While...Wend могут иметь любую глубину вложенности. Каждая инструкция Wend соответствует предшествующей инструкции While. Инструкция Do...Loop обеспечивает более структурированный и гибкий способ организации циклов.

Пример 7.

Игральная кость бросается до тех пор, пока не выпадет шесть очков. При выпадении шести очков игра заканчивается и отображается сообщение с указанием, на каком броске она закончилась.

```
Dim brosok as integer, ochki as integer
Radomize
Ochki = Int(6*Rnd()+1)
Brosok = 1
```

```
While ochki<6
    Brosok = Brosok + 1
    Ochki = Int(6*Rnd()+1)
Wend
MsgBox "Победа на"; Brosok; "броске"
```

Задание 1. Дана функция y = 0.5+sin 5x, причем x изменяется от 0 до 2π с шагом Δx = π/6. Составить программный код для печати сообщений ДА, если y ≥ 0, и НЕТ, если y <0.

Задание 2. Составить программный код для расчета функции y при значениях x = 0; 0,1; 0,2; …; 10:

$$y = \begin{cases} \dfrac{x-1}{2x^2+3}, если \ldots x \leq 1 \\ 1.05(x-1)^2, если \ldots x > 1 \end{cases}$$

Задание 3. Цех вводится в строй постепенно, выдавая в первый день x1% продукции от нормы, во второй день — x2%, в третий день — x3%, …, в n–й день — xn%. Составить программный код для расчета продукции s за n дней, если в первый день цех выдал A[т] продукции. Результат вычисления вывести на печать.

Задание 4. При сооружении железобетонного фундамента площадь сечения арматуры определяется по формулам:

F = (N-RS)/(D - R), если *F*>0.03 *S*
F = (N-RS)/D, если *F*<0.03 *S*

Где *N* — расчетная сила, приложенная по оси элемента; *S* — площадь сечения бетона; *R* — сопротивление бетона осевому сжатию; D — сопротивление сжатой арматуры. Составить программный код для расчета *N*, если известны *R, D, S*, а *F* изменяется от *F1* до *F2* с шагом Δ*F.*

Задание 5. Дана числовая последовательность {8,2 7,9 7,6 }. Найти сумму всех положительных членов.

Задание 6. Составить программный код для расчета функции y = (10sin dx)/(1+d2x2), если x изменяется от 0,1 до 10 с шагом Δx = 0.13, a d от 1,2 до 5,4 с шагом Δd = 1.1.

Задание 7. Дана функция x = a sin kt + 2 cos kt. Составить программный код для расчета этой функции, если a изменяется от 5 до

7 с шагом 0,12, t изменяется от 4,2 до 6,2 с шагом 0,17, а k = 1,2,3, …, 12.

Задание 8. При забивании сваи в грунт предельное сопротивление грунта определяется по формуле

$$p = \frac{NF}{2}\left(\sqrt{1+\frac{4}{NF}\times\frac{QH}{E}\times\frac{Q+0.2G}{Q+C}}-1\right),$$

где F – площадь поперечного сечения сваи; см^2; E – погружение сваи от одного удара, см; Q – масса ударной части молота; G – масса сваи, т; H – высота падения ударной части молота, см; N – коэффициент, зависящий от материала сваи.

Составить программный код для выдачи на печать таблицы значений P, если F, N, E и G известны и постоянны, Q изменяется в пределах от Q_1 до Q_2 с шагом ΔQ, а H – в пределах от H_1 до H_2 с шагом ΔH.

Задание 9. Составить программный код для вычисления функции $y = \sum_{i=1}^{\infty} \sin\frac{x}{2i+1}$ при заданном x с точностью до ε.

Задание 10. Дана функция y= –2x2 + 3x +1,5. Составить программный код для поиска максимального значения y. Если x изменяется в диапазоне $0{,}1 \le x \le 1$ с шагом $\Delta x = 0.01$

Задание 11.Предприниматель, начав дело, взял кредит размером k рублей под p процентов годовых и вложил его в дело. По прогнозам, его дело должно давать прибыль r рублей в год. Сможет ли он накопить сумму, достаточную для погашения кредита, и если да, то через сколько лет?

Задание 12. Вычислить наибольшее целое положительное число A, удовлетворяющее условию:

$3A^2 - 127\,A < 0$;

$7A^3 + 81A^2 - 10^6 < 0$

Контрольные вопросы.

Синтаксис оператора For ... Next

Синтаксис оператора Do ….Loop

Синтаксис оператора While ….Wend

Какие команды служат для принудительного выхода из циклов?

Какое значение имеет счетчик после выхода из цикла For ... Next?

ТЕМА 7. Одномерные массивы

Массивами в программировании называются совокупности данных одного типа, для хранения которых назначается одно имя переменной, а отдельные элементы из совокупности отличаются по их номеру. Поэтому массивы называют еще индексированными переменными. Кроме имени, индексов и хранимых в них значений массивы имеют еще две характеристики: размерность и количество элементов в массиве. Одномерные массивы, например A(i), где A - имя массива, i - номер элемента массива, используются для представления некоторых списков: номенклатура товара, его стоимость, координаты вектора и т.п. Двумерные массивы имеют два индекса, например AB(i, j), используются для представления таблиц, матриц, двумерных тензоров и т.п. В VBA возможно использование многомерных массивов. Число индексов может достигать 60.

По каждой из размерностей обычно указывается количество элементов, хранящихся в данном массиве. Например, для двумерного массива, представляющего таблицу, указывается число строк и столбцов в ней. Число строк и столбцов соответствует значениям первого и второго индексов.

Массивы, используемые в программе, должны быть до их применения объявлены в операторе Dim, например следующим образом;

```
Dim A(5) As Single
Dim AB(3, 4) As Integer
```

Здесь определено, что будет использоваться одномерный массив A с 5-ю элементами вещественного типа одинарной точности и двумерный массив AB с 20-ю элементами целого типа. Первый элемент массива A — A(0), массива AB — AB(0,0). При этом считается, что 0 — базовый индекс. Базовый индекс можно изменить, написав в начале листа модуля инструкцию Option Base 1. После этого индексы массивов A и AB будут начинаться с единицы. Другим способом изменения базового индекса является использование ключевого слова To:

```
Dim A(1to б) As Single
```

```
Dim AB(1 to 3, 1 to 4) As Integer
```

После объявления элементы массива могут использоваться в выражениях подобно простым переменным, но с указанием индекса в круглых скобках: A(4), AB(3,2).

Поскольку массивы в программе обрабатываются как единое целое, то доступ к их отдельным элементам, как правило, достигается в циклах. Ввод элементов массива может производиться с помощью оператора присваивания:

```
Dim Mas(1 to 5) As Integer
Mas(1)= 150
Mas(2)= 250
Mas(3)= 10
Mas(4)= 350
Mas(5)= 50
```

Ввод элементов массива может производиться в режиме диалога. Например, следующий код осуществляет ввод одномерного массива:

```
Dim Ab(1 to 15) As Single
For i=1 To 15
AB(i) =CSng(InputBox("Введите очередной элемент массива"))
Next i
```

Для ввода значений в одномерный массив удобно использовать функцию Array(). Эта функция задает список значений в соответствии каждому элементу объявленного массива. Синтаксис этой функции требует объявления массива через переменную типа Variant. Например,

```
Dim My_Mas As Variant
My_Mas= Array(150, 110,140,430,120)
```

По умолчанию базовый индекс равен нулю, поэтому первым элементом массива будет My_Mas(0)=150, последним My_Mas(4)=120.

В системе программирования VBA можно использовать и динамические массивы, когда количество элементов в массиве может меняться по ходу выполнения программы. В этом случае массив объявляется как динамический:

```
Dim Mas() As Byte
```

Затем в программе необходимо вычислить размер массива, присвоив это значение некоторой переменной, например Razmer, затем

указать размер динамического массива с помощью инструкции ReDim:

```
ReDim Mas(Razmer) или ReDim Mas(1 To Razmer)
```

Пример 1. В примере реализованы три варианта поиска по образцу с проверкой условия в начале цикла, в конце цикла и в середине цикла для варианта поиска по образцу с барьером:

```
Public Sub Loop1()
Const Size = 5
Dim X() As Integer
Dim i As Integer
Dim Found As Boolean
Const pat = 7
'Инициализация случайными числами в интервале [1 - 10]
ReDim X(1 To Size)
Randomize
For i = 1 To Size
    X(i) = Int(11 * Rnd)
Next i
'Поиск по образцу с проверкой в начале цикла
i = 1: Found = False
Do While (i <= Size) And (Not Found)
    If X(i) = pat Then
        Found = True
    Else
        i = i + 1
    End If
Loop
If Found Then
    Debug.Print "Найден образец!"
Else: Debug.Print "Образец не найден!"
End If
'Поиск по образцу с проверкой в конце цикла
i = 1: Found = False
Do
    If X(i) = pat Then
        Found = True
    Else: i = i + 1
    End If
```

```
Loop Until Found Or (i = Size + 1)
If Found Then
    Debug.Print "Найден образец!"
Else: Debug.Print "Образец не найден!"
End If
'Поиск с барьером
ReDim Preserve X(1 To Size + 1)
X(Size + 1) = pat
i = 1
Do
    If X(i) = pat Then Exit Do
    i = i + 1
Loop
If i = Size + 1 Then
Debug.Print "Образец не найден!"
Else: Debug.Print "Образец найден!"
End If
End Sub
```

Для обработки массивов используется цикл For Each...Next. Этот цикл повторяет заданную последовательность операторов для каждого элемента массива или набора.

Синтаксис оператора:

For Each элемент In группа
тело цикла
Next [элемент]

Здесь элемент - переменная, которая принимает в качестве значений элементы коллекций или массива. Для коллекций элемент может быть переменной типа Variant, переменной типа Object или переменной (объектом) некоторого класса. В случае цикла по массиву элемент обязан быть переменной типа Variant. Группа - это имя набора объектов (чаще всего это коллекция объектов) или массива, для элементов которых выполняется цикл. Цикл не применим для массивов, тип элементов которых определен пользователем, так как такие элементы не могут быть значениями переменной типа Variant. Тело цикла - последовательность операторов, выполняемая для каждого элемента набора или массива, - может содержать операторы Exit For, позволяющие прервать выполнение цикла и передать управление оператору, следующему за Next (обычно такой вы-

ход происходит при выполнении некоторого условия, проверяемого в операторе If…Then…Else). Указывать переменную элемент после ключевого слова Next не обязательно, но желательно.

Пример 2. Требуется создать программный код, в котором находится сумма элементов массива.

```
Sub Summa_N()
Dim b As Variant, Mass as Variant, S as Integer
Mass = Array(150,55,12,485,77,65,602)
S = 0
For Each b In Mass
S = S + b
Next b
Msg Box “Сумма элементов массива = ” & CStr(S)
End Sub
```

Задание 1. Набрать следующие программные коды, к каждой инструкции написать комментарий. Определить назначение программ. Изменить текст программного кода, вводя размер массивов с клавиатуры.

```
a)Dim z(10) As Byte
For i = 0 to 9
z(i) = Cbyte(InputBox(“Введите очередной элемент массива”))
Debug.Print “z(”; I;”)=”; z(i)
Next i
b)Dim D() As Single
Dim Maks As Single, n as Integer, i as Integer, nn as Byte
nn = CByte(InputBox (“Введите размер массива”))
ReDim D(nn)
For i = 0 to (nn-1)
D(i) = CSng(InputBox (“Введите очередной элемент массива”))
Debug.Print “D(“; i; “)=”; D(i)
Next i
Maks = D(0)
n = 1
For i = 1 to nn-1
If D(i) > Maks Then
Maks = D(i)
n=i
End If
```

Next i

Debug.Print “Максимальное значение = ”, Maks, “элемент с номером ”, n

Задание 2. В соответствии с заданным вариантом (вариант задания представлен в таблице 5) создать и выполнить программу ввода в память исходного массива, массив должен быть введен с использованием функции Array. Программа должна выполнять вычисление суммы значений элементов массива, произведения значений элементов массива, определение максимального и минимального значения элементов массива и их номера.

Таблица 5

NN п/п	Формула для вычислений	Исходные данные
1.	$h = d + \sum_{i=1}^{n} a_i^2$	n = 6; d = $12.55*10^{-4}$; a = {0.8; 4; -0.17; 2; 0.9; 1}
2.	$g = c* \sum_{j=0}^{4} (b_j + 1)^2$	m = 4; c = -0.0045; b = {0.49; 0.55; -2; -0.71}
3.	$z = \sum_{i=1}^{m} 0.1* x_j + \sum_{i=1}^{m} (x_i - 2)^2$	m = 5; x = {-2.1; 0; 1.1; 2.7; 4.3}
4.	$f = k!$ $g = f/\sum_{i=1}^{5} (a_i / i)$	k = 5; a = {2.3; 7; -7.2; -4; 2; 9}
5.	$w = \begin{cases} a + v_i, \sum_{i=1}^{5} v_i > 0 \\ b / v_i, \sum_{i=1}^{5} v_i \le 0 \end{cases}$	a = 1.75; b = -4.15; i = 1(1)5; v = {1; 1.5; -4; -12.9; 3}
6.	$q = \sum_{i=1}^{n} (x_i * y_i)$	n = 5; x = {-2.7; -5; 4; 3.5; -7.7} y = {2; 3; -1.5; -2.1}
7.	$c_i = \sqrt{a_i + b_i}$	n = 6; a = {0.5; 2; 2.5; 1; 0; 0.7}

	$d = \sum_{i=1}^{n} c_i \,/\, i$	b = {2.3; 4; 0.5; 2; 3; 9}
8.	$d = \sum_{i=1}^{n} a_i - c * \sum_{i=1}^{m} (b_i - 1)^2$	n = 6; m = 5; a = {0.8; 12; -4; 39; 2; 3} b = {19; 1; -24; -4.2; 8}
9.	$a_i = (d * e^{-x_i} * \sin x_i)/(\sqrt{5 + \cos x_i})$	$d = 51.05 * 10^{-4}$; i = 1(1)6; x = {0.7; 6; -7; 0.9; -0.2; 1}
10.	$Z = \begin{cases} \sin^2 c, \sum_{i=1}^{6} y_i > c \\ \cos^2 c, \sum_{i=1}^{6} y_i \le c \end{cases}$	y = {4; -6; 3; -3; 9; 11} c = 10.1
11.	$q = 1 + \frac{x_1 + 1}{x_1} + \frac{x_2 + 1}{x_2} + \ldots + \frac{x_6 + 1}{x_6}$	x = {1.1; 6.2; 3; -4; 6; 1}

Задание 3. Создать программный код для ввода массива случайных чисел, значения которых берутся из диапазона [-50; 50). Количество элементов массива вводить с клавиатуры. Определить номер первого и последнего положительного элемента.

Задание 4. Создать программный код для ввода массива случайных чисел, значения которых берутся из диапазона [-40; 35). Количество элементов массива вводить с клавиатуры. Определить среднее арифметическое положительных и отрицательных элементов.

Контрольные вопросы.

Какие данные можно рассматривать как массив?

Как определяется массив переменных в программе?

Как занести в память значения для элементов массива?

Как записываются значения элементов массива в расчетных формулах?

Как определить количество элементов в массиве?

Как задать нумерацию элементов массива в нужном интервале?

Как найти сумму элементов массива?

Как найти произведение элементов массива?
Как найти минимальный элемент массива?
Как найти максимальный элемент массива?

ТЕМА 8. Многомерные массивы

Часто встречаются массивы данных, которые имеют не одну, а несколько характеристик, в соответствии с которыми эти данные могут изменяться. Если таких характеристик две, то данные обычно записывают в виде таблиц или матриц, а каждая отдельная запись имеет свои номера соответствующей строки и столбца. Это находит свое отражение в способе записи элементов таких массивов — например A(i, j), где i — номер строки, j — номер столбца. Такие массивы называются двумерными.

Если представить себе многослойную таблицу или матрицу, то для идентификации конкретной записи, кроме номера строки и столбца, неизбежно появляется третий индекс — номер слоя. Такие массивы называются трехмерными и обозначаются — C(i, j, k). Двумерные массивы объявляются операторам Dim аналогично одномерным массивам. При этом указываются через запятую максимальные значения (либо с помощью ключевого слова To интервал значений) для каждого индекса в отдельности. Например, строки

Dim A(4,5) As Integer

объявляют двумерный массив для хранения числовых данных целого типа. Этот массив будет содержать 30 элементов (5 строк с номерами 0, 1, 2, 3, 4 и 6 столбцов с номерами 0, 1, 2, 3, 4, 5).

Для обработки двумерных массивов чаще всего используются два вложенных друг в друга цикла For ... Next, каждый их которых позволяет перебирать элементы в массиве по соответствующему индексу строки или столбца. При этом для вывода двумерного массива в виде таблицы или матрицы необходимо использовать оператор Debug.Print внутри каждого цикла.

Пример 1. В примере три вложенных цикла For … Next использованы для вычисления произведения двух целочисленных матриц, инициализированных случайными числами. Затем результирующая матрица проверяется на наличие в ней нулевого значения:

```
Public Sub For1()
Dim A(1 To 5, 1 To 5) As Integer
Dim B(1 To 5, 1 To 5) As Integer
```

```
    Dim C(1 To 5, 1 To 5) As Integer
    Dim I As Integer, J As Integer, K As Integer
    Dim Res As String
    'Инициализация матриц A и B случайными числами в интервале [-10, +10]
    Randomize
    For I = 1 To 5
        For J = 1 To 5
            A(I, J) = Int(21 * Rnd) - 10
        Next J
    Next I
    For I = 1 To 5
        For J = 1 To 5
            B(I, J) = Int(21 * Rnd) - 10
        Next J
    Next I
    'Вычисление произведения матриц
    For I = 1 To 5
        For J = 1 To 5
            C(I, J) = 0
            For K = 1 To 5
                C(I, J) = C(I, J) + A(I, K) * B(K, J)
            Next K
        Next J
    Next I
    Res = "No"
    C(2, 2) = 0
    'Проверка на нулевое значение
    For I = 1 To 5
        For J = 1 To 5
            If C(I, J) = 0 Then
                Debug.Print "Индексы: ", I, J
                Res = "Yes"
                Exit For
            End If
        Next J
    Next I
    Debug.Print Res
```

```
End Sub
```

Оператор выхода Exit For прекращает выполнение только внутреннего цикла, так что проверка на ноль будет осуществляться в каждой строке матрицы, независимо от существования нулей в предыдущих строках. По окончании цикла счетчик цикла сохраняет свое значение в момент выхода и его можно использовать, например, для анализа преждевременного выхода из цикла. В операторе Next можно не указывать имя переменной, задающей счетчик, это имя подразумевается по умолчанию, как завершающее последний открытый оператор For. Рекомендуется всегда явно указывать имя счетчика в операторе Next.

Для ускорения поиска данных в больших массивах используется метод упорядочивания данных – сортировка. В отсортированных массивах поиск происходит значительно быстрее. Разработаны разные способы сортировки массивов, которые зависят от объемов используемой оперативной памяти компьютера. Чем мощнее сортировка, тем больше памяти она требует. Для понимания сущности сортировки часто используется метод пузырька. Алгоритм этого метода состоит из повторных проходов по элементам массива с попарным их сравнением. В зависимости от условий сортировки (по возрастанию или по убыванию) вперед ставится или меньший элемент (по возрастанию), или больший (по убыванию).

Задание 1. Набрать следующие программные коды. К каждой инструкции написать комментарий. Определить назначение программ. Изменить текст программного кода, вводя размер массивов с клавиатуры. Запятая в конце оператора Debug.Print служит для того, чтобы печать каждого следующего числа в цикле по индексу j осуществлялась на одной строке. Оператор Debug.Print без параметров переводит курсор в начало следующей строки перед выполнением нового цикла по индексу i.

```
a)Dim A(1 to 2, 1 to 3) As Single, B(1 to 3, 1 to 2) As Single
Dim i As Integer, j As Integer
A(1,1)=1: A(1,2)=0.4: A(1,3)=0
A(2,1)=—4.1: A(2,2)=2.2: A(2,3)=1.7
For i=1 To 2
    For j=1 To 3
        Debug.Print A(i,j);
    Next j
```

```
Debug.Print
Next i
For i=1 To 2
    For j=1 To 3
        B(j, i) = A(i, j)
    Next j
Next i
For i=1 To 3
    For j=1 To 2
        Debug.Print B(i,j);
    Next j
    Debug.Print
Next i
b)Dim a(1 To 2, 1 To 3) As Single
Dim b(1 To 3, 1 To 2) As Single
Dim c(1 To 2, 1 To 2) As Single
Dim i As Integer, j As Integer, k As Integer
a(1, 1) = 1: a(1, 2) = 0.4: a(1, 3) = 0
a(2, 1) = -4.1: a(2, 2) = 2.2: a(2, 3) = 1.7
For i = 1 To 2
    For j = 1 To 3
        b(j, i) = a(i, j)
Next j: Next i
For i = 1 To 2
    For j = 1 To 2
        For k = 1 To 3
            c(i, j) = c(i, j) + a(i, k) * b(k, j)
Next k: Next j: Next i
For i = 1 To 2
    For j = 1 To 2
        Debug.Print c(i, j);
    Next j: Debug.Print: Next i
```

Задание 2. Объявить матрицу, выбранную в соответствии с заданным вариантом, доступной для всех процедур модуля.

Написать программный код, который выводит матрицу в отформатированном виде в окно Immediate Window.

Написать программный код транспонирования исходной матрицы (перемена местами строк и столбцов матрицы) и вывести ее в окно Immediate Window.

Написать программный код по условию выбранной в соответствии с вариантом задачи (вариант задания представлен в таблице 6).

Таблица 6.

NN пп	Формула для вычислений	Исходные данные
1	Для транспонированной матрицы вычислить сумму элементов первого ряда	2.1 2.4 2.3 3.5 6.2 8.3 3.5 7.5 9.1
2	Для транспонированной матрицы вычислить сумму элементов второго столбца	3.5 4.6 7.3 8.1 -7.4 1.2 6.5 6.0 2.4
3	Определить максимальный элемент для каждой строки	4.1 2 0.3 7.4 7.3 –1 -2.5 6 6.1 -10 7.4 7.6
4	Определить минимальный элемент для каждого столбца	12.4 -4.1 0.6 10 0.5 9.9 -3.5 1.5
5	Определить произведение всех элементов матрицы, значение которых положительны	2.4 3 0.3 -5.1 4.2 -1.9
6	Определить минимальную сумму строки	3.5 3.7 9.1 2.8 11.5 2.7 4.7 7.2 1.9
7	Определить максимальное произведение элементов столбца	-5.1 4.1 -1.99 1 0.75 1.1 -1.9 -0.9 -10

8	Для элементов матрицы, которые меньше или равны двум вычислить произведение.	-5.1 4.1 -1.99 1 0.75 1.1 -1.9 -0.9 -10
9	Для элементов матрицы, которые больше единицы, вычислить сумму обратных величин	2.5 -1 0.455 1.91 2.52 -10 -4.4 3.33 0.947

Задание 3. При помощи датчика случайных чисел заполнить массив. Размерность массива ввести с клавиатуры., вывести в отладочное окно. Отсортировать массив в порядке возрастания и вывести в отладочное окно.

Контрольные вопросы

Какие массивы данных можно представить как двумерный массив?

Как определяется общее количество элементов двумерного массива?

Правила использования вложенных циклов.

Как осуществляется ввод двумерных массивов?

Как вывести двумерный числовой массив в виде матрицы?

Как запрограммировать вычисление суммы элементов матрицы?

Как запрограммировать вычисление суммы элементов в столбцах матрицы?

Как запрограммировать вычисление произведения элементов матрицы?

Как запрограммировать вычисление произведения матриц?

Как запрограммировать вычисление транспонированной матрицы?

ТЕМА 9. Работа с символьной информацией

При решении информационно-поисковых задач, в экономических задачах наряду с вычислительными операциями широко используется обработка символьных данных.

Средства обработки символьной информации представлены операцией конкатенации, операцией сравнения Like, специальными

функциями, которые можно разделить на группы, представленные в таблице 7.

Задание 1. При помощи датчика случайных чисел ввести последовательность {ai}, состоящую из N(случайное двузначное число) целых положительных случайных 4-хзначных чисел. Из этой последовательности получить последовательность {bi}, переставив в

Таблица 7.

1.	Преобразование кодов	Chr(), Asc() и др.
2.	Выделение определенной части символьного выражения	Left(), Mid(), Right()
3.	Замена части символьного выражения	Mid(), LTrim(), RTrim()
4.	Формирование строк одинаковых символов	String(), Space()
5.	Обработка числовой информации в тексте	Val(), Str()
6.	Перевод из одной системы счисления в другую	Oct(), Hex()
7.	Определение количественных характеристик символьных строк	Len(), InStr()

элементах {ai}последовательности вторую и четвертую цифры. Обе последовательности вывести в соседние столбцы отладочного окна.

Пояснение. Пусть дано число 1234. Надо поменять местами вторую и четвертую цифры числа и получить новое число 1432. Эту операцию удобно провести не над числом, а над соответствующей строкой символов ”1234”, используя символьную функцию Mid().

Встроенная функция Mid(a, b, c) имеет следующие аргументы:

a – строка, которую обрабатывает функция Mid;

b – номер позиции в строке a, с которого будут выделяться символы;

c – число, которое показывает, сколько символов должно быть выделено в строке a.

Функция Mid(a,b,c) возвращает подстроку, которая состоит из **c** символов, выделенных из строки **a**, начиная с позиции b.

Алгоритм решения задачи следующий:

1. Вырезать из строки S = ”1234” второй символ и сохраним его в переменной t: t = Mid(S, 2, 1);

2. во вторую позицию этой строки надо записать содержимое четвертой позиции: Mid(S, 2, 1) = Mid(S, 4, 1)

3. в четвертую позицию надо записать сохраненное в переменной t значение второго символа.

Оформить этот алгоритм в отдельную функцию. Вызов функции осуществлять из головной процедуры.

Задание 2. Вычислить сумму цифр во вводимом числе. Использовать функции Mid(a,b,c) и Len(a) – возвращает длину строки **a**.

Задание 3. Определить, есть ли среди цифр трехзначного числа две одинаковые цифры.

Задание 4. Найти все четырехзначные числа, в которых сумма цифр является однозначным числом. Все числа вывести в отладочное окно, а число таких чисел вывести с помощью MsgBox.

Задание 5. Найти все четырехзначные числа, у которых сумма крайних цифр равна сумме средних цифр, а само число делится на 4 и на 22. Все числа вывести в отладочное окно, а число таких чисел вывести с помощью MsgBox.

Задание 6. Найти все четырехзначные числа, в которых только две одинаковые цифры. Все числа вывести в отладочное окно, а число таких чисел вывести с помощью MsgBox.

Задание 7. Вывести наибольшую из первых цифр трех введенных чисел. Числа вырабатываются датчиком случайных чисел, числа целые, их значения из диапазона от ноля до тысячи.

Контрольные вопросы

Назначение и работа функции Mid().

Назначение и работа функции InStr().

Продемонстрировать работу функции Asc() в отладочном окне.

Продемонстрировать работу функции Chr() в отладочном окне.

Продемонстрировать работу функции Len() в отладочном окне.

Продемонстрировать работу функции Chr() в отладочном окне.

ТЕМА 10. **Создание пользовательских форм**

VBA позволяет создавать свои собственные диалоговые окна, используя пользовательские формы UserForm. Для создания формы необходимо войти в редактор Visial Basic Editor и выполнить команду Insert/UserForm. При этом будет создана пустая форма с именем UserForm1 (рис. 14). По умолчанию формы имеют имена UserForm1, UserForm2, и т. д.

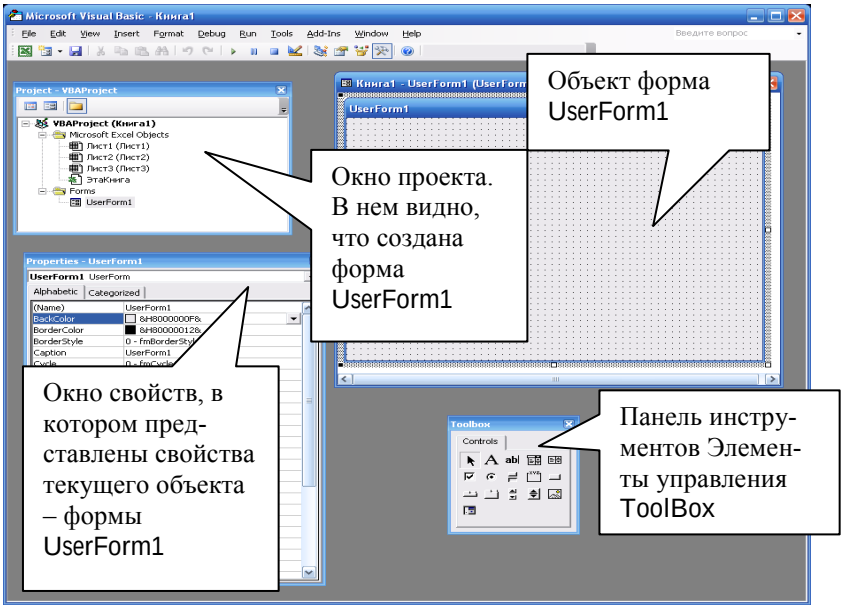


Рис. 14. Вид окна редактора VBE при создании формы

Пользовательские формы и элементы управления являются объектами. Поэтому они обладают различными свойствами (характеристиками), набором методов (набором действий, которые может выполнять каждый из объектов) и событий, то есть действий, распознаваемых объектом на которые можно запрограммировать отклик.

В окне свойств Properties задаются свойства самой формы или выбранного на форме элемента управления. Для задания свойств надо выделить форму или элемент управления, щелкнув по ним левой кнопкой мыши. На экране появится окно Properties (рис. 14), относящееся к выбранному объекту. Присваивать или изменять значения свойств можно программно или в окне свойств. Про-

граммный способ имеет больший приоритет. Такими свойствами могут быть заголовок выбранного объекта (Caption), его имя (Name), цвет фона (BackColor), шрифт (Font), всплывающие подсказки (ControlTipText) и другие.

Панель инструментов ToolBox обычно содержит следующие значки: Выбор объектов, Надпись, Поле, Поле со списком, Список, Флажок, Переключатель, Выключатель, Рамка, Кнопка, Набор вкладок, Набор страниц, Полоса прокрутки, Счетчик, Рисунок, Редактор ссылок. Набор элементов на панели инструментов ToolBox может быть при необходимости изменен. Одни элементы могут быть удалены, а другие добавлены. Для добавления элемента управления на панель ToolBox необходимо выполнить команду **Tools/Additional Controls….**

В окне проекта Project содержится список всех файлов, необходимых для выполнения создаваемого проекта приложения.

Имя пользовательской формы используется в программных кодах приложения. При задании имени формы, отличного от имени, задаваемого по умолчанию, используется свойство Name. Этому свойству присваивается задаваемое имя. Имя формы должно начинаться с префикса frm, например, frmSpisok. заголовка формы достигается изменением значения свойства Caption в окне Properties формы или в программе.

Для удобства работы при размещении элементов управления на форме имеется разметка в виде точек. Используя контекстное меню, можно выравнивать размеры и положение элементов управления на форме. Для получения одинаковых по размеру элементов управления их можно копировать и вставлять в нужное место.

Создание проекта приложения состоит из двух взаимосвязанных процессов:

- размещение на очередной форме элементов управления и задания необходимых начальных значений этим элементам управления в окне свойств или в программах инициализации формы;
- написания текста программы (программного кода) в соответствии с алгоритмом решаемой задачи в окне кода формы или стандартного модуля.

Важнейшим событием, которое может происходить с формой, является инициализация. Инициализация формы - сборка формы в соответствии с заданными параметрами и с параметрами, принятыми по умолчанию. Событие инициализация происходит перед появлением формы на экране первый раз. Для организации программы инициализации формы надо дважды щелкнуть по форме мышкой. Появится лист стандартного модуля (рис. 15), в котором необходимо подготовить процедуру для написания программного кода.

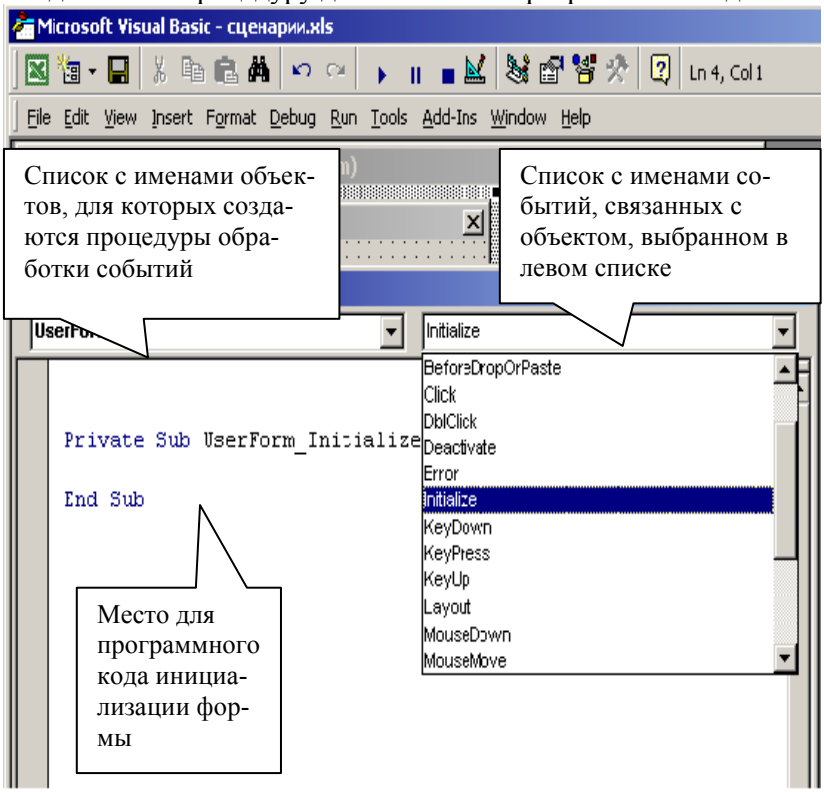


Рис. 15. Модуль программного кода инициализации формы

Для этого надо выбрать из левого раскрывающегося списка модуля имя формы, а из правого раскрывающегося списка - событие Initialize. На листе модуля появится заголовок процедуры, сформированный редактором VBA:

```
Private Sub UserForm_Initialize()
```

End Sub

Форма может быть активизирована из среды VBA. Для этого:

- установить курсор в области процедуры, выводящей форму на экран, или сделать активным окно с необходимой формой;
- выполнить команду Run/Run или нажать клавишу F5.

Назначить форме процедуру, которая должна будет выводить на экран форму для работы с ней, можно, написав в стандартном модуле следующий код.

Sub Задача ()
имяФормы. Show
End Sub

Эта процедура использует метод .Show (показать) для вывода формы на экран.

Убрать форму с экрана и очистить от нее оперативную память компьютера можно, используя инструкцию UnLoad *имя формы*. Вместо *имя формы* удобно использовать ключевое слово Me, означающее имя текущего объекта.

Задание 1. Войти в редактор VBE. Создать пользовательскую форму. Создать заголовок формы — Пример 1. Задать имя формы – – frmPervaja в окне свойств Properties. Активизировать форму. Написать программу инициализации формы. В теле процедуры инициализации формы набрать код, который задает новый заголовок формы - ПЕРВАЯ ФОРМА, задает цвет фона формы с помощью встроенной переменной vbWhite. Активизировать форму.

Задание 2. В созданной форме разместить одно текстовое поле и три командные кнопки. По щелчку на каждой кнопке в текстовое поле поочередно должен выводиться текст, соответствующий нажатой кнопке. Щелчок по первой командной кнопке должен вызывать появление в текстовом поле фразы “Изучайте VBA!!!”. Щелчок по второй командной кнопке должен вызывать появление в текстовом поле фразы “Show – это метод формы, отображающий форму на экране”. Щелчок по третьей командной кнопке должен вызывать появление в текстовом поле определение события. Текстовое поле размещается на форме с использованием пиктограммы 

. Ко-

мандные кнопки (CommandButton) можно поместить на форме, рисуя каждую кнопку, а можно нарисовать одну, скопировать ее и вставить две точные копии первой. Задать надписи на командных кнопках. На первой кнопке – “Первая строка”, на второй – “Вторая строка”, на третьей – “Третья строка”.

Выполнение задания заключается в редактировании программы инициализации формы, созданной в задании 1, и написании программ обработки событий – щелчок по первой командной кнопке, щелчок по второй и щелчок по третьей командным кнопкам.

В программу инициализации формы должна быть внесена возможность вызова всплывающей подсказки к первой кнопке. Программы обработки щелчков по кнопкам должны вызывать присвоение текстовому полю (его свойству .Text) соответствующих вводимых значений.

Задание 3. Создать пользовательскую форму “Ввод числовых данных”. Разместить в ней текстовое окно и командную кнопку ”Ввод”. Вводимое в текстовое окно вещественное число надо вывести в диалоговое окно MsgBox с точностью до двух знаков, в отладочное окно – с точностью до четырех знаков.

Задание 4. Написать процедуры для вывода на экран разработанных в заданиях 1 и 2 форм для работы с ними.

Контрольные вопросы

Перечислить основные свойства пользовательской формы.

Как задать имя пользовательской форме?

Что означает событие инициализация формы?

Где записывается код инициализации формы?

Использование какого метода выводит форму на экран?

Как убрать форму с экрана и очистить от нее оперативную память?

ТЕМА 11. Использование элементов управления в пользовательских формах.

Элементы управления – это объекты, которые содержатся в объектах-формах. Каждый тип элемента управления имеет свой собственный набор свойств, методов и событий, которые делают его наиболее подходящим для определенной цели. Некоторые элементы

управления лучше всего использовать для ввода или отображения текста, другие для доступа к приложениям и данным.

Ниже приведен перечень основных элементов управления, а на рис. 16 – значки различных элементов управления на панели элементов.

- CheckBox - флажок;
- ComboBox - комбинированный список (поле со списком);
- CommandButton - командная кнопка;
- Image - изображение (окно изображения);
- Label - метка (надпись, статический текст);
- ListBox - список (окно списка);
- MultiPage - набор страниц;
- OptionButton - переключатель (кнопка зависимого выбора);
- ScrollBar - полоса прокрутки;
- SpinButton - счетчик (ворот);
- TabStrip - полоса вкладок;
- TextBox - поле ввода (окно редактирования, текстовое поле);
- ToggleButton - выключатель.

Элементы управления, которые можно использовать при создании пользовательских форм располагаются на панели инструментов ToolBox. Эта панель появляется на экране при активизации формы, т.е. при ее выделении. Элементы управления (э.у.), расположенные на панели ToolBox, различаются в программе по именам.

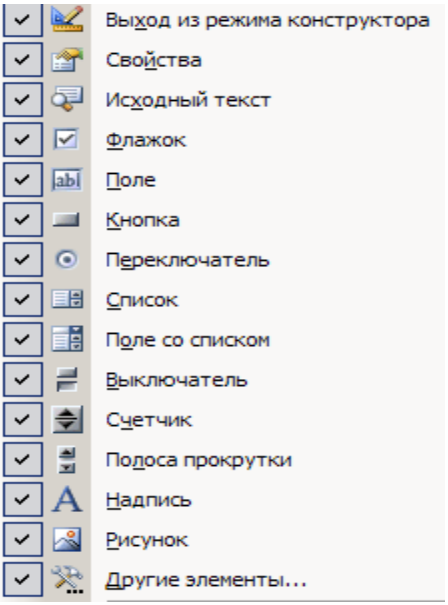


Рис. 16. Панель элементов управления

Имена элементов управления, задаваемые пользователем, начинаются с определенных префиксов. Например, имя элемента управления Label (Надпись) должно начинаться с префикса lbl, элемента управления TextBox(Текстовое поле) с префикса txt, элемента управления CommandButton(Командная кнопка) – cmd.

Когда пользователю необходимо перевести курсор (фокус) на другой элемент управления, для элемента управления, на который переводится курсор, возникает событие *Получение фокуса*. А для элемента, с которого переведен курсор возникает событие *Потеря фокуса*. Поэтому к объекту, на который переводится курсор, можно применить метод SetFocus. Метод SetFocus объекта переводит курсор на этот объект. К объекту, с которого должен быть переведен курсор, можно применить метод LostFocus – потеря фокуса.

Оператор SendKeys “{Home}+{End}” позволяет выделить данные в активном элементе управления.

Для того, чтобы элементу управления в пользовательской форме назначить всплывающую подсказку, надо установить соответствующее значение свойству ControlTipText в окне свойств для данного

элемента управления. Доступность элемента управления задается свойством элемента управления Enabled = True (доступен) или Enabled = False (недоступен).

Создание пользовательской формы ДЕЛЕНИЕ

При создании формы необходимо использовать элементы управления: Label (Надпись), TextBox (Текстовое поле), CommandButton (Коммандная кнопка). Основным свойством э.у. Label является свойство Caption. Именно это свойство хранит задаваемое значение надписи. Основным свойством э.у. TextBox является свойство Text. Именно оно принимает и хранит значение, введенное в текстовое окно.

Задание 1. Создать диалоговое окно Деление, содержащее три элемента управления Надпись(Label), три элемента управления текстовое поле (TextBox) и две командные кнопки (CommandButton) (рис. 17). Задать элементам управления имена.

При нажатии на кнопку *Закончить работу* должен происходить выход из программы.

Заголовок формы и свойства элементов управления задать в программе инициализации формы. Третье текстовое окно не должно быть доступно пользователю. Для кнопки *Закончить работу* должна появляться всплывающая подсказка “Это выход из программы”.

По нажатию клавиши *Делить* данные из первого текстового окна должны проверяться на принадлежность к числовому типу данных. Если введено не число, то пользователю надо выдать сообщение об ошибке и в чем она заключается, перевести курсор в первое текстовое окно, выделить неверные данные и выйти из программы. Данные из второго текстового окна проверяются на принадлежность к числовому типу данных и проверяются на равенство нулю. Если введен ноль, то пользователю надо выдать соответствующее сообщение об ошибке, перевести курсор во второе текстовое окно, выделить неверные данные и выйти из программы. Если все данные введены верно, то в третье текстовое окно надо вывести результат деления с тремя знаками после десятичной точки.

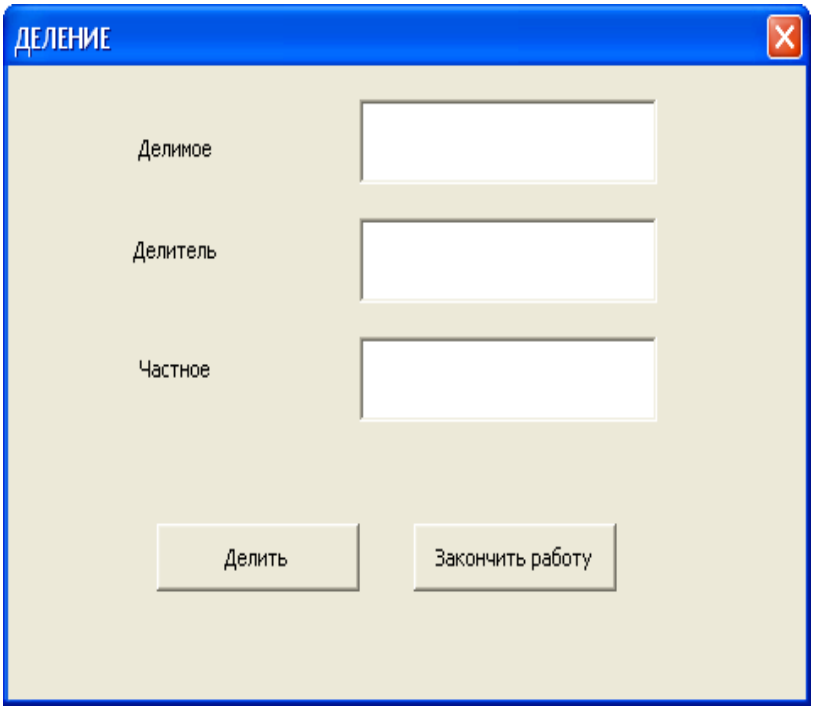


Рис. 17. Вид формы Деление

Задание 2. Написать программы обработки событий *Вход* в первое и второе текстовые поля и *Выход* из них. При вводе данных в первое и второе окно цвет поля должен становиться желтым. При выходе из поля цвет должен меняться на белый.

Задание 3. Создать процедуру, запускающую форму ДЕЛЕНИЕ.

Создание пользовательской формы СЛОЖЕНИЕ

При создании формы необходимо использовать элементы управления: Label (Надпись), TextBox (Текстовое поле), CommandButton (Командная кнопка), Frame (Рамка), OptionButton (Переключатель). Основным свойством э.у. Frame является свойство Caption. Это свойство хранит заголовок рамки. Основным свойством элемнта управления OptionButton является свойство Value. Оно принимает значение True при включении э.у. и принимает значение False при

выключении э.у. Для создания фона формы и элементов управления используется свойство BackColor формы и соответствующего э.у. Если CommandButton.Default = True, то кнопка становится основной. Это значит, что можно использовать на клавиатуре кнопку Enter вместо основной командной кнопки. Если у элемента управления CommandButton свойство Cancel = True, то можно использовать на клавиатуре кнопку Esc.

Задание 1. Создать форму, которая в первый раз предстает перед пользователем в виде, представленном на рисунке 18.

Задание 2. Создать программу инициализации формы, в которой задать:

- заголовок формы;
- включение переключатель «Сложение»;
- недоступность пользователю текстового окна «Сумма»;
- всплывающие подсказки к кнопкам «ОК» и «Cancel». Всплывающие подсказки должны быть соответственно: «Можно использовать клавишу Enter», «Можно использовать клавишу Escape». Организовать возможность использования на клавиатуре клавиш Enter и Escape;
- фон формы и надписей одинаковым цветом;
- шрифт Times New Roman, его размеры и начертание как это показано на рисунке 18.

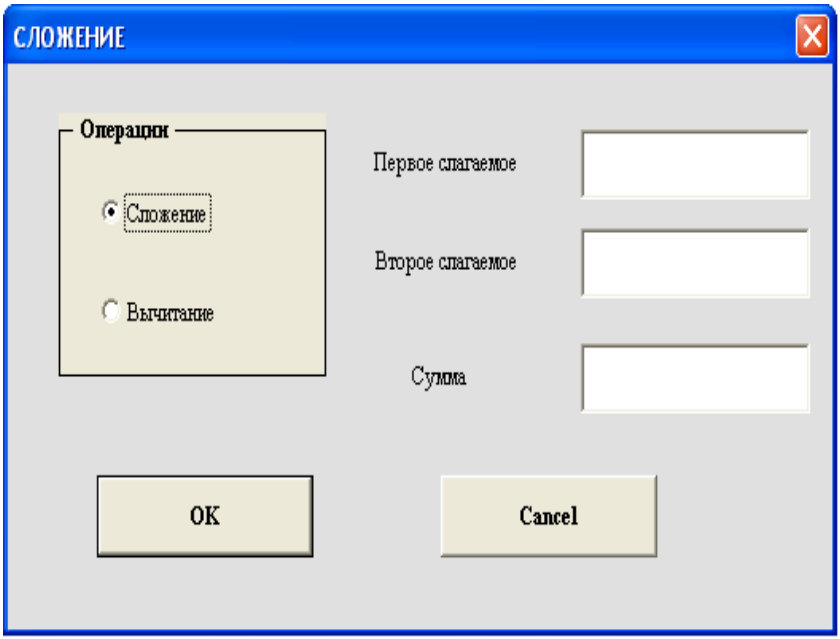


Рис. 18. Вид формы Сложение

Задание 3. Написать программу обработки события – нажатие на кнопку «Cancel».

Задание 4..Написать программу обработки события – нажатие на кнопку «ОК». При вводе использовать действительные числа. Результат выдавать с тремя знаками после запятой.

Задание 5.. Написать программу обработки события – нажатие на переключатель «Вычитание». Предусмотреть при этом изменение заголовка формы и надписей к текстовым окнам. Заголовок должен стать «Вычитание», надписи соответственно «Уменьшаемое», «Вычитаемое», «Разность».

Создание пользовательской формы РЕНТА

При создании формы необходимо использовать элементы управления: Label (Надпись), TextBox (Текстовое поле), CommandButton (Командная кнопка), CheckBox(Флажок). Основным свойством элемнта управления CheckBox является свойство Value. Оно при-

нимает значение True при включении флажка. и принимает значение False при его выключении.

Задание 1. Создать форму, которая в первый раз предстает перед пользователем в виде, представленном на рисунке 19.

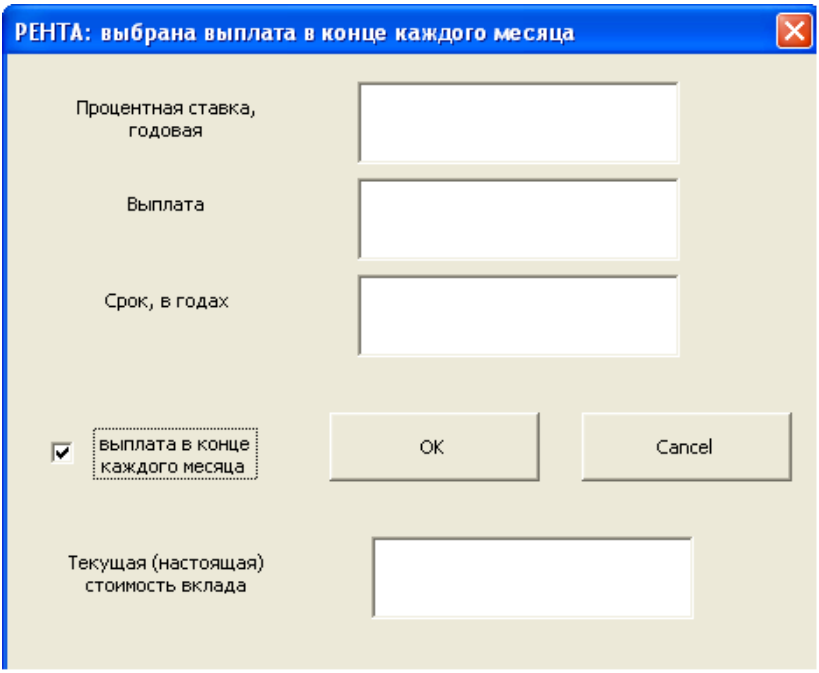


Рис. 19. Вид формы Рента

Задание 2. Создать программу инициализации формы, которая должна:

- задать недоступность пользователю текстового окна «Текущая (настоящая) стоимость вклада»;
- организовать всплывающие подсказки к кнопкам «OK» и «Cancel» соответственно: «Можно использовать клавишу Enter», «Можно использовать клавишу Escape». Должна быть организована возможность использовать эти клавиши на клавиатуре;
- задать фон формы и надписей одним цветом;

Задание 3. Написать программу обработки события – нажатие на кнопку «Cancel».

Задание 4. Написать программу обработки события – нажатие на кнопку «ОК». В программе используется встроенная финансовая функция PV(ставка; период; выплата; будущая стоимость; тип). Расчет проводить для включенного и выключенного флажка. Посмотреть справку по этой функции. Использовать действительные числа. Результат выдавать с двумя знаками после запятой.

Задание 5. Написать программу обработки события – изменение значения флажка. Предусмотреть, чтобы при этом изменился заголовок формы. Он должен стать «РЕНТА: выбрана выплата в начале каждого месяца».

Создание пользовательской формы Работа со списком

При создании формы необходимо использовать элементы управления: Label (Надпись), TextBox (Текстовое поле), CommandButton (Командная кнопка), ListBox(Список). Основными свойствами элемнта управления ListBox являются следующие свойства:

- MultiSelect задает возможность выбора нескольких элементов списка.
- ColumnCount задает количество столбцов в списке.
- ColumnWidths задает ширину столбцов в списке.
- List(i,j) хранит значение элемента, стоящего в списке в строке i, столбце j, или выводит задаваемое значение в соответствующий элемент списка.
- Selected(i) принимает значение True для выделенного элемента списка и значение False для невыделенного.

Метод .AddItem f добавляет значение f в i-ый столбец списка.

Работу с элементами списка удобно осуществлять в цикле.

Задание 1. Создать форму, которая в первый раз предстает перед пользователем в виде, представленном на рисунке 20.

Задание 2. В элемент управления Список должны выводиться 3 столбика. Значения 1-го столбца: целые числа, идущие по порядку от 0 до N; 2-го столбца; целые случайные числа из диапазона от K до M; 3-го столбца: целые случайные числа из диапазона от T до P. Надо выделять несколько строк списка и в окно “Результат” выдать:

1-й вариант. Сумму выделенных элементов из 2-го и 3-го ряда. N = 10, K = 20, M = 80, T = 10, P = 60.

2-й вариант. Сумму произведений элемента 2-го ряда на 3-го для выделенных строк. N = 50, K = 10, M = 90, T = 0, P = 100.

3-й вариант. Среднее значение выделенных элементов. N = 25, K = 9, M = 66, T = 30, P = 99.

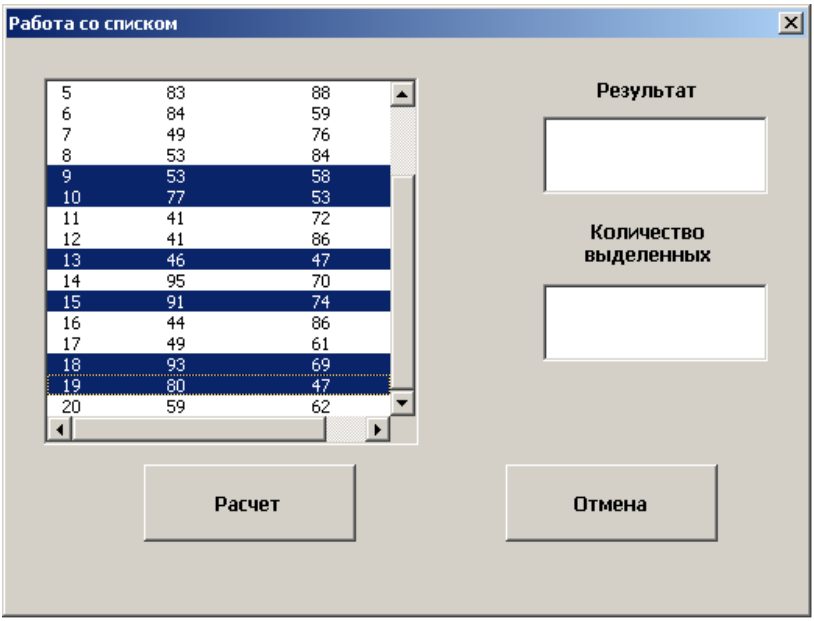


Рис. 20. Вид формы Работа со списком

4-й вариант. Абсолютное значение разности выделенных элементов 2-го и 3-го ряда. Уменьшаемое – 1-й выделенный элемент 2-го ряда. N = 55, K = 15, M = 110, T = 5, P = 55.

В окно “Количество выделенных элементов” выдать соответствующее число. Оформить подсказку к окну “Результат” с номером выполненного варианта.

Контрольные вопросы

Как задаются имена элементам управления?

Назначение оператора SendKeys.

Назначение свойств Default, Cancel э.у. Кнопка.

Какое свойство скрывает э.у.?

Перечислить основные финансовые функции, привести их синтаксис.

Как можно использовать свойство List элемента управления Список.

Как обеспечить выбор нескольких элементов из э.у. Список.

ТЕМА 12. **Объектная модель** MS Excel

В модели объектов Excel имеется более 100 объектов. Главная его иерархия выглчдит следующим образом: Application (приложение) — Workbook (рабочая книга) — Worksheet (рабочий лист) — Range (диапазон).

Объект Application представляет собой все приложение Excel. Обращение к этому объекту происходит при вызове Excel из других приложений (Word, Access и т.д.). В активном приложении Excel создавать объект Application не требуется.

Объект Workbook необходим для получения ссылки на нужную книгу в наборе открытых книг Excel, для настройки общих свойств и выполнения общих действий со всеми листами книги.

Объекты Worksheet в книге объединены в коллекцию Sheets (листы).

Для ввода данных в электронную таблицу требуется определиться с листом, на который будет выполняться ввод данных – либо выбрать его, либо вначале создать, а потом выбрать.

Пример создания нового листа может выглядеть следующим образом.

```
Dim oExcel As New Excel.Application        'Запуск Excel
oExcel.Visible = True                      'Делаем его видимым
Dim oWbk As Excel. Workbook
Set oWbk = oExcel. Workbooks. Add()        'Создание новой книги
Dim oSheet As oExcel. Worksheet
Set oSheet = oWbk. Worksheets.Add()        'Создание нового листа
oSheet.Name = "Новый лист"
```

Другой задачей является поиск нужного листа среди листов открытой книги. Например, это можно сделать следующим образом.

```
Dim oExcel As New Excel.Application
oExcel.Visible = True
Dim oWbk As Excel. Workbook
```

```
Set oWbk = oExcel. Workbooks. Add()
Dim oSheet As oExcel. Worksheet
Set oSheet = oWbk. Worksheets.Item(“Лист2)
oSheet.Name = ”Новый лист”
```

У коллекции Sheets есть уже известные свойства и методы коллекций VBA такие как Count, Item, Add(), Delete(). Для объекта Worksheetприменяются свойства и методы Visible(), Copy(), Move(), Select(). Для этой коллекции предусмотрен специфический метод FillAcrossSheets() – скопировать объект диапазона Range (полностью, только содержимое или только оформление) во все листы данной книги. Самым важным событием объекта Worksheet является событие Change. Существует много практических задач, когда изменение пользователем значения ячейки должно приводить к изменению значения в ячейке другого листа. Соответствующая событийная процедура работает со специальным параметром Target, представляющим собой изменившуюся ячейку. При помощи свойств и методов объекта Range можно получить информацию об изменившемся значении, столбце и строке, в котором произошли изменения.

У объекта Worksheet есть еще два очень удобных события – это BeforeRightClick() и BeforeDoubleClick(). Первое событие позволяет перехватывать щелчок правой клавиши мыши по любому месту на листе, а второе событие – двойной щелчок мышью. При помощи этих событий можно назначить свою реакцию (открытие контекстных меню, выдачу предупреждающих сообщений, переход в другой режим работы и т.п.) на действия пользователя.

Наиболее часто используемым объектом в иерархии объектной модели Excel является объект Range. Он может представлять одну ячейку, несколько ячеек (в том числе несмежные ячейки или наборы несмежных ячеек) или целый рабочий лист.

Для получения объекта Range можно воспользоваться свойством Cells объекта Worksheet. Это свойство возвращает диапазон, состоящий только из одной ячейки, зато можно использовать более удобный синтаксис.

Примеры получения объекта Range.

Чтобы получить ссылку на объект Range, представляющий ячейку A1, надо выполнить следующие команды.

```
Dim oRange As Range
```

```
Set oRange = Worksheets("Лист1").Range("A1")
```

Ссылка на диапазон ячеек с A1 по D10 создается так:

```
Dim oRange As Range
Set oRange = Worksheets("Лист1").Range("A1:D10")
```

Для получения ссылки на ячейку D1 можно использовать следующий код:

```
Dim oRange As Range
Set oRange = Worksheets("Лист1").Cells(1, 4)
```

Чтобы получить диапазон, состоящий из нескольких ячеек, удобно применять свойства Range и Cells вместе:

```
Dim oRange As Range
Set oRange = Range(Cells(1, 1), Cells(5, 3))
```

После того, как нужная ячейка найдена, в нее нужно что-то записать. Для этой цели используется свойство Value. Например, oRange.Value = 10589. У объекта Range очень много свойств и методов. Далее представлены некоторые наиболее часто употребимые свойства.

Таблица 8.

Address	позволяет вернуть адрес текущего диапазона в абсолютной адресации ($A$1:$F$24, $B$12). Свойство доступно только для чтения. На практике встречается много ситуаций, когда из адреса ячейки надо выделить номер столбца или номер строки. это можно сделать при помощи строковых функций. Имя столбца для объекта oRange, представляющего одну ячейку можно вернуть так: sColName=Mid(oRange.Address, 2, (InStr(2, oRange.Address, "$")-2)), а номер строки – так: sRName= Mid(oRange.Address, 2, (InStr(2, oRange. Address, "$")+1)).
Areas	позволяет разбить подобные нестандартные диапазоны на набор стандартных. Это свойство можно использовать и для проверки "нестандартности" диапазона
Cells	это свойство для объекта Range, использованное без индексов, возвращает все ячейки диапазона.

	Например, Range (“B2:D6”). Cells вернет диапазон “B2:D6”. Это свойство, использованное с индексами, возвращает ячейку, стоящую на пересечении строчки и столбца. Причем свойство Cells(i, j) задает относительное расположение ячейки по отношению к диапазону. Например, значение 10 вводится в ячейку C3: Range (“B2:D6”).Cells(2,2).Value=10 Если требуется задать абсолютное местоположение ячеек, то надо воспользоваться свойством Cells рабочего листа, Cells(5, 3).Value = 10
Count	Возвращает количество ячеек в диапазоне. Может использоваться для проверок.
CurrentRegion	возвращает текущий диапазон, т.е. максимальный диапазон, который окружен любой комбинацией пустых строк и столбцов. Ячейки текущего диапазона не пусты. Например, в диапазоне A1:E12 имеется таблица, которая по столбцу F и 13 строке окружена пустыми ячейками. Инструкция Cells(1, 1).CurrentRegion.Select вызовет выделение всей таблицы.
Formula	это свойство доступно и для чтения, и для записи. Возвращает текст формулы, прописанной в ячейку (а не вычисленное значение) или позволяет записать формулу в ячейку.
Interior	свойство связано с форматированием, позволяет выделить цветом ячейки диапазона.
Range	позволяет создать новый диапазон на основе существующего.
Resize	позволяет изменить текущий диапазон. Например, увеличение его на один столбец вниз и на одну строку вправо можно выполнить так: oRange.Resize(oRange.Rows.Count+1, oRange. Columns. Count+1).Select
ShrinkToFit	свойство позволяет автоматически настроить размер текст в диапазоне таким образом, чтобы текст помещался в ширину столбца.

Text	получает значение первой ячейки диапазона в виде значения типа String. Доступно только для чтения.
Value	наиболее часто используемое свойство объекта Range. Позволяет получить или назначить значение ячейкам диапазона.

Информация о методах объекта Range представлена в таблице 9.

Таблица 9.

Activate()	выделяет текущий диапазон и устанавливает курсор ввода на его первую ячейку
AutoFill()	позволяет использовать автозаполнение для диапазона
Clear …	Методы с этим префиксом позволяют очистить содержимое диапазона от значений, форматирования, комментарий и т.п.
Copy()	копирует диапазон, по умолчанию в буфер обмена. По методу Cut данные исходного диапазона вырезаются в буфер обмена.
Delete()	удаляет данные текущего диапазона
Find()	позволяет провести поиск по ячейкам диапазона и вернуть новый объект Range, который представляет собой первую ячейку, в которой было найдено нужное значение.
GoalSeek()	позволяет применить автоподбор значений для функции Excel программным способом. Аналогом является команда **Сервис/Подбор параметра…**
Merge()	позволяет слить все ячейки диапазона в одну(верхнюю левую). Разбить обратно на несколько обычных можно при помощи метода UnMerge()
Parse()	позволяет разбить одну ячейку на несколько по указанному шаблону
PasteSpecial()	позволяет вставить то, что лежит в буфере обмена (вставлять с добавлением к существующим данным, с умножением, вычитанием, делением и т.п.)

Replace()	позволяет производить поиск и замену значений в указанном диапазоне
Select()	выделяет указанный диапазон
Sort()	проводит сортировку ячеек в диапазоне
SubTotal()	позволяет посчитать итоговое значение для диапазона

Пример 1. Решение уравнения с одной неизвестной.

Пусть надо найти корни уравнения $X^3 - 3X - 5 = 0$

Для решения этой задачи следует использовать метод GoalSeek объекта Range. Этот метод подбирает значение параметра (неизвестной величины), которое является решением уравнения с одной переменной. вычисляет корень, используя метод последовательных приближений, результат выполнения которого зависит от начального приближения. Синтаксис метода GoalSeek:

объект.GoalSeek(Goal, ChangingCell), где

объект — ячейка, в которую введена формула, являющаяся правой частью решаемого уравнения. В этой формуле роль параметра играет ссылка на ячейку, указанную в аргументе ChangingCell;

Goal — обязательный параметр, задающий значение левой части решаемого уравнения, не содержащей параметра;

ChangingCell — обязательный параметр, указывающий ссылку на ячейку, отведенную под параметр. Значение, введенное в данную ячейку до активизации метода GoalSeek, рассматривается как начальное приближение к искомому корню. Значение, возвращаемое в эту ячейку после выполнения метода GoalSeek, является найденным приближением к корню. Определение корня с заданной точностью и за предельно допустимое число итераций можно задать следующим образом:

```
With Application
    .MaxIterations = 1000
    .MaxChange = 0.0001
End With
```

Метод GoalSeek возвращает значение True, если решение найдено, и значение False в противном случае.

Следующий код ищет корень заданного уравнения при начальном приближении 1.

```
Sub Koren()
```

```
        Range("A1").Name = "x"
        Range("A1").Value = 1
        Range("B1").Formula = "=x^3-3*x-5"
If Range("B1").GoalSeek(Goal:=0, ChangingCell:= Range("x")) Then
        MsgBox "Корень:   " & Range("A1").Value
Else
        MsgBox "Корень не найден"
End If
   End Sub
```

Задание 1. Программно заполнить на рабочем листе 2 диапазон ячеек A1:A25 случайными числами со значениями из диапазона [0.1; 1). Двумя способами вывести упорядоченные значения в столбец C и в столбец D.

Задание 2. Программно заполнить на рабочих листах 1 и 2 диапазон A1:A25 целыми двузначными числами двумя способами.

Задание 3. Программно заполнить на рабочем листе 1 диапазон ячеек C2:E8 случайными числами со значениями из диапазона [0.1; 1). Перенести данные в диапазон A1:C7, скопировать последние данные в диапазоны, начиная с ячеек F5 и G3. Задать значениям столбца F желтый цвет, значениям столбца G — зеленый.

Задание 4. Создать форму с э.у. RefEdit. Программно заполнить диапазон C2:H24 рабочего листа 3 случайными числами со значениями из диапазона [0; 100]. Выбрать элементом управления RefEdit произвольный диапазон ячеек из заданного C2:H24. Найти среди выбранных значений минимальное, максимальное и среднее значения. Результат вывести в первую строку активного рабочего листа.

Задание 4. Написать программу нахождения корней уравнения $x^3 - ax^2 + b = 0$

Задание 5. В первом столбце первой и второй строки рабочего листа 1 записаны два натуральных числа — двузначное и трехзначное. В ячейку A3 вывести разность суммы цифр этих чисел. Контрольный пример: для A1 = 57, A2 = 695, то в A3 — -8.

Контрольные вопросы

Распечатать объектную модель MS Excel, Word.

В чем заключаются события BeforeRightClick() и BeforeDoubleClick().

Как можно получить ссылку на объект Range.

Привести пример использования свойства Formula объекта Range.

Описать синтаксис метода GoalSeek. Что возвращает этот метод?

Литература

VBA и программирование в MS Office для пользователей. СПб., БХВ-Петербург, 2006, 372с.

Демидова Л.А., Пылькин А.Н., Программирование в среде Visual Basic for Applications, М., Горячая линия-Телеком, 2004г., 174 с.

Гарнаев А. Самоучитель VBA. Технология создания пользовательских приложений. - СПб, БХВ-Петербург, 2004, 542с.

Гарнаев А. Excel, VBA, Internet в экономике и финансах - СПб, BHV, Санкт-Петербург, 2001, 336 с.

Курилович В. Информатика в задачах, примерах, алгоритмах, М., Солон-Р, 2005г.,143 с.

Юркин А.Г. Задачник по программированию. СПб., Питер, 2002, 182с.