



НАРОДНАЯ УКРАИНСКАЯ АКАДЕМИЯ

СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ
ТЕХНОЛОГИИ В ЭКОНОМИКЕ.

ОСНОВЫ ПРОГРАММИРОВАНИЯ
В СРЕДЕ MS OFFICE

Учебное пособие

Издательство НУА

НАРОДНАЯ УКРАИНСКАЯ АКАДЕМИЯ

СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
В ЭКОНОМИКЕ

ОСНОВЫ ПРОГРАММИРОВАНИЯ
В СРЕДЕ MS OFFICE

Учебное пособие

Для студентов 4 курса факультета «Бизнес-управление»,
обучающихся по специальности – Экономика

Харьков
Издательство НУА
2017

УДК 004.42:33(075.8)
ББК 32.973p30-1
С 56

*Утверждено на заседании кафедры информационных технологий
и математики Народной украинской академии
Протокол № 10 от 8. 05. 2017*

Р е ц е н з е н т: канд. техн. наук *К. С. Барашев*

В пособии рассмотрено использование языка Visual Basic for Applications для автоматизации решения экономических задач, механизмы взаимодействия приложений Microsoft Office.

С56 **Современные** информационные технологии в экономике. Основы программирования в среде MS Office : учеб. пособие для студентов фак. «Бизнес-управление», обучающихся по специальности – Экономика / Нар. укр. акад., [каф. информ. технологий и математики ; авт.-сост.: Е. И. Бобыр, С. Б. Данилевич]. – Харьков : Изд-во НУА, 2017.– 112 с.

У посібнику розглянуте використання мови Visual Basic for Applications для автоматизації рішення економічних задач, механізми взаємодії додатків Microsoft Office.

УДК 004.42:33(075.8)
ББК 32.973p30-1

© Народная украинская академия, 2017

ВВЕДЕНИЕ

От современного специалиста требуются умения использовать в своей работе новейшие достижения информационных технологий, в том числе и иметь представление о программировании на достаточном для решения своих профессиональных задач уровне. При решении сложных задач пользователь со знаниями основ программирования намного эффективнее сможет взаимодействовать с программистом-разработчиком.

Основным способом представления информации в любой организации является документ. Значительную часть рабочего времени уходит на поиск необходимой информации, ее обработку, создание и отправку документов. Помочь обработать и сформировать информацию в нужном виде и предназначены компьютерные технологии. Понятие документ давно вышло за рамки только отпечатанного на бумаге текста. Сейчас это целесообразно структурированная информация, которая чаще всего хранится на магнитных носителях. Она может быть выведена на печать, отправлена и получена средствами информационных сетей, представлена в виде презентации PowerPoint, включать в себя данные Access, диаграммы и расчеты таблиц Excel и т.п. Мощные средства вычисления, анализа и прогнозирования Microsoft Office позволяют обрабатывать такую бизнес-информацию. Пользователь имеет возможность подобрать для этого соответствующие инструменты. Стандартные приложения MS Office позволяют создавать сложные электронные документы. Но наибольший эффект достигается организацией взаимодействия приложений Microsoft Office, поддерживающих общий макроязык – *Visual Basic for Applications (VBA)*. Слово Visual означает, что с помощью этого языка реализуется визуальный стиль программирования, отличающийся тем, что программы создаются в основном из уже созданных профессиональными программистами унифицированных средств. Открывается возможность использовать как готовые, мощные средства создания документов (текстовые и графические редакторы, табличные процессоры, базы данных), так и VBA – одно из встроенных инструментальных средств приложений MS Office. Навыки, полученные при освоении других приложений, позволяют пользователю овладеть такой средой программирования. Использование макросов, пользовательских функций и форм, созданных с применением VBA, дополняют средства Microsoft Office создания документов.

Знакомство с VBA открывает путь освоения VB-технологии, включающей использование VBA (для создания сложных электронных документов, объединяющих приложения Microsoft Office), разработку отдельных приложений Windows с использованием, например, пакета VB-6 и Web-сценариев для Интернет на языке VBScript. Языки программирования VBA, VB, VBScript сходны и настроены для работы именно в среде Windows. Знакомство с идеологией программирования помогает глубже

познать возможности стандартных Windows приложений, повысить эффективность работы с ними.

Освоение технологии офисного программирования откроет новые возможности для повышения эффективности Вашей работы (автоматизация повторяющихся задач, создание новых возможностей в приложениях MS Office и др.).

ТЕМА 1. ВВЕДЕНИЕ В VBA

Суть объектно-ориентированного программирования. Объект. Свойство. Метод. Событие. Классы. Основные объекты MS Office.

Пакет программ MS Office и позволяет решить множество задач, связанных с ведением бизнеса, планированием, финансами, статистикой и т.п. и без явного программирования. Однако использование элементов программирования позволяет существенно автоматизировать работу, приспособить пакет для решения конкретных, специфических задач данной фирмы. Этому призван способствовать *Редактор Visual Basic (Сервис – Макрос – Редактор Visual Basic)*. Он позволяет записывать последовательности инструкций, которые должен выполнить компьютер, на языке программирования *Visual Basic для приложений (VBA)*. Это общий язык для всех приложений MS Office. Предусмотрена возможность записи программ (макросов) в автоматическом режиме (см. Тема 2. Создание и редактирование макропроцедур).

Редактор Visual Basic существенно расширяет арсенал инструментов для создания электронных документов, использующих как необходимые встроенные приложения, так и собственные разработки для достижения конкретных целей.

Пользователю, успешно освоившему работу с такими сложными стандартными приложениями, как Word, Excel, Access, Power Point и др., для дальнейшего совершенствования необходимо ознакомиться и взять на вооружение VBA.

VBA – язык объектно-ориентированного программирования (ООП).

1.1. Суть объектно-ориентированного программирования

Ориентирование на события – это стержень создания приложений Windows в VBA.

Windows – система, базирующаяся на сообщениях и событиях. Это значит, что каждое действие в Windows вызывает событие, которое в виде сообщения передается в приложение. Приложение анализирует сообщение и выполняет соответствующие этому сообщению действия.

По событийному принципу работают и приложения, созданные с помощью VBA. Сообщение передается объекту (например, *кнопке*), где затем вызывается необходимое событие (например, событие *Click*).

Одним из основных понятий VBA, таким образом, является понятие **объект**.

1.2. Объект, свойство, метод, событие

Многое из того, с чем мы будем работать в VBA, является объектами.

Объект – это программный элемент, который имеет свое отображение на экране, рассматриваемое как единое целое (элемент управления, форма или компонент приложения), содержит некоторые переменные, определяющие его свойства, и некоторые методы для управления объектом.

В VBA имеется много встроенных объектов, причем некоторые из них содержат другие (т.н. вложенные объекты). Так, Excel (объект **Application**) содержит рабочую книгу (объект **Workbook**), которая содержит рабочие листы (**Worksheets**), рабочий лист содержит диапазон ячеек (**Cells**) и т.д. Объектом самого высокого уровня является приложение (**Application**).

Свойство – это атрибут объекта, определяющий его характеристики (например, цвет, размер, положение на экране) или состояние объекта (например, видимость).

Чтобы изменить характеристики объекта нужно изменить значения его свойств. Свойства отвечают за внешний вид и поведение объекта.

Так, свойство **Caption** определяет текст надписи на объекте.

Каждый объект имеет свой специфический набор свойств. Изменяя свойства, можно изменять характеристики объекта. Установка значений свойств – это один из способов управления объектами. Для установки свойства необходимо ввести имя объекта, поставить точку, а за ней – имя свойства. Далее следует знак равенства и значение свойства. Синтаксис установки значения свойства объекта выглядит следующим образом:

Объект.Свойство = Значение свойства.

Методы – это рабочие операторы объекта. Они позволяют управлять объектом (перемещать по экрану, изменять его размеры, сделать объект видимым, скрыть и т.д.).

Синтаксис применения метода:

Объект. Метод.

Например, у объекта Excel – диапазон ячеек – имеется метод **Clear**, позволяющий очистить содержимое диапазона. Так, диапазон ячеек **Начальные_данные** можно удалить командой **Range ("Начальные_данные").Clear**.

Завершить работу с Excel можно, применив метод **Quit** (выход) к объекту **Application**:

Application.Quit.

Методы реализуются с помощью встроенных процедур VBA.

Событие – это характеристика объекта, которая описывает внешнее воздействие, на которое реагирует объект во время выполнения программы.

Событиями могут быть: нажатие на клавишу клавиатуры, щелчок мышью, движение мыши, перемещение объекта по экрану и т.п.

События инициируются:

действиями пользователя;

сообщениями от системных средств;

сообщениями от приложения, которое выполняется.

Чтобы изложенное не казалось слишком абстрактным, поясним основные понятия на примере телефона.

Телефон – объект.

Свойство объекта – цвет телефона, громкость звонка, его тембр.

Звонок телефона – это событие.

Процедура – подъем трубки.

Чтобы позвонить применяем метод – набор номера абонента.

1.3. Классы, основные объекты MS Office

Объекты объединяются в классы. К одному классу принадлежат объекты с одинаковым набором свойств, методов, событий.

Получить информацию о некоторых объектах можно, воспользовавшись специальным каталогом объектов VBA, который содержит список всех объектов, сгруппированных по категориям. Эти категории называются **библиотеками объектов**.

Каталог объектов можно открыть в окне диалога **Просмотр объектов (Object Browser)**.

Для этого нужно запустить редактор VBA (VBE:) командой **Сервис – Макрос – Редактор Visual Basic**, нажатием кнопки **Редактор Visual Basic** панели инструментов Visual Basic или сочетанием клавиш **Alt+F11**.

Браузер объектов вызывается из VBE:

выбором из меню объектов **Вид – Просмотр объектов**;

нажатием клавиши **F2**;

щелчком мышью на кнопке  – просмотр объектов панели инструментов.

В открывшемся окне в поле **Библиотеки\Книги** можно выбрать нужное приложение (Excel, Word, Access);

в списке **Объекты\Модули** – имя объекта.

Например, для Excel основными объектами являются: **Collection, Debug, Error, UserForm, UserForms**.

Каждый из указанных объектов имеет собственный набор свойств и методов, с помощью которых можно управлять поведением объекта, а значит и ходом выполнения программы.

Вопросы для самоконтроля

1. В чем заключается суть объектно-ориентированного программирования?
2. Что называется объектом?
3. На какие категории можно разделить параметры объекта?
4. Что называется событием объекта?
5. Что описывают процедуры?
6. Что называется методом объекта?
7. Что называется свойством объекта?
8. Каков синтаксис установки значения свойств объекта?
9. Приведите пример объекта, его событий, методов и свойств.
10. В каком окне редактора VBA можно выбрать свойства объекта?

Практическая работа к теме № 1

1. Определите в записях объект, метод, свойство, значение свойства:

Application.Quit

Worksheets ("Лист1").Chartobjects.Delete

Worksheets.Visible = False

Range("D1").Value = 2017

Range("C1:C10").Text = "Офисноепрограммирование"

UserForm12.Hide

2. Щелкните правой кнопкой мыши на ярлыке Лист1 и выберите команду **Просмотреть код**. В открывшемся окне в списке (**General**) выберите пункт **Worksheet** между появившимися строками *PrivateSubWorksheet_SelectionChange(ByValTargetAsExcel.Range)* и *EndSub* впишите код:

MsgBox "Браузер объектов вызывается из VBE: выбором из меню объектов Вид - Просмотр объектов; нажатием клавиши F2; щелчком мышью на кнопке – просмотр объектов панели инструментов."

Проверьте работу программы на VBA, произведя какие-либо изменения на Листе1.

3. Замените код следующим:

Dim Имя As String

Имя = InputBox("Введите свое имя", "Окно ввода", "Имя", 100, 100)

MsgBox "Привет, " & Имя

Проверьте выполнение программы.

ТЕМА 2. СОЗДАНИЕ И РЕДАКТИРОВАНИЕ МАКРОПРОЦЕДУР

Macrorecorder. Порядок работы с макросами в Word. Порядок записи макроса в Excel. Просмотр макроса. Создание макроса в редакторе VBA.

MS Office обладает уникальным средством записи программ в автоматическом режиме (макросов) даже без знания языков программирования. Это средство (*Macrorecorder*) позволяет заменить множество действий пользователя при создании электронного документа одним (нажатием на клавиши или щелчком мыши по созданному элементу управления, например, кнопке). При этом происходит автоматический перевод всех действий пользователя на язык VBA, что позволяет в дальнейшем многократно использовать эту процедуру, а после знакомства с основами VBA, изменить или усовершенствовать ее.

2.1. Macrorecorder

Макрос (макрокоманда) — это последовательность команд, записанная под определённым именем. Это одно из средств автоматизации повторяющихся действий. Макросы создаются специальным инструментом программирования — транслятором действий *Macrorecorder*. С момента запуска и до момента остановки транслятора *Macrorecorder*, на языке VBA создаётся программа (макрос). Пользователь может работать с объектами, представленными на экране (вводить текст в нужный абзац, изменять характеристики шрифта, помещать формулы в ячейки таблиц, использовать кнопки панелей инструментов, работать с папками и т.п.) и все его действия записываются и представляются в виде программы, которую впоследствии можно многократно применять. После знакомства с основами VBA, макросы можно изменять, модернизировать и использовать в других программах. Более того, *Macrorecorder* является уникальным средством автоматизации составления программ и изучения языка VBA.

2.2. Порядок работы с макросами в Word

Прежде, чем записывать макрос, нужно продумать последовательность действий (полезно даже записать их на бумаге).

Если в Word нет вкладки *Разработчик*, то следует создать эту вкладку. В Word 2007: *Кнопка Office – Параметры – Основные* – установить флажок *Показывать вкладку "Разработчик" на ленте*. В Word 2010 и выше: *Файл – Параметры – Настройка ленты* – установить флажок *Разработчик* в области *Настройка ленты: Основные вкладки*.

Чтобы записать макрос в программе Word существует несколько способов: в меню на вкладке Вид, Разработчик и кнопка в левом нижнем углу экрана

1. Так, в меню **Вид** последовательно выберите пункты **Макросы** и **Записать макрос**,
2. В поле **Имя макроса** появившегося окна **Запись макроса** (см. Рис. 1) введите имя макроса латинью или кириллицей без пробелов.

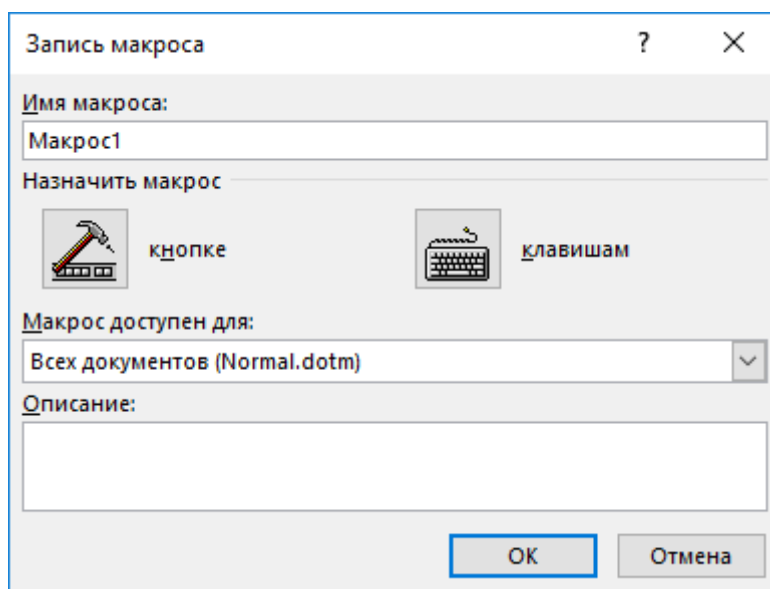


Рис. 1. Окно Запись макроса

Желательно, чтобы имя было "говорящим", т.е. помогающим вспомнить впоследствии цель создания макроса, его особенности. Если новому макросу будет присвоено имя уже существующего макроса, то последний будет заменен.

3. Выберите из списка **Макрос доступен для:** шаблон или документ, в котором будет храниться макрос.

Макросы хранятся в шаблонах или документах. Чтобы они были доступны всем документам в поле **Макрос доступен для:** укажите значение **Всех документов (Normal.dotm)**. Иначе выберите пункт с указанием только данного документа.

4. Заполните поле **Описание** (назначение макроса, комментарии и т.п.) для себя или пользователей, которые будут работать с данными документами;
5. Выберите одну из кнопок **Назначить макрос:** **кнопке** (чтобы запускать макрос нажатием кнопки на панели быстрого доступа) или **клавишам** (для запуска макроса с помощью сочетания клавиш);
6. Чтобы запускать макрос кнопкой на ленте следует добавить кнопку макроса на ленту: в меню **Файл** последовательно выберите пункты **Парамет-**

ры и *Настроить ленту*. В пункте *Выбрать команды из* выберите пункт *Макросы*. Выберите нужный макрос.

7. В пункте *Настройка ленты* выберите вкладку и настраиваемую группу, в которую вы хотите добавить макрос;

8. Если настраиваемой группы нет, то нажмите кнопку *Новая группа*. Затем нажмите кнопку *Переименовать*, введите имя группы (например, Мои_макросы). Нажмите кнопку *Добавить*.

9. Изменить изображение и имя для макроса можно кнопкой *Переименовать*.

10. Нажмите кнопку *Заккрыть*, чтобы начать записывать макрос;

11. Нажмите кнопку *Клавишам* для присвоения макросу сочетания клавиш;

12. Выберите записываемый макрос в списке *Команды*, введите сочетание клавиш в поле *Новое сочетание клавиш*, а затем нажмите кнопку *Заккрыть* для записи макроса;

13. Нажмите кнопку *ОК*, чтобы начать запись макроса;

14. Выполните действия, которые следует включить в макрос. Используйте клавиатуру для записи таких действий, как выделение текста или перемещение курсора. Мышь может быть использована для выбора нужных команд и параметров;

15. Для окончания записи макроса нажмите кнопку *Остановить запись*.

Для выполнения макроса выберите команду *Макросы*. В появившемся диалоговом окне *Макрос* (Рис. 2), выберите имя макроса, который требуется выполнить, в списке *Имя*;

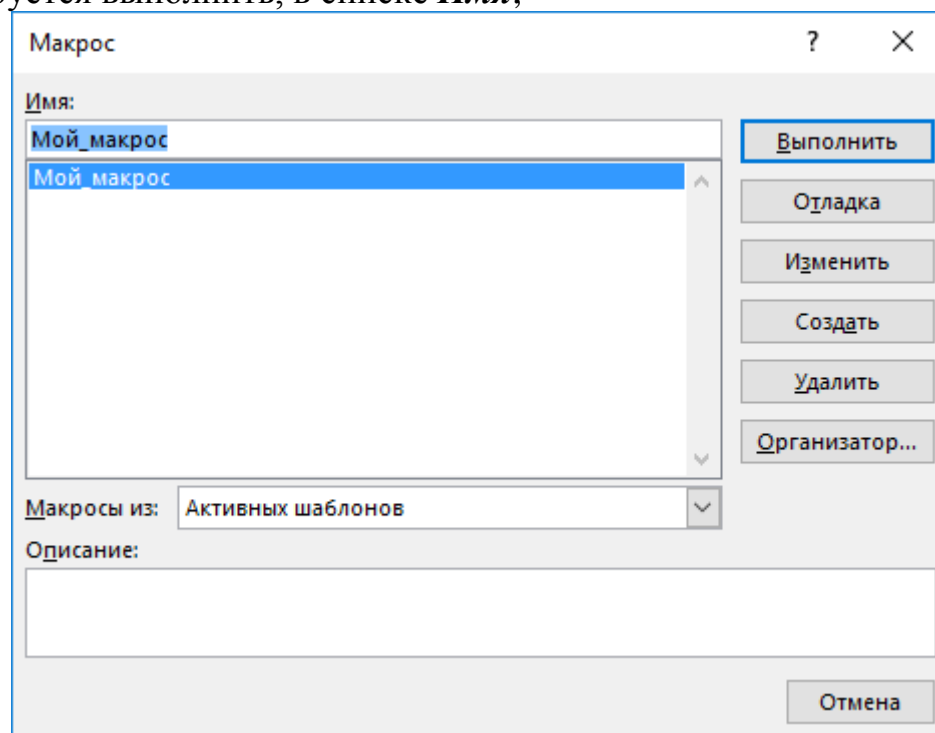


Рис. 2 Окно Макрос

Если нужного макроса нет в списке, выберите в поле **Макросы** другой из списка макросов.

Нажмите кнопку **Выполнить**.

Ненужный макрос можно удалить: используя команду меню **Сервис – Макрос**, нажать кнопку **Удалить** или открыв модуль в редакторе VBA, выделить весь блок с записью макроса и удалить её вместе с названием.

2.3. Порядок записи макроса в Excel

Для записи макроса в Excel нужно активизировать вкладку, отвечающую за управление и навигацию макросов. Для этого нужно перейти на вкладку **Файл** в группу **Параметры**. В появившемся диалоговом окне **Параметры Excel**, перейдите по вкладке **Настройка ленты** и в правом поле со списком поставьте маркер напротив вкладки **Разработчик**. В появившейся на ленте вкладке **Разработчик** в группе **Код** нажмите кнопку **Запись макроса**. В появившемся диалоговом окне (см. Рис. 3) следует ввести информацию о будущем записываемом коде, который будет фиксировать каждое действие в модуле VBA (ввод данных, создание диаграмм и т.п.). Чтобы остановить запись макроса, нажмите кнопку **Остановить запись** в группе **Код**. Другой вариант записи макросов – кнопка **Запись макроса** в левом нижнем углу рабочей книги Excel.

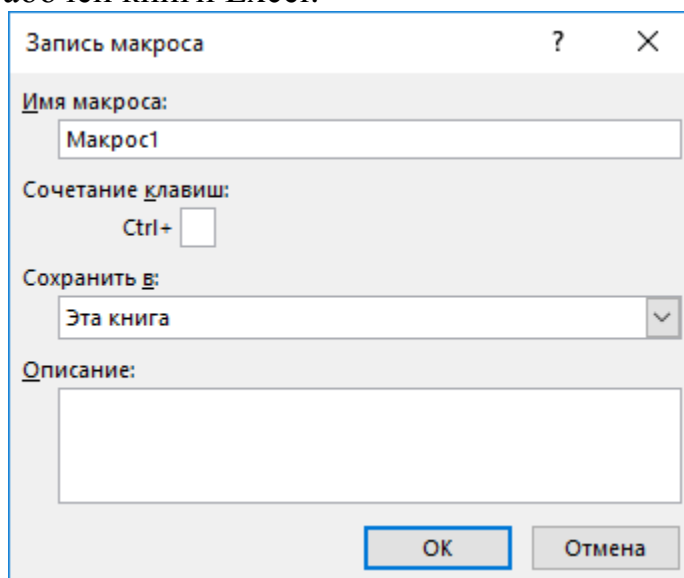


Рис. 3 Окно Запись макроса Excel

При задании имени следует избегать пробелов, заменяя их знаком подчёркивания (**Shift+mupe**) или начиная каждое слово с заглавной буквы без пробелов (например, МойМакрос).

2.4. Просмотр макроса

Чтобы посмотреть, как устроена макрокоманда следует нажать на кнопку **Макрос** в группе **Код**, а в появившемся диалоговом окне выделить имя макроса и нажать кнопку **Изменить**. Вам откроется окно редактора VBA со средой программирования.

В тексте программного модуля макрос выглядит так:

Sub Имя_макроса() – строка объявления макроса
тело макроса (различные операторы языка VBA)
End Sub.

Здесь *Sub* (сокращение слова *subprogram* – подпрограмма) и *End Sub*, – операторы, означающие начало и конец макроса. За оператором *Sub* следует имя макроса. Между этими двумя операторами расположено "тело" макроса – группа операторов языка, которые обрабатывают данные.

2.5. Создание макроса в редакторе VBA

Макросы можно записать и без использования программы *Macrorecorder*. Для этого следует нажать на кнопку **Макрос** в группе **Код**, а в появившемся диалоговом окне впишите правильно имя макроса. В поле **Находится** определите местонахождения макроса – **Все открытые книги, Эта книга**. Нажмите кнопку **Создать**.

Появится редактор VBA (см. Рис. 4). Между операторами *Sub* и *End Sub* можно записать программу.

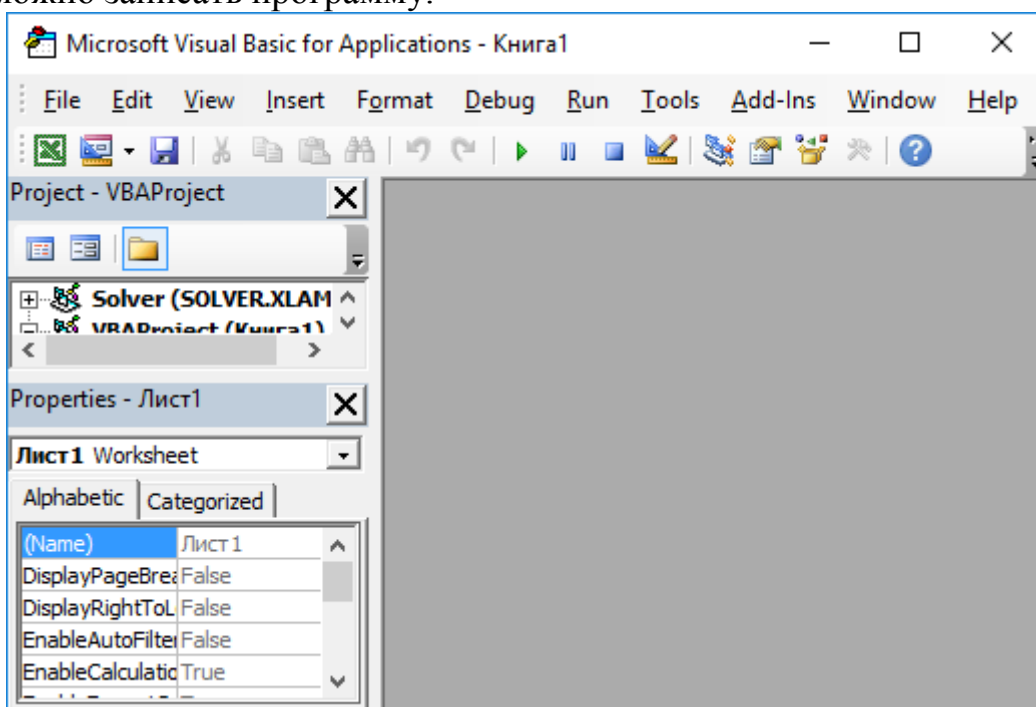


Рис. 4 Окно редактора VBA

Вопросы для самоконтроля

1. Что называется макросом?
2. Каково назначение инструмента программирования *Macrorecorder*?
3. Каков порядок действий при записи макросов в Word?
4. Каков порядок действий при записи макросов в Excel?
5. Чем отличаются команды **Остановить запись** и **Пауза**?
6. Какие ограничения накладываются на запись имени макроса?
7. Как запустить макрос на выполнение?

8. Как посмотреть код макрокоманды?
9. Что означают слова *Sub* и *End Sub*?
10. Что располагается между *Sub* и *End Sub*?

Практическая работа к теме № 2

1. При работе по составлению документа возможна ситуация, когда необходимо поменять местами два слова. Например, *ситуация благоприятная* → *благоприятная ситуация* и т.п. Для таких ситуаций заблаговременно подготовьте макрос в Word:

- 1.1. Создайте документ Word и наберите два слова.
- 1.2. Установите курсор на первом из двух слов.
- 1.3. Начните запись макроса. Назовите его "Поменять Местами".
- 1.4. Заполните окно ввода **Описание**.
- 1.5. Назначьте горячие клавиши или элемент управления для вызова макроса.
- 1.6. Выделите первое слово (**Shift-Ctrl-→**).
- 1.7. Вырежьте выделение в буфер обмена (**Ctrl-X**).
- 1.8. Передвиньте курсор на слово вправо (**Ctrl-→**).
- 1.9. Наберите пробел и вставьте слово (**Ctrl-V**).
- 1.10. Проверьте, как работает макрос, и просмотрите текст макроса.
2. Запишите макрос в Word для обмена местами двух букв, двух параграфов.
3. Запишите макрос в Word для вставки в начало текста выражения: "Уважаемый г-н (г-жа)", а в конец текста – "С уважением, Ваш покорный слуга." (или другую фразу по Вашему выбору).
4. Запишите макрос в Word для вставки символа *телефон* из таблицы символов Wingdings в указанное место текста.
5. Запишите макрос в Word для вставки в указанное место текста формулы, пересчитывающей сумму без НДС 20% в сумму с НДС. Для этого:
 - 4.1. Установите курсор в месте для вставки суммы.
 - 4.2. Начните запись макроса. Назовите его "НДС".
 - 4.3. Заполните окно ввода **Описание**.
 - 4.4. Назначьте горячие клавиши или элемент управления, для вызова макроса.
 - 4.5. На вкладке **Вставка** в группе **Текст** нажмите кнопку **Экспресс-блоки** и выберите пункт **Поле**.
 - 4.6. В открывшемся окне **Поле** выберите категорию **Формулы**.
 - 4.7. Введите код поля: = 1,2 и нажмите **ОК**.
 - 4.8. Выделите появившееся значение поля и с помощью павой кнопки мыши выберите **Изменить поле...**
 - 4.9. Вставьте после 1,2 знак умножения (*) и поле **Fill-in** категории **Слияние**. Нажмите **ОК**.

- 4.10. Введите в поле ввода появившегося окна какое-либо число. Нажмите **ОК**.
- 4.11. Остановите запись макроса.
- 4.12. Проверьте его выполнение, обновляя поле нажатием клавиши **F9**.
5. Создайте в Excel макрос, заполняющим группу ячеек по образцу, заданном в некоторой ячейке (например, A2).
- 5.1. В автоматическом режиме.
- 5.2. В редакторе VBA:
Sub Образец() 'Объявляется макрос под именем Образец.
Range("A2").Select 'К объекту: диапазон ячейка A2 применяется метод выделить (*Select*). 'В результате активизируется ячейка.
Selection.Copy 'Выделенный объект копируется.
Range("A2:G9").Select 'Выделяется диапазон ячеек A2:G9.
ActiveSheet.Paste 'Объявляется объект – активный лист, который заполняется в соответствии с информацией из буфера обмена.
Range("A2").Select 'Опять выделяется ячейка A2.
End Sub 'Конец макроса.
- 5.3. Проверьте выполнение макросов.

ТЕМА 3. СОЗДАНИЕ ПРОСТЫХ ПРОГРАММ НА VBA

Иерархия объектов VBA. Некоторые объекты Word. Некоторые объекты Excel. Алфавит и основные конструкции VBA. Синтаксис VBA. Типы данных. Переменные и константы. Преобразования. Область определения и видимости переменных и констант. Инструкция Option Explicit. Организация диалога.

Создатели MS Office постарались предусмотреть наиболее типичные ситуации, возникающие при создании сложных электронных документов, максимально облегчить процесс их создания с помощью встроенных инструментов (в том числе *Macrorecorder*). Однако каждая фирма наряду с типичными операциями выполняет и специфические, уникальные, предусмотреть которые заранее в быстро меняющемся мире невозможно. Кроме того, есть операции, которые вручную выполнить слишком сложно или просто невозможно. Квалифицированный пользователь, знакомый со спецификой решаемых экономических задач и с VBA, может достаточно эффективно настроить систему с учетом возможных нюансов. Затраты на изучение основ VBA позволят сэкономить как материальные, так и временные ресурсы.

VBA – относительно простой язык программирования, но, чтобы написать программу, нужно понять основные принципы построения языка,

изучить алфавит, систему правил и т.п. (как при изучении нового иностранного языка).

3.1. Иерархия объектов VBA

Основа языка – **объект** – элемент, предназначенный для разработки приложений и управления их выполнением.

Понятие объект позволяет объединить в нечто целое данные и программный код для их обработки.

Все визуальные объекты, такие как известные Вам в Excel рабочий лист (*Worksheet*), диапазон (*Range*), диаграмма (*Chart*), форма (*User Form*) являются объектами VBA.

Семейство представляет собой объект, содержащий несколько однотипных объектов. Так, объект *Рабочие Книжки (Workbooks)* содержит все открытые объекты *Рабочая книга (Workbook)*. Объекты семейства нумеруются и может быть идентифицирован по номеру, либо по имени.

Объектная библиотека VBA включает более 100 объектов, находящихся на различных уровнях иерархии подчиненности.

.Рис. 5 представляет иерархическая модель встроенных объектов VBA.

Иерархия подчиненности определяет связь между объектами и показывает пути доступа к ним.

Полная ссылка на объект состоит из ряда имен, вложенных последовательно друг в друга объектов. Разделителями имен объектов в этом ряду являются точки. Ряд начинается с объекта *Application* и заканчивается именем самого объекта. Например, полная ссылка на ячейку A1 рабочего листа *Лист1* рабочей книги с именем *Архив* имеет вид:

Application.Workbooks (“Архив“).Worksheets (“Лист1“).Range (“A1”)

Application (Приложение)	
Workbooks (Workbook) <i>Рабочая книга</i>	AddIns (AddIn) <i>Расширение</i>
→ Worksheets (Worksheet) <i>Рабочий лист</i>	AutoCorrect <i>Автом.коррект.</i>
→ Charts (Chart) <i>Диаграмма</i>	Assistant <i>Помощник</i>
→ DocumentProperties (DocumentProperty) <i>Свойство документа</i>	Debug <i>Отладка</i>
→ VBProject <i>VBпроект</i>	Dialogs (Dialog) <i>Диалог</i>
→ CustomViews (CustomView) <i>Заказной вид</i>	CommandBars (CommandBar)

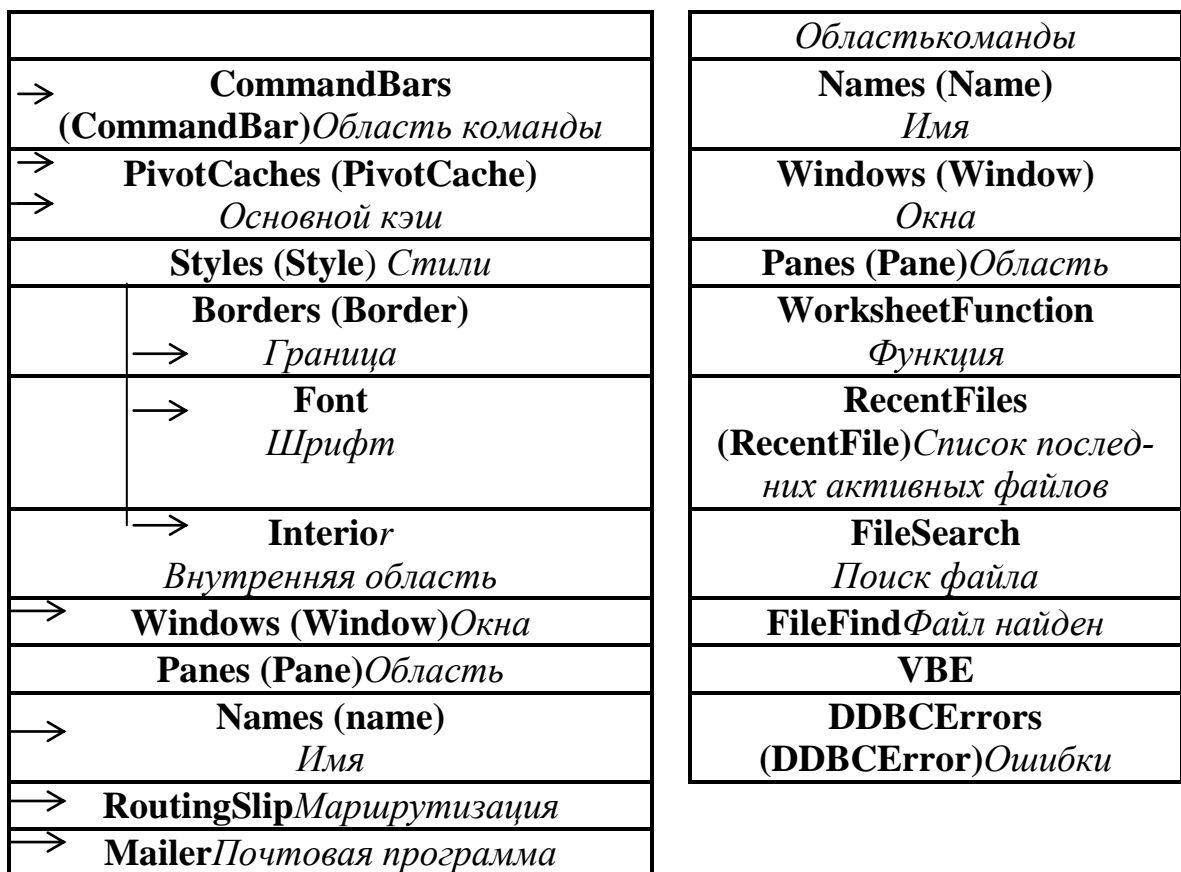


Рис. 5. Иерархическая модель встроенных объектов VBA

Однако приводить каждый раз ссылку на объект совершенно не обязательно, так как объекты, которые активны в данный момент, можно опускать, т. е. если в нашем примере активны Excel и Рабочая книга Архив, то ссылка будет иметь вид:

Worksheets (“Лист1“). Range (“A1”)

Если активен и *Рабочий лист Лист1*, то можно записать:

Range(“A1”)

3.2. Некоторые объекты Word

Объект *Application* является центральным (корневым) объектом Word. Он определяет само это приложение. Все другие объекты Word встроены в него. Во многих случаях при записи программ он не упоминается т.к. является глобальным объектом. Именно он определяет свойства и поведение других объектов Word таких, как открытые окна, документы и др.

Некоторые свойства объекта *Application*

Caption	определяет заголовок окна Word.
ActiveWindow	содержит ссылку на текущее окно.
ActiveDocument	содержит ссылку на текущий документ.
DisplayScrollBars	отвечает за включение (=True) или выключение (=False) отображения полосы прокрутки.

DisplayStatuslBar отвечает за включение (=True) или выключение (=False) отображения строки состояния.

StatusBar позволяет записать произвольный текст в строку состояния.

Некоторые методы объекта *Application*

Quit закрывает окно и завершает работу Word.

Quit wdSaveChanges закрывает окно и завершает работу Word с сохранением всех изменений.

Работа с объектами семейства **Documents**. **ActiveDocument**. **ThisDocument**.

Семейство **Documents** представляет все открытые документы Word.

Объект **ActiveDocument** указывает на активный документ.

Объект **ThisDocument** указывает на документ, в котором находится выполняемый макрос.

Некоторые методы семейства **Documents**

Add используется для создания нового документа, **Open** – для открытия существующего документа, а **Close** – для группового закрытия всех открытых документов.

Чтобы закрыть только некоторые документы, нужно их обозначить по имени (например, **Documents("Документ1").Close**) или по номеру (например, **Documents(1).Close**).

Метод **Activate** используется для выбора указанного документа из открытых документов и его активизации.

Некоторые методы объекта **ActiveDocument**

Close используется для закрытия активного документа, **Save** – для сохранения активного документа под текущим именем, а **SaveAs** – для сохранения активного документа под указанным именем.

Некоторые методы объекта **ThisDocument**

Close используется для закрытия документа, в котором находится выполняемый макрос, **Save** – для сохранения документа, в котором находится выполняемый макрос, под текущим именем, а **SaveAs** – для сохранения документа, в котором находится выполняемый макрос под указанным именем.

Объект **Range** представляет собой диапазон текста. Он используется для работы с текстом всего документа, абзацами, предложениями, словами, символами.

Задать текстовый диапазон можно разными способами, например:

Range(Start:=номер первой позиции, End:=номер последней позиции).

Чтобы создать объект **Range** в документе применяют метод **Range**.

У многих объектов (таких, как **Paragraph**) есть свойство **Range**, которое используется, чтобы вернуть объект **Range**.

Семейства: *Paragraphs, Sentenses, Words, Characters* представляют соответственно параграфы, предложения, слова, символы.

Устройство этих семейств сходно.

Некоторые свойства этих семейств:

First указывает на первый элемент семейства;
Last указывает на последний элемент семейства
Count содержит число элементов семейства;
Alignment задаёт способ выравнивания.
Метод *Delete* удаляет указанные элементы семейства.

3.3. Некоторые объекты Excel

Объект *Application* представляет собой всё приложение Excel и является центральным (корневым) объектом Excel. Он определяет само это приложение. Все другие объекты Excel встроены в него. Во многих случаях при записи программ он не упоминается т.к. является глобальным объектом. Именно он определяет свойства и поведение других объектов Excel.

Некоторые свойства объекта *Application*:

Caption определяет заголовок окна Excel.
Workbooks содержит семейство всех открытых книг.
ActiveWorkbook содержит ссылку на активное окно.
Worksheets содержит семейство рабочих листов активной книги.
Activsheet указывает на активный лист активной книги.
ThisWorkbook указывает на рабочую книгу, в которой находится выполняемый макрос.
DisplayScrollBars отвечает за включение (**=True**) или выключение (**=False**) отображения полосы прокрутки.
DisplayStatuslBar отвечает за включение (**=True**) или выключение (**=False**) отображения строки состояния.
StatusBar позволяет записать произвольный текст в строку состояния.

Некоторые методы объекта *Application*:

SaveAs сохраняет активную рабочую книгу с заданным названием.
Activate делает активной указанную книгу.
Close закрывает указанную книгу без сохранения изменений.
Open открывает заданную книгу.

Семейство Sheets

Семейство *Sheets* включает в себя все листы открытой книги. Обращаться к элементам семейства можно по номерам или именам.

Некоторые методы семейства Sheets:

Add	создаёт новые листы.
Delete	удаляет лист.
Visible	скрывает лист.
Move	перемещает лист в заданную позицию в книге.
Copy	копирует лист в заданную позицию в книге.
Quit	завершает работу с приложением.

Основная работа в рабочем листе ведётся с объектами *Range*. Он определяет диапазон ячеек.

Некоторые свойства (семейства) объекта Range:

Cells	позволяет обратиться к ячейкам диапазона.
EntireColumn	вызывает столбец, к которому принадлежит диапазон.
EntireRow	вызывает строку, к которой принадлежит диапазон.
Formula	позволяет поместить в ячейку формулу.
Value	содержит значение ячейки.
Font	содержит объект <i>Font</i> , свойства которого позволяют изменять параметры шрифта текста ячеек.
Style	определяет стиль для ячеек диапазона.

3.4. Алфавит и основные конструкции VBA

Программирование – это умение описать алгоритм решения задачи с помощью средств языка программирования. В связи с этим, прежде чем приступить непосредственно к программированию, необходимо определить какие действия должна выполнить программа (составить алгоритм), а затем описать эти действия с помощью операторов и служебных слов языка.

В VBA имеется около двухсот встроенных операторов и функций. Каждый оператор и функция имеют четкую структуру (синтаксис), то есть правила грамматики, пунктуации и орфографии. Для записи операторов используются алфавит, различные символы и ключевые слова.

Алфавит языка включает большие и малые буквы латинского алфавита и десятичные цифры от нуля до девяти. Арифметические операции задаются символами:

+	сложение;
-	вычитание;
*	умножение;
/	деление;
\	деление без остатка;
mod	деление по модулю.

Кроме символов, для записи операторов в языке используются ключевые слова (зарезервированные слова), такие как *Dim, As, New, If, Then, Else, While, End*.

Пользователь не может использовать ключевые слова в других целях.

3.5. Синтаксис

Синтаксис- это правило записи операторов и других конструкций программы. В каждой строке кода программы обычно записывается один оператор. Если операторов в строке несколько, то они разделяются двоеточием, например,

$$Y = A + B: X = C * D.$$

Разделители строк – это символ подчеркивания (*_*), который дает возможность сформировать длину строки так, чтобы она умещалась на экране. Строка программы в VBA может иметь максимум 1023 символа и не больше 10 разделителей.

Комментарии используются в языке для пояснения отдельных фрагментов программы и игнорируются во время ее выполнения. Для выделения начала комментария можно использовать верхнюю запятую (*'*) или команду *Rem*, например,

A = 2 Rem – это оператор присваивания

A = 2' это оператор присваивания

Таким образом, текст комментария размещается справа от символа комментария.

В VBA код программы состоит из одного либо нескольких операторов, процедур и функций, которые для выполнения программы преобразуются компилятором в машинный код.

3.6. Типы данных

Программы оперируют с данными, которые хранятся в оперативной памяти компьютера во время его работы. Все данные, с которыми работает VBA, относятся к определенному типу в зависимости от роли которые они играют в программе.

Целые цифровые значения в различных диапазонах описываются типами:

Byte используется для описания целых чисел 0 до 255;

Integer – от -32768 до 32767;

Long – от -2147483648 до 2147483647.

Для работы с числами с плавающей запятой используются типы:

Single – в диапазоне от -3,4E38 до -1,4E-45 для отрицательных значений и от 1,4E-45 до 3,4E38 для положительных значений.

Double – в диапазоне от -1,7E308 до -4,9E-324 для отрицательных значений и от 4,9E-324 до 1,7E308 для положительных значений.

Для денежных расчетов, а также для проведения расчетов с фиксированной десятичной точкой, в которых требуется обеспечить высокую точность, используется тип данных

Currency. Такое представление позволяет отобразить числа в диапазоне от -922 337 203 685 477,5808 до 922 337 203 685 477,5807.

Для обработки информации о дате и времени используют тип данных **Date**. Значение даты и времени надо поместить между двумя #:

#11/12/2001#

#14:25:00#.

Тип **String** служит для хранения текстовых строк. Каждый символ, хранимый в данной переменной, занимает 1 байт памяти. Передаваемый текст надо заключать в кавычки, например: "Азбука Visual Basic for Applications".

Существует два типа строковых значений:

строки переменной длины, которые могут содержать до приблизительно 2 миллиардов символов;

строки постоянной длины, которые могут содержать от 1 до приблизительно 64К (2¹⁶) символов.

Тип **boolean** имеет область значений **True** и **False**. Значение **True** в VB соответствует 1, **False** соответственно – 0.

Тип **Object** служит для хранения других объектов (например, **OLE**). Переменные типа **Object** сохраняются как 32-разрядные (4-байтовые) адреса, в которых содержатся ссылки на объекты. Переменной, описанной этим типом, можно затем присвоить (с помощью инструкции **Set**) ссылку на любой объект, созданный в приложении.

Тип **VARIANT** может передавать значения всех выше описанных типов, за исключением строк (тип **String**) фиксированной длины и определяемых пользователем типов. Его можно не указывать (это называется **Неявным объявлением**). Этот тип используется, если заранее неизвестно, какого типа данные потребуются. Следует учитывать, что каждого типа резервируется область в памяти ПК, а тип **VARIANT** является самым громоздким. Допустимыми числовыми данными являются любые целые или действительные числа в диапазоне от -1,797693134862315E308 до -4,94066E-324 для отрицательных значений и от 4,94066E-324 до 1,797693134862315E308 для положительных значений.

3.7. Переменные и константы

Для хранения данных используют **переменные** и **константы**.

Переменная – это именованная область памяти, отведенная для временного хранения данных, которые могут изменяться при выполнении программы.

Каждая переменная должна иметь имя, которое должно начинаться с буквы, быть уникальным в области ее определения, не должна содержать

более 255 символов и не содержать точку или символ описания типа (**%**,**&**, **#**, **!**, **@**, **)**).

Чтобы использовать переменную в программе, её нужно объявить, используя зарезервированное слово **Dim**.

Например, объявим переменную по имени **X**, которая может принимать целочисленные значения:

Dim X As Integer.

При попытке присвоить **X** число, выходящее за пределы диапазона от -32768 до 32767, возникает ошибка. При присваивании **X** дробного числа, выполняется округление. Например:

X = 32768 'Вызывает ошибку.

X = 5.9 'Задаёт для **X** значение 6.

Константа – это именованный элемент, сохраняющий постоянное значение в течение выполнения программ. Константа может быть задана как строковый или числовой литерал (константа в явном представлении), другая константа или любая комбинация констант и арифметических и логических операторов, за исключением операторов **Is** и возведения в степень.

Константы применяют в случаях, когда нужно много раз использовать одно и то же значение.

Чтобы использовать константу в программе, её нужно описать и присвоить имя.

Для описания константы и определения её значения используется инструкция **Const**. После описания константу нельзя модифицировать и нельзя присваивать ей новое значение.

В следующем примере константа под именем **conAge** описывается как **Integer**, и ей присваивается значение 34.

Const conAge As Integer = 34

Допускается также описание нескольких констант в одной строке. В этом случае, чтобы задать тип данных, надо указать определённый тип для каждой константы. В следующем примере константы **conAge** и **conWage** описываются как **Integer**.

Const conAge As Integer = 34, conWage As Currency = 35000

3.8. Преобразования

VBA содержит процедуры, которые преобразовывают значения одного типа в значения другого типа. Так, функция **Val** преобразует строки в число, а **Str** преобразует числа в строку. **Hex**, **Oct** преобразуют числа в другую систему счисления. **CBool**, **CByte**, **CCur**, **CDate**, **CDbl**, **CDec**, **CInt**, **CLng**, **CSng**, **CStr**, **CVar**, **CVErr**, **Fix**, **Int** – один тип данных в другой. Например, **CDbl** – в тип **Double**.

Функции преобразования типов (например, **CDbl** и **CLng**) корректно работают, если возможно преобразование значений строк в числовые зна-

чения типа (*Double* и *Long*, соответственно). Иначе происходит ошибка. На экране появиться сообщение "Несовпадение типов".

3.9. Область определения и видимости переменных и констант

Область определения определяет доступность переменной, константы, или процедуры для других процедур. Имеется три уровня областей определения: уровень процедуры; локальный уровень модуля и общий уровень модуля.

Переменные и константы не могут быть использованы программой за пределами их видимости.

Если, например, константа может использоваться во всех модулях и процедурах, то сначала нужно написать зарезервированное слово **Public**:

Public Const MyConst As Integer = 59

Если константа может использоваться только в данном модуле, то сначала нужно записать слово **Private**:

Private Const Stavka As Single = 0.007

Константа, предназначенная для использования только в данной процедуре, объявляется без слов **Public** и **Private**:

Const W As String = "Подпись директора".

Область видимости переменной или константы определяется двумя обстоятельствами: где она объявлена и какое дополнительное ключевое слово было использовано.

Глобальная переменная будет доступна во всех процедурах или модулях, относящихся к данной книге или документу.

3.10. Инструкция Option Explicit

При использовании инструкции **Option Explicit** необходимо явно описать все переменные с помощью инструкций **Dim**, **Private**, **Public**, **ReDim** или **Static**. При попытке использовать неописанное имя переменной возникает ошибка во время компиляции.

Когда инструкция **Option Explicit** не используется, все неописанные переменные имеют тип **VARIANT**.

Рассмотрим пример части кода:

Option explicit 'Заставляет явно объявлять все переменные.

Dim MyVar 'Объявляет переменную.

MyInt = 10 'Необъявленная переменная – причина ошибки.

MyVar = 10 'Использование объявленной переменной' не приводит к ошибке.

Для автоматической вставки в программу инструкции **Option Explicit** откройте редактор VBA, дайте команду меню: **Сервис – Параметры...**, а в открывшемся диалоговом окне **Параметры** установите флажок **Явное описание переменных**.

3.11. Организация диалога

VBA содержит около двадцати встроенных функций и операторов, обеспечивающих взаимодействие пользователя с приложениями.

Функция **InputBox** используется для получения информации от пользователя.

Синтаксис: **InputBox(Message, Title, Default, y, x)**.

Здесь **Message** – сообщение-подсказка; **Title** – заголовок; **Default** – значение по умолчанию; **y, x** – координаты расположения окна.

Функция **MsgBox** используется для выдачи информации пользователю.

Синтаксис: **MsgBox(Msg, Style, Title, Help, Ctxt)**.

Здесь **Msg** – сообщение для пользователя;

Style – кнопки (например, **vbOKOnly** – отображается только кнопка "OK");

Title – заголовок;

Help – имя файла справки со сведениями о данном диалоговом окне (необязательный). Если этот аргумент указан, необходимо указать также аргумент контекст

Ctxt – контекст – номер соответствующего раздела справочной системы (необязательный). Числовое выражение, определяющее, что если этот аргумент указан, то необходимо указать также аргумент **helpfile**.

Вопросы для самоконтроля

1. Как программно установить свойство объекта?
2. Как программно воздействовать на объект?
3. Как иницируются события?
4. Что называется классом объектов?
5. Какой объект является корневым (центральным) приложения Word?
6. Что представляет собой объект **Range** приложения Word?
7. Что представляет собой объект **Range** приложения Excel?
8. В каком окне редактора VBA можно выбрать свойства объекта?
9. Какие символы используются для написания программ?
10. Какими символами задаются арифметические операции?
11. Что называется ключевым словом?
12. Где хранятся данные в ПК?
13. Какой тип данных нужно использовать для описания целых чисел от 0 до 255?
14. Какой тип данных нужно использовать для описания целых чисел от -32768 до 32767?
15. Какой тип данных нужно использовать для описания целых чисел от -2147483648 до 2147483647?
16. Для описания, каких данных используется тип **Single**?
17. Какие данные описываются типом **Double**?

18. Какие данные описываются типом *Currency*?
19. Какого типа эти данные: *#11/12/2001#*, *#14:25:00#*?
20. Для чего служит тип данных *String*?
21. Какова область значений данных типа *Boolean*?
22. Что называется *Неявным объявлением* данных?
23. Что называется переменной?
24. Что называется константой?
25. В каких случаях используют зарезервированные слова *Dim*, *As*, *Const*, *Public*, *Private*?
26. Что означает *Область видимости*?
27. Что означает *Option Explicit*?
28. Как автоматизировать вставку инструкции *Option Explicit* в программу?

Практическая работа к теме № 3. Работа с объектами.

1. Работа с объектом *Application*.

Постановка задачи. Знание основных свойств объектов (прежде всего корневого – *Application*) и умение с помощью VBA их изменять, позволяет целесообразно оформить документ согласно потребностям и вкусам пользователя, а не только довольствоваться встроенными возможностями приложения.

Создайте макрос, запуск которого приведет к переименованию заголовка окна на “Документы”, нормализует активное окно, увеличит кнопки панелей инструментов, с экрана уберет полосу прокрутки. Во время работы макроса строка состояния должна содержать сообщение: “Выполняется макрос...”, а в появившемся окне будет запрашиваться фамилия пользователя в родительном падеже, и после его указания заголовок окна изменится на “Документы (имя пользователя)”. Функция *MsgBox* выведет на экран сообщение о завершении работы макроса.

Выполнение:

- 1.1. Создайте новый документ Word.
- 1.2. Откройте редактор VBA.
- 1.3. Нажмите кнопку “Окно проекта” на панели инструментов.
- 1.4. Щёлкните 2 раза по значку *ThisDocument*.
- 1.5. В открывшемся окне наберите текст программы:
`Sub Create() ' Объявляется макрос под именем Create`
`'Application.Caption = "Документы" 'Устанавливается заголовок`
главного окна.

`'Application.ActiveWindow.Windowstate = wdWindowStateNormal` Нормализует активное окно.

`'Application.CommandBars.LargeButtons = True` Увеличивает кнопки панелей инструментов.

`'Application.DisplayScrollBars = True` Убирает с экрана полосу прокрутки.

'Application.StatusBar = "Выполняется макрос..." Устанавливает запись в строке состояния.

Application.Caption = "Документы " & InputBox("Введите фамилию в родительном падеже", "Окно ввода", , 2500, 1000) Запрашивается фамилия пользователя.

MsgBox "Работа макроса завершена" Выводится окно сообщений.

End Sub Конец макроса

Поочередно убирая знак апострофа, проверьте выполнение программ, не забудьте вернуть всё к первоначальному состоянию.

2. Работа с семейством *Documents*.

Постановка задачи. Создать макросы, для открытия существующего документа, активизации документа под определенным номером (например, вторым) из коллекции документов, закрытия документов с сохранением изменений.

Выполнение:

2.1. Создайте новый документ Word.

2.2. Откройте редактор VBA.

2.3. Нажмите кнопку "Окно проекта" на панели инструментов

2.4. Щёлкните 2 раза по значку *ThisDocument*.

2.5. В открывшемся окне наберите текст программы:

Sub MyDocument() Объявляется макрос под именем MyDocument.

Dim MyDoc As Document Объявляется переменная типа документ.

Set MyDoc = Documents.Add Новый документ добавляется в коллекцию документов.

'Откроем существующий документ.

'В скобках укажите адрес существующего документа

'Set MyDoc = Documents.Open("Set MyDoc = Documents.Open("C:\Program Files\Microsoft Office\Шаблоны\Другие документы\Изысканное резюме.dot")'

'Documents(2).Activate' Активизируем второй документ из коллекции документов

'Documents("Документ1").Close' Закроем документ под именем Документ1 из коллекции документов.

'Documents.Close wdPromptToSaveChanges' Закроем документы с сохранением изменений.

End Sub

2.6. Поочередно убирая знак апострофа, проверьте выполнение программ, не забудьте вернуть всё к первоначальному состоянию.

3. Работа с объектом *Range*.

Постановка задачи. Создать макрос, запуск которого делает первые десять позиций активного документа полужирным шрифтом и вставляет в начало документа фразу "*Документ №__*".

Выполнение:

- 3.1. Создайте новый документ Word.
- 3.2. Откройте редактор VBA.
- 3.3. Нажмите кнопку "Окно проекта" на панели инструментов.
- 3.4. Щёлкните 2 раза по значку *ThisDocument*.
- 3.5. В открывшемся окне наберите текст программы.
Sub MyRange()
ActiveDocument.Range(Start:=0, End:=9).Bold 'Первые десять позиций 'активного документа сделать полужирным шрифтом
ActiveDocument.Range(Start:=0, End:=0).InsertBefore "Документ №__" "Вставить в начало документа фразу в кавычках.
End Sub
- 3.6. Поочерёдно убирая знак апострофа, проверьте выполнение программ, не забудьте вернуть всё к первоначальному состоянию.

4. Работа с объектами *Paragraphs, Sentences, Words*.

Постановка задачи. Создать макрос, запуск которого производит к выравниванию по центру первого параграфа активного документа, окружает его объемной границей толщиной 3 пт., вставляет в конец последнего абзаца слово " *Подпись*", производит заливку второго абзаца узором типа 15%, устанавливает стиль первого абзаца *Заголовок 1*, первому предложению активного документа устанавливает шрифт типа *Arial*, значение регистра: "Начинать с Прописных", для первого слова активного документа устанавливает курсив.

Выполнение:

- 4.1. Создайте новый документ Word.
- 4.2. Откройте редактор VBA.
- 4.3. Нажмите кнопку "*Окно проекта*" на панели инструментов
- 4.4. Щёлкните 2 раза по значку *ThisDocument*.
- 4.5. В открывшемся окне наберите текст программы.
Sub MyParagraph()
'Выравнивание по центру первого параграфа активного документа.
'ActiveDocument.Paragraphs(1).Alignment = wdAlignParagraphCenter
'Параграф из коллекции параграфов активного документа, обладающий свойством быть первым, окружить границей с определенными свойствами.
'ActiveDocument.Paragraphs.First.Borders.OutsideLineStyle = wdLineStyleEmboss3D
'К параграфу из коллекции параграфов активного документа, обладающий свойством быть последним, применить метод вставки в конец параграфа выражение в кавычках.
'ActiveDocument.Paragraphs.Last.Range.InsertAfter "End"

'Ко второму абзацу активного документа применить заливку узором типа 15%.

```
'ActiveDocument.Paragraphs(2).Shading.Texture = wdTexture15Percent
```

'Установить свойство стиль первого параграфа.

```
'ActiveDocument.Paragraphs.First.Style = "Заголовок 1"
```

'Первому предложению активного документа установить шрифт типа Arial.

```
'ActiveDocument.Sentences(1).Font.Name = "Arial"
```

'Установить для первого предложения значение регистра: "Начинать с Прописных".

```
'ActiveDocument.Sentences(1).Case = wdTitleWord
```

'Установить курсив для первого слова из коллекции слов активного документа.

```
'ActiveDocument.Words.First.Italic = True
```

End Sub

4.6. Поочерёдно убирая знак апострофа, проверьте выполнение программ, не забудьте вернуть всё к первоначальному состоянию.

5. Исследование свойств объектов Excel.

Постановка задачи. Создать макрос, запуск которого позволяет: установить заголовок окна "**Цены за 2016 год**", сохранить текущую книгу в файле с именем "**Обзор цен**", изменить имя текущего листа на "**Январь**", увеличить масштаб текущего окна на 50%, включить режим десятичный формат при вводе. Задать число десятичных разрядов, равных 3, заполнить диапазон ячеек A1:A12 названиями месяцев, установить формат шрифта: Полужирный; тип: "Arial"; размер: 13, закрасить ячейки и снабдить узором, записать в ячейку A2 значение 10, установить в ячейку A12 формулу "**=SUM(A2:A11)**".

Выполнение:

5.1. Запустите редактор VBA(меню **Сервис – Макрос – Макросы**).

5.2. В появившейся таблице:впишите имя макроса "**МесяцыГода**". В поле **Находится** определите местонахождения макроса – **Эта книга**. Нажмите кнопку **Создать**.

5.3. В тексте программного модуля редактора VBA запишите макрос:

```
Sub МесяцыГода()
```

'Инструкция With выполняет последовательность инструкций над объектом Application, представляющим приложение Microsoft Excel.

```
With Application
```

'свойство Caption (заголовок)принимает значение текста, который помещается в строке заголовка приложения.

```
.Caption = "Цены за 2001 год"
```

'Активная книга сохраняется под именем "Обзор цен".

```
.ActiveWorkbook.SaveAs "Обзор цен"
```

'Активный лист переименовывается.

```

.ActiveSheet.Name = "Январь"
'Устанавливается масштаб 150%.
.ActiveWindow.Zoom = 150 'устанавливается числовой формат с дву-
мя знаками после разделителя.
.FixedDecimal = True
.FixedDecimalPlaces = 2
End With 'Конец инструкции
'Устанавливаются значения ячеек.
Range("A1") = "Обзор цен"
Range("A2") = "Январь"
Range("A12") = "Итого:"
'Данные ячейки A2 заполняются в диапазон A2:A11.
Range("A2").AutoFill Range("A2:A11"), xlFillMonths 'Инструкция With
выполняет последовательность инструкций над объектом Font, вызываемо-
го соответствующим свойством объекта Range. With Range("A1:A12").Font
'устанавливаются свойства объекта Font: полужирное начертание
символов типа Arial 13-го размера.
.Bold = True
.Name = "Arial"
.Size = 13
End With 'Конец инструкции 'вводится значение цены программно.
ActiveSheet.Range("B2") = 10
'формулаавтосуммы
Range("B12").Formula = "=SUM(B2:B11)"
End Sub

```

5.4. Проверьте работу макроса.

Практическая работа к теме № 3. Типы переменных.

1. Проверка правильности ввода числовых данных.

Постановка задачи. Одним из способов создания надежных программ является проверка правильности ввода данных. Создайте макрос, определяющий является ли введенное числом, снабдите макрос элементом управления (кнопкой или "горячими клавишами").

Выполнение:

1.1. Заполните тело макроса кодом:

```

Dim Результат As Boolean
Dim a As Variant
a = Val(InputBox("Введите число", "Boolean"))
Результат = CBool(a)
MsgBox prompt:="Вы ввели число? Ответ:" & Результат

```

1.2. Проверьте выполнение макроса, вводя в диалоговое окно цифровые и буквенные символы. По справке изучите особенности функции Val().

2. Проверка правильности ввода данных типа **Byte**.

Постановка задачи.Создайте макрос, определяющий является ли введенное числом типа *Byte* снабдите его элементом управления (кнопкой или "горячими клавишами").

Выполнение:

2.1. Заполните тело макроса кодом:

```
Dim Результат As Byte
Результат = CByte(InputBox("Введите число от 0 до 255, иначе
ошибка", "Byte "))
MsgBox "Число " & Результат & " " & "в пределах допустимых зна-
чений"
```

2.2. Проверьте выполнение макроса, вводя в диалоговое окно числа, например, 1, 255, 256.

3. Проверка правильности ввода данных типа *Date*.

Постановка задачи.Создайте макрос, который определит дату по введенному числу. Снабдите его элементом управления (кнопкой или "горячими клавишами").

Выполнение:

3.1. Заполните тело макроса кодом:

```
Dim Результат As Date
Результат = CDate(InputBox("Введите число", "Date"))
MsgBox "Этому числу соответствует " & Результат
```

3.2. Проверьте выполнение макроса, вводя в диалоговое окно числа. Обратите внимание, что целая часть соответствует дате от 1 января 100 года до 31 декабря 9999 года, а дробная часть соответствует времени. Отрицательные числа соответствуют датам до 30 декабря 1899 года.

4. Проверка правильности ввода данных типа *Currency*.

Постановка задачи. Создайте макрос для ввода данных типа *Currency* снабдите его элементом управления (кнопкой или "горячими клавишами").

Выполнение:

4.1. Заполните тело макроса кодом:

```
Dim Результат As Currency
Результат = CCur(InputBox("Введите число от -
922337203685477,5808_
до 922337203685477,5807", "Currency"))
MsgBox Результат
```

4.2. Проверьте выполнение макроса, вводя в диалоговое окно числа. Обратите внимание, что выведенное число имеет 4 знака после запятой. Если дробная часть имеет более 4 знаков, то производится округление.

5. Проверка правильности ввода данных типа *Integer*.

Постановка задачи. Создайте макрос для ввода данных типа *Integer*. Снабдите его элементом управления (кнопкой или "горячими клавишами").

Выполнение:

5.1. Заполните тело макроса кодом

Dim Результат As Integer

Результат = CInt(InputBox("Введите целое число от -32768 до 32767 число,

иначе ошибка", "Integer"))

MsgBox Результат

5.2. Проверьте выполнение макроса, вводя в диалоговое окно числа.

6. Проверка правильности ввода данных типа **Double**.

Постановка задачи.Создайте макрос для ввода данных типа **Double**. Снабдите его элементом управления (кнопкой или "горячими клавишами").

Выполнение:

6.1. Заполните тело макроса кодом:

Dim Результат As Double

Результат = CDbI(InputBox("Введите значение типа Double", "Double"))

MsgBox Результат & " – значение типа Double является вещественным числом удвоенной точности с плавающей запятой "

6.2. Проверьте выполнение макроса, вводя в диалоговое окно числа типа **Double**.

7. Проверка правильности ввода данных типа **Long**.

Постановка задачи. Создайте макрос для ввода данных типа **Long**. Снабдите его элементом управления (кнопкой или "горячими клавишами").

Выполнение:

7.1. Заполните тело макроса кодом

Dim Результат As Long

7.2. *MsgBox Результат & " – значение типа Long – целые числа от -2147483648 до 2147483647. Дробы округляются."*

Проверьте выполнение макроса, вводя в диалоговое окно числа, округляются ли дроби.

8. Проверка правильности ввода данных типа **Single**.

Постановка задачи. Создайте макрос для ввода данных типа **Single**. Снабдите его элементом управления (кнопкой или "горячими клавишами").

Выполнение:

8.1. Заполните тело макроса кодом

Dim Результат As Single

Результат = CSng(InputBox("Введите вещественное число от -3,402823E38 до -1,401298E-45 для отрицательных и от 1,401298E-45 до 3,402823E38 для положительных", "Single"))

MsgBox Результат & " – значение типа Single"

8.2. Проверьте выполнение макроса, вводя в диалоговое окно числа, округляются ли дроби.

9. Проверка правильности ввода данных типа *String*.

Постановка задачи. Создайте макрос для ввода данных типа *String*. Снабдите его элементом управления (кнопкой или "горячими клавишами").

Выполнение:

9.1. Заполните тело макроса кодом

```
Dim Результат As String
```

```
Результат = CStr(InputBox("Введите значение типа String",  
"String"))
```

```
MsgBox Результат & " – значение типа String"
```

9.2. Проверьте выполнение макроса, вводя в диалоговое окно символы (цифры в том числе).

10. Работа с различными системами счисления.

Постановка задачи. Создайте макрос, переводящий десятичное число в число шестнадцатеричной системы счисления, снабдите его элементом управления (кнопкой или "горячими клавишами").

Выполнение:

10.1. Заполните тело макроса кодом:

```
Dim Результат As Variant
```

```
Dim a As Variant
```

```
a = Val(InputBox("Введите число" "Hex"))
```

```
Результат = Hex(a)
```

```
MsgBox prompt:="Число " & a & " " _
```

```
& "в шестнадцатеричной системе счисления = " & Результат.
```

10.2. Проверьте выполнение макроса.

10.3. Создайте макрос, переводящий десятичное число в число восьмеричной системы счисления.

ТЕМА 4. ОПЕРАТОРЫ ЯЗЫКА VBA.

ФУНКЦИИ

Операторы. Оператор присваивания. Функции языка VBA.

Экономика стала одной из наиболее математизированных наук. При решении задач, связанных с анализом экономической ситуации, прогнозированием, процессом принятия решений используются как стандартные математические функции, логические, текстовые и т.д., так и специальные экономические. При этом пользователь ПК освобождается от многих громоздких вычислений. Основной задачей становится правильное использование операторов и функций, знание их конструкции, структуры, синтаксиса. С помощью VBA пользователь имеет возможность

создавать собственные (пользовательские) функции, отражающие специфику решаемых задач бизнеса.

4.1. Операторы

Оператор – базовая единица действия в алгоритмических языках. Каждому оператору языка соответствует своя алгоритмическая конструкция.

Простейшими операторами являются оператор присваивания и оператор условного перехода.

4.2. Оператор присваивания

Оператор присваивания предназначен для присвоения переменной одного значения.

Структура оператора имеет вид:

A=Выражение

Символ "=" означает операцию присваивания.

Порядок выполнения: вычисляется значение выражения, стоящего в правой части оператора, а полученное значение присваивается переменной A.

Переменные в выражении должны быть одного типа (исключение: переменная может быть вещественного типа, а выражение целого).

Пример оператора:

$Y = a * d * \text{Sqr}(x) / (21 - d)$

Для записи операторов в программе в одну строку необходимо использовать специальный разделитель – двоеточие, например:

$A = 1 : B = A * C : Y = A + B.$

Для присвоения объекта переменной, описанной как объект, применяется инструкция **Set**.

Пример инструкции Set.

В этом примере инструкция **Set** присваивает диапазон на листе **Sheet1** объектной переменной **myCell**:

Sub ApplyFormat()

Dim myCell As Range

Set myCell = Worksheets("Sheet1").Range("A1")

With myCell.Font

.Bold = True

.Italic = True

EndWith

End Sub

Инструкции, задающие значение свойства, также являются инструкциями присвоения.

В следующем примере задается свойство **Bold** объекта **Font** для активной ячейки:

ActiveCell.Font.Bold = True

Таким образом, оператор присваивания используется в VBA программах как при записи арифметических, логических и выражений типа сравнения, так и при задании свойств и методов объектов и объектных переменных.

4.3. Функции языка VBA

Функция – это оператор, который выполняет определенные действия, а затем возвращает результат своей работы в программу.

Стандартные функции – это функции, вычисления которых являются встроенными процедурами языка.

К стандартным функциям принадлежат математические, строковые, финансовые и др. функции.

Наличие встроенных функций значительно упрощает процесс программирования, т.к. не требует создания программ для расчета этих величин.

Для обращения к встроенной функции необходимо указать её имя, передать ей необходимые аргументы и получить результат – возвращаемое значение.

Например: $Y=Sqr(x)$

Здесь переменная Y получает значение корня квадратного из x .

Ниже приведены наиболее часто используемые встроенные математические функции языка:

Функция	Действие функции
<i>Abs(x)</i>	Возвращает абсолютное значение x
<i>Ant(x)</i>	Возвращает арктангенс x , угол задается в радианах
<i>Cos(x)</i>	Возвращает косинус x , угол задается в радианах
<i>Exp(x)</i>	Возвращает e в степени x
<i>Rnd(x)</i>	Генерирует случайное число в диапазоне 0-1
<i>Sgn(x)</i>	Возвращает знак числа x , 1 при $x < 0$, 0 при $x = 0$, -1 при $x > 0$
<i>Sin(x)</i>	Возвращает синус угла x , угол задается в радианах
<i>Sqr(x)</i>	Возвращает квадратный корень x
<i>Log(x)</i>	Возвращает натуральный логарифм x
<i>Tan(x)</i>	Возвращает тангенс угла x , угол задается в радианах
<i>Str(x)</i>	Преобразует числовое значение x в строку
<i>Val(x)</i>	Преобразует строку в числовое выражение

В отличие от математических встроенных функций строковые функции возвращают строку и работают с одним либо несколькими строковыми аргументами

Часто используемые встроенные строковые функции:

Функция	Действия функции
<i>Chr(int)</i>	Возвращает символ, ASCII-код которого равен числовому аргументу
<i>Len(str)</i>	Возвращает число символов в строке

<i>Right(str,int)</i>	Возвращает int завершающих символов строки аргумента str
<i>Str()</i>	Преобразует числовой аргумент в строку – цифровую запись этого числа
<i>Lcase(str)</i>	Возвращает аргумент, записанный символами только нижнего регистра
<i>Ucase(str)</i>	Возвращает аргумент, записанный символами только верхнего регистра
<i>Left(str,int)</i>	Возвращает int первых символов строки

Часто используемые встроенные функции даты и времени

<i>Функция</i>	<i>Действие функции</i>
<i>DateAdd</i>	Возвращает дату, к которой добавлен указанный временной интервал.
<i>DateDiff</i>	Возвращает число интервалов между двумя датами.
<i>DatePart</i>	Возвращает указанный компонент даты.
<i>Day</i>	Возвращает целое число (от 1 до 31 включительно), которое представляет день месяца
<i>Date</i>	Возвращает текущую системную дату.
<i>Hour</i>	Возвращает целое число (от 0 до 23 включительно), которое представляет часы в значении времени.
<i>Time</i>	Возвращает текущее время по системным часам компьютера.
<i>Timer</i>	Возвращает число секунд, прошедших после полуночи.
<i>DateSerial</i>	Преобразует три числа: год, месяц и день в значение даты и времени.
<i>DateValue</i>	Возвращает значение даты/времени для строки, представляющей дату
<i>Now</i>	Возвращает текущую дату и время по системному календарю и часам компьютера.
<i>Minute</i>	Возвращает целое число (от 0 до 59 включительно), которое представляет минуты в значении времени
<i>Second</i>	Возвращает целое число (от 0 до 59 включительно), которое представляет секунды в значении времени
<i>TimeSerial</i>	Возвращает значение времени, соответствующее указанным часу, минуте и секунде.
<i>TimeValue</i>	Возвращает значение, содержащее время.
<i>Year</i>	Возвращает целое число, представляющее год.
<i>Weekday</i>	Возвращает целое число, представляющее день недели.

Особенно часто применяется функция ***DatePart***, которая позволяет выбрать любую составляющую даты и времени, а также производить вычисления.

Ниже перечислены допустимые значения аргумента ***interval*** функции ***DatePart***:

<i>Значение</i>	<i>Описание</i>
<i>уууу</i>	Год.
<i>Q</i>	Квартал.
<i>m</i>	Месяц.
<i>Y</i>	День года.
<i>D</i>	День месяца.
<i>w</i>	День недели.
<i>ww</i>	Неделя.
<i>h</i>	Часы.
<i>n</i>	Минуты.
<i>s</i>	Секунды.

Часто используемые встроенные финансовые функции:

<i>Функция</i>	<i>Действие функции</i>
<i>DDB</i>	Возвращает величину амортизации имущества для заданного периода с использованием метода двукратного учета амортизации или иного явно указанного метода.
<i>FV</i>	Возвращает значение, указывающее будущее значение суммы регулярных платежей при заданной учетной ставке.
<i>IPmt</i>	Возвращает значение, указывающее часть периодического платежа, приходящуюся на выплату процентов по займу. Предполагается, что выплаты делаются регулярно, а сумма выплат и учетная ставка остаются постоянными.
<i>IRR</i>	Возвращает значение, указывающее норму прибыли для последовательности периодических финансовых операций (выплат или поступлений).
<i>MIRR</i>	Возвращает значение, указывающее модифицированную норму прибыли для последовательности периодических финансовых операций (выплат или поступлений).
<i>Nper</i>	Возвращает значение, указывающее количество периодов (платежей), необходимых для накопления на счете заданной суммы при фиксированной процентной ставке.
<i>NPV</i>	Возвращает значение, указывающее итоговое сальдо ряда финансовых операций в проекции на текущий момент (с учетом оценки капитала).
<i>PPmt</i>	Возвращает значение, указывающее часть периодического платежа, приходящуюся на выплату собственно займа. Предполагается, что выплаты делаются регулярно, а сумма выплат и учетная ставка остаются постоянными.

<i>Pmt</i>	Возвращает значение, указывающее объем регулярных платежей по займу при фиксированной учетной ставке.
<i>Rate</i>	Возвращает значение типа Double, указывающее процентную ставку, необходимую для получения указанной суммы за определенный срок путем регулярных взносов.
<i>SLN</i>	Возвращает значение, указывающее снижение стоимости за выбранный интервал времени по методу с равномерной амортизацией
<i>SYD</i>	Возвращает значение, указывающее снижение стоимости за выбранный интервал времени по методу с линейной амортизацией.

VBA предоставляет возможность пользователю создавать собственные пользовательские функции, работать с которыми на рабочем листе можно при помощи мастера функций точно так же, как и с любой встроенной функцией.

Для создания функции пользователя создайте книгу Excel, откройте редактор VBA, вставьте стандартный модуль (меню: **Вставка – Модуль**) и введите текст функции.

Например:

```
Function Celsius(fDegrees)
```

```
    Celsius = (fDegrees - 32) * 5 / 9
```

```
End Function
```

(в примере функция *Celsius* пересчитывает градусы Фаренгейта в градусы Цельсия.);

После нажатия клавиши **F2** в появившемся окне объектов в поле **Классы** найдите объект **Модуль1**. Щелкните правой кнопкой мыши на значке объекта в поле **Компонент Модуль1**, выберите команду **Свойства..** и впишите описание функции в появившемся окне **Параметры компонента**. Затем перейдите в Excel с помощью **Мастера функций** в категории **Полный алфавитный перечень** вызовите пользовательскую функцию *Celsius*.

Вопросы для самоконтроля:

1. Что называется оператором?
2. Для чего предназначен оператор присвоения?
3. Что называется функцией в VBA?
4. Что называется стандартной функцией?
5. Как вызвать встроенную функцию?
6. Для чего предназначены строковые функции?
7. Для чего предназначены функции даты и времени?
8. Какие аргументы допустимы для функции **DatePart**?
9. Каковы аналоги финансовых функций VBA и Excel?

Практическая работа к теме №4
Практическая работа к теме Функции VBA

1. Работа с математическими функциями.

Постановка задачи. При решении конкретных задач всем (в том числе и экономистам) приходится использовать и неэкономические величины. После создания соответствующего макроса отпадает необходимость поиска свойств таких величин в справочниках, учебниках и т.п. Создайте макрос, переводящий значение угла из градусов в радианы:

Выполнение:

1.1. Заполните тело макроса кодом:

```
Option Explicit
Sub ГрадусыВРадианы()
Dim УголВГрадусах As Double
Dim УголВРадианах As Double
Dim pi As Double 'число pi – математическая константа, прибли-
тельно равная 3,1415926535897932.
'Вычисление числа pi.
pi = 4 * Atn(1) 'Atn(1) возвращает значение типа Double, содержа-
щее арктангенс числа
УголВГрадусах = CDbI(InputBox("Введите числовое значение угла
в градусах"))
УголВРадианах = CStr(УголВГрадусах * pi / 180) 'Значение угла
в радианах, преобразованное в текст.
'Вызов на экран окна сообщений со значением угла в радианах
MsgBox prompt:="Угол в радианах равен " & УголВРадианах
'Вызов на экран окна сообщений со значениями тригонометрических
функций угла в радианах.
'Format() возвращает значение типа Variant (String), содержащее
выражение, отформатированное согласно инструкциям, заданным
в описании формата.
'Chr(13)- символ возврата каретки позволяет разделить строки.
'Chr(10)- символа перевода строки.
MsgBox prompt:="Синус равен " _
& Format(Sin(УголВГрадусах * pi / 180), "0.00") _
& Chr(13) _
& "Косинус равен " & Format(Cos(УголВГрадусах * pi / 180), "0.00") _
& Chr(10) & "Тангенс равен " & _
Format(Tan(УголВГрадусах * pi / 180), "0.00")
End Sub
```

1.2. Выйдя из режима конструктора, проверьте работу макроса.

2. Работа с функциями даты.

Постановка задачи. При заключении договоров, составлении отчетности и т.п. важно оперативно получать информацию о той или иной дате: к какому кварталу она относится, какой по счету это день с начала года, к какой неделе эта дата относится и т.п. Создайте подобный макрос, выводящий информацию о вводимой дате:

Выполнение:

2.1. Заполните тело макроса кодом:

```
Option Explicit
Sub ДатаВремя()
Dim Дата As Date
Дата = InputBox("Введите дату в формате даты")
MsgBox prompt:="Дата: " & Дата _
    & Chr(13) _
    & "Квартал " & DatePart("Q", Дата) _
    & Chr(10) & "День в году: " _
    & DatePart("Y", Дата) _
    & Chr(10) & "Неделя в году: " _
    & DatePart("ww", Дата) _
    & Chr(10) & "Месяц в полном формате: " _
    & Format(Дата, "mmmm") _
    & Chr(10) & "День недели: " _
    & Format(DatePart("w", Дата), "dddd") _
    & Chr(10) & "Сегодняшняя дата в полном формате: " _
    & Format$(Дата, "Long date") _
    & Chr(10) & "Дата через 5 лет в полном формате: " _
    & Format$(DateAdd("yyyy", 5, Дата), "Long date")
End Sub
```

2.2. Проверьте работу макроса.

3. Работа со строковыми функциями (**Left**, **Right**, **Len**).

Постановка задачи. При создании финансовых документов часто требуется записывать значения сумм с указанием наименования валюты. Создайте макрос, преобразующий сумму, выраженную десятичной дробью, в сумму в грн и коп.

Выполнение:

3.1. Заполните тело макроса кодом:

```
Sub Сумма()
Dim Строка As String
Строка = InputBox("Введите значение суммы.Пример: 123,44")
Строка = Format(Строка, "#0.00")
ActiveDocument.Range(Start:=ActiveDocument.Paragraphs(1).Range.End - 1, _
```


*End:=ActiveDocument.Paragraphs(1).Range.End – 1).InsertAfter
Left(Строка, Len(Строка) – 3) + " грн. " _
+ Right(Строка, 2) + " кон"*

End Sub

3.2. Проверьте работу макроса.

4. Создание пользовательских функций в Excel.

4.1. Пользовательская функция, отображающая имя пользователя на листе.

Постановка задачи. Создайте пользовательскую функцию, которая возвращает в ячейку имя пользователя, указанное на вкладке *Общие* диалогового окна *Параметры*. Для этого создайте модуль и введите текст:

*Function USERNAME()
 USERNAME = Application.USERNAME
End Function*

Проверьте работу функции при помощи *Мастера функций*.

4.2. Пользовательская функция, отображающая имя книги.

Постановка задачи. Создайте пользовательскую функцию, которая возвращает в ячейку имя книги. Для этого в модуль введите текст:

*Function WORKBOOKNAME() As String
 WORKBOOKNAME = Application.
 Caller.Parent.Parent.Name
End Function*

Проверьте работу функции при помощи *Мастера функций*.

4.3. Пользовательская функция, отображающая имя листа.

Постановка задачи. Создайте пользовательскую функцию, которая возвращает в ячейку имя листа. Для этого в модуль введите текст:

*Function SHEETNAME(rng as Range) As String
 SHEETNAME = rng.Parent.Name
End Function*

Проверьте работу функции при помощи *Мастера функций*, указывая ячейки из разных листов.

4.4. Создание пользовательской функции индекс притока в страну инвестиций.

Постановка задачи. Несмотря на обширную коллекцию встроенных функций, может возникнуть потребность в своей (пользовательской) функции. Создайте функцию, **индекс притока в страну инвестиций**, который определяется по формуле: $INV = (I_i / I_w) / (G_i / G_w)$, **G** - совокупный мировой продукт страны (i) и всего мира (w); **I** - приток инвестиций в страну (i) и во все страны мира (w). **INV** - индекс притока в страну прямых иностранных инвестиций.

Выполнение:

4.4.1. Создайте рабочую книгу Excel. Войдите в редактор VBA.

4.4.2. Вставьте модуль (меню: **Вставка – Модуль**). Запишите код:

Option Explicit

Function INV (Ii As Double, Gi As Double, Iw As Double, Gw As Double) As Double

$$INV = (Ii / Iw) / (Gi / Gw)$$

End Function

4.4.3. Откомпилируйте проект (меню: **Отладка – Компилировать VBAProject**).

4.4.4. Дайте команду **Макросы в Разработчике**.

4.4.5. В открывшемся окне **Макрос** введите имя макроса: *INV*.

4.4.6. Нажмите кнопку **Параметры**.

4.4.7. В открывшемся диалоговом окне **Параметры макроса** заполните поле **Описание**:

Данный показатель характеризует относительные успехи страны в привлечении инвестиций.

Нажмите кнопку **ОК**.

4.4.8. Созданный объект можно просмотреть в окне **Просмотр объектов (ObjectBrowser)**, где выберите **VBAProject** из перечня **Проект/Библиотека**.

4.4.9. Щелкните правой кнопкой мыши по названию *INV*.

4.4.10. В открывшемся контекстном меню выберите команду **Свойства** и в открывшемся диалоговом окне **Параметры компонента** просмотрите поле **Описание**.

4.4.11. Откройте рабочий лист Excel.

4.4.12. Найдите пользовательскую функцию *INV* в полном алфавитном перечне всех функций окна **Мастер функций**. Проверьте её выполнение.

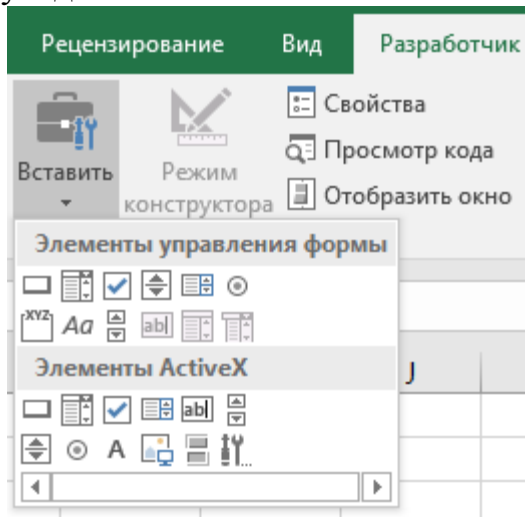
ТЕМА 5. ЭЛЕМЕНТЫ УПРАВЛЕНИЯ

Элементы управления. Создание и настройка элемента управления. Некоторые общие свойства элементов управления.

Создавая свой уникальный электронный документ, следует позаботиться о его внешнем виде и максимальном удобстве в его управлении другими пользователями, т.е. о "дружественном" интерфейсе документа. Технология визуального программирования позволяет создавать рабочую поверхность документа и элементы его управления непосредственно на экране. Встроенный стандартный набор элементов управления (кнопки, списки и др.) и средства изменения их свойств позволяют, с одной стороны, создавать вполне профессиональный, интуитивно понятный интерфейс, а, с другой, – проявить собственную индивидуальность.

5.1. Элементы управленияActiveX.

Элементы управления(вкладка *Разработчик* – группа *Элементы управления* – кнопка *Вставить*) позволяют пользователю взаимодействовать с приложениями, автоматизировать работу с документами, упростить работу с данными.



Все элементы управления делятся на элементы управления формами и элементы управления ActiveX. Для работы с элементами управления ActiveX требуется использовать VBA, что делает их более гибкими.

Элементы управления:

1. **Кнопка.** – при нажатии выполняется команда.
2. **Поле со списком.** Вставляет объект, являющийся сочетанием списка и поля. Пользователь может либо выбрать нужное значение из списка, либо ввести его в поле.
3. **Флажок** – элемент, который независимо от других может находиться в трех состояниях: включенном, выключенном (могут быть определены как пользователем, так и программистом) и неактивном (определяется программой).
4. **Список.** Средство выбора одного варианта из нескольких представленных. Допускается прокручивание списка, если не все его элементы видны одновременно.
5. **Текстовое поле** – дает пользователю возможность ввести текстовую информацию с целью последующей обработки в программе.
6. **Полоса прокрутки.** Может передавать в программу число, равное расстоянию в пунктах от ее начала. Верхний предел расстояния неограничен.
7. **Счетчик.** Может передавать в программу свое значение от 1 до 100.
8. **Переключатель** – элемент, который может находиться во включенном, выключенном и неактивном состояниях. В отличие от флажка, если

один из переключателей в группе включен, то остальные включены быть не могут.

9. **Подпись** (поле отображения текста) – отображает неизменяемый текст, например, подпись к рисунку.

10. **Изображение**. В этот элемент можно вставить из файла рисунок.

11. **Выключатель**. Это кнопка, которая может находиться в нажатом или отжатом состоянии.

5.2. Создание и настройка элемента управления

На рабочем поле документа (например, листе Excel) можно размещать элементы управления при помощи кнопки **Вставить** на вкладке **Разработчик** в группе **Элементы управления** в разделе **Элементы ActiveX**.

Изменить элемент управления можно в режиме конструктора. Для этого на вкладке **Разработчик** в группе **Элементы управления** включите **Режим конструктора**.

Кнопка **Свойства** позволит придать элементу нужный вид и свойства.

Нажатие кнопки **Просмотр кода** позволяет ввести название макроса для выполнения.

5.3. Некоторые свойства элементов управления

Рассмотрим некоторые свойства, которыми обладает большинство элементов управления (ЭУ).

Позиция. Позицию ЭУ определяют четыре свойства **Left**, **Top**, **Height**, **Width**. Свойства **Left**, **Top** задают координаты верхнего левого угла элемента управления. Свойства **Height** и **Width** – его высоту и ширину. Отсчет в системе координат ведется сверху вниз Y и слева направо X.

Цвет. Управление цветом осуществляется с помощью свойств **BackColor**, **FillColor**, **ForeColor**, которым по умолчанию назначаются стандартные цвета.

BackColor – цвет фона выбирают в диалоговом окне настройки цвета.

Свойства **ForeColor** определяет цвет, используемый для отображения текста и графики в элементе управления, а с помощью свойства **FillColor** – установить цвет заполнения рисованных объектов.

Параметры шрифта. Вид шрифта в элементах управления выбирается путем установки значений свойства **Font**.

Доступность и видимость ЭУ. Для этого используются два свойства – **Enabled** и **Visible**.

Enabled – определяет, будет ли элемент управления реагировать на события или нет. (**False**).

Visible – позволяет сделать ЭУ невидимым (**False**).

Свойство Имя. Свойство **Name** является идентификатором элемента управления. Оно играет особую роль. Ошибка при его задании приводит

к серьезным последствиям (невыполнению кода программы). Поэтому сначала всегда следует задавать имя ЭУ и лишь, затем писать для него процедуру обработки события.

Свойства *Parent* и *Container*. *Parent* указывает на родительский объект и обеспечивает доступ к его методам и свойствам. Свойство *Container* действует аналогично, но позволяет не только считывать, но и изменять контейнер ЭУ.

Свойство *Tag*. Это свойство предназначено для хранения любых дополнительных данных, необходимых пользователю:

Text.Tag = "Ввод имени".

Вопросы для самоконтроля

1. Для чего предназначены элементы управления *Кнопка, Флажок, Поле, Переключатель, Список, Поле со списком, Выключатель, Счетчик, Полоса прокрутки, Подпись, Рисунок*?
2. Как разместить на рабочей поверхности элемент управления?
3. Для чего предназначен режим конструктора?
4. Как войти в режим конструктора?
5. Как можно изменить свойства элементов управления?
6. Как изменить параметры шрифта элементов управления?
7. Что определяет свойство *Enabled*?
8. Как задать позицию элемента управления на экране?
9. Для чего предназначено свойство *Name*?
10. Для чего предназначено свойство *Capture*?

Практическая работа к теме № 5

1. Работа с элементом *Подпись*.

Постановка задачи. Элемент *Подпись* позволяет выразительно, информативно оформить документ. Создайте элемент *Подпись* "Элемент управления Подпись" шрифтом со свойствами: Arial, 14 пт, полужирный курсив, кириллица, темно-синего цвета. Исследуйте другие свойства этого элемента.

Выполнение:

- 1.1. На вкладке *Разработчик* в группе *Элементы управления* выберите кнопку *Вставить* и щелкните мышью по элементу *Подпись*. Установите этот элемент управления на рабочем поле документа.
- 1.2. Щелкните правой кнопкой мыши по появившемуся элементу управления и выберите команду *Свойства*.
- 1.3. Установите в появившемся окне *Свойства*:
 - заголовок (свойство *Caption*): "Элемент управления Подпись";
 - шрифт (свойство *Font*): Arial, 14 пт, полужирный курсив, набор символов: кириллица;
 - цвет темно-синий (свойство *ForeColor*).
2. Работа с элементом *Изображение*.

Постановка задачи. Этот элемент позволяет не только проиллюстрировать текст документа, но и сделать его "говорящим", интерактивным, что недоступно бумажной технологии.

Создайте **Изображение**, иллюстрирующий успешную договоренность, по щелчку, на котором появится окно сообщений с информацией: "Мы пришли к соглашению!"

Выполнение:

- 2.1. Установите на рабочем поле документа элемент **Изображение**;
- 2.2. Вызовите окно **Свойства** и выберите для свойства **Picture** соответствующий рисунок;
- 2.3. Для свойств:

PictureSizeMode выберите значение 3- **fmPictureSizeModeZoom** (без искажений);

BackColor выберите зеленый цвет;

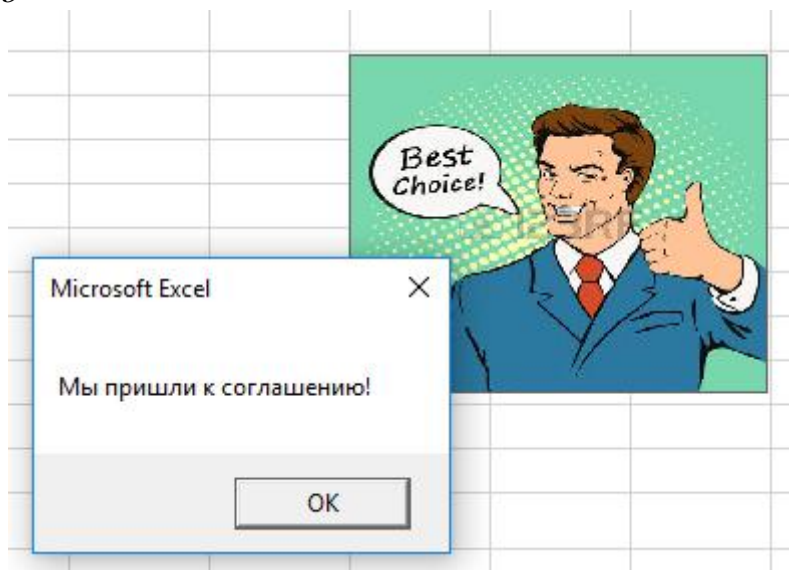
PictureAlignment выберите значение **fmPictureAlignmentCenter** (расположение по центру);

- 2.4. Щелкните правой кнопкой мыши на элементе управления **Изображение** и выберите команду **Исходный текст** и впишите код:

```
Private Sub Image1_Click()
```

```
MsgBox "Мы пришли к соглашению!"
```

```
End Sub
```



- 2.5. Проверьте выполнение программы, щелкнув мышью на рисунке.

3. Работа с элементом **Переключатель**.

Постановка задачи. Создайте на рабочей поверхности документа систему из двух **Переключателей**, цвет фона которых позволяет более наглядно показать, какой из переключателей выбран. Пусть при выборе первого переключателя его фон становится розовым, а второго – зеленым, а при выборе второго переключателя его фон становится розовым, а первого – зеленым.

Выполнение:

3.1. Щелкните правой кнопкой мыши на элементе управления **Переключатель**, выберите команду **Исходный текст** и впишите код:

```
Private Sub OptionButton1_Click()
```

'При выборе первого переключателя установим цвет фона для первого переключателя розовый 'а для второго – зеленый.

```
OptionButton1.BackColor = RGB(255, 128, 128)
```

```
OptionButton2.BackColor = RGB(0, 128, 64)
```

```
OptionButton2.Value = False ' Второй переключатель не выбран
```

```
EndSub
```

```
PrivateSubOptionButton2_Click()
```

'При выборе второго переключателя установим цвет фона для первого переключателя зеленый, а для второго – розовый цвет.

```
OptionButton1.BackColor = RGB(0, 128, 64)
```

```
OptionButton2.BackColor = RGB(255, 128, 128)
```

```
OptionButton1.Value = False ' Первый переключатель не выбран.
```

```
End Sub
```

3.2. Проверьте выполнение программы.

4. Работа с элементом **Поле** на примере расчета производительности труда.

Постановка задачи. Создайте систему элементов управления **Поле** на рабочем пространстве документа Word для расчета производительности труда в момент t , которая дается формулой:

$$f(t) = \frac{2}{3t + 4} + 3.$$

Выполнение:

4.1. Установите два поля ввода данных.

4.2. После двойного щелчка мышью на первом поле войдите в редактор VBA и запишите код:

```
Private Sub TextBox1_Change() 'При изменении содержимого первого поля (введении в него числа) произойдет выполнение программы.
```

```
TextBox2.Value = Format(2 / (3 * CDbI(TextBox1.Value) + 4) + 3, "##0.00") 'Значение, введенное в первое поле, проверится, является ли оно числом, введется в формулу, отформатируется до точности два знака после запятой и результат выведется во втором поле.
```

```
End Sub
```

4.3. Изменяя значения свойства объектов в окне Свойств, сделайте пояснения к форме и придайте ей привлекательный вид.

4.4. Выйдя из режима конструктора, проверьте работу формы.

5. Работа с элементом **Кнопка**.

Постановка задачи. Этот наиболее популярный и понятный элемент управления может запускать программу, заменяющую множество действий при работе вручную.

Пусть даны формулы функций спроса $q = \frac{p + 8}{p + 2}$ и предложения $S = p$, где q , S – количество товара, соответственно покупаемого и предлагаемого на продажу в единицах времени; p – цена товара (от 1 до 10).

Создайте **Кнопку**, при нажатии которой произойдет заполнение ячеек Excel и построится линейная диаграмма.

Выполнение:

5.1. Установите на рабочем поле документа элемент **Кнопка**.

5.2. Сделайте двойной клик левой кнопкой мыши на элементе управления **Кнопка** и впишите код:

```
Private Sub CommandButton1_Click() 'Нажатие кнопки приведет к запуску программы.
```

```
    'Заполнение ячеек.
```

```
    Range("A1") = "Цена товара" 'Заголовок первого столбца.
```

```
    Range("B1") = "Количество покупаемого товара" 'Заголовок первого столбца.
```

```
    Range("C1") = "Количество предлагаемого товара" 'Заголовок первого столбца.
```

```
    'Произведем автоподбор ширины столбцов.
```

```
    Range("A1:C1").Select 'Выделим ячейки с заголовками столбцов.
```

```
    Selection.Columns.AutoFit 'Выделенные столбцы соответственно расширятся.
```

```
    Range("A2") = 1 'В ячейку A2 введем начальное значение p, равное 1.
```

```
    Range("A3:A11") = "=R[-1]+1" 'Для диапазона ячеек A3:A11 содержимое ячейки, лежащей ниже на одну строку, равно содержимому вышележащей ячейки увеличенному на единицу.
```

```
    Range("B2:B11") = "=RC[-1]" 'Ячейки диапазона B2:B11 содержат значения ячеек, расположенных в той же строке, но на один столбец левее.
```

```
    Range("C2:C11") = "=(RC[-2]+8)/(RC[-2]+2)" 'Для ячеек диапазона C2:C11 используется относительная ссылка на ячейку, расположенную на два столбца левее в той же строке.
```

```
    Charts.Add 'Добавим диаграмму.
```

```
    With ActiveChart 'Установим свойства диаграммы.
```

```
        .ChartType = xlLineMarkers 'Выберем линейную диаграмму.
```

```
        .SetSourceData Source:=Worksheets("Лист1").Range("B1:C11"),
```

```
        PlotBy:=xlColumns 'Диаграмма будет получать данные для построения из столбцов диапазона B1:C11.
```

```
        .Location Where:=xlLocationAsObject, Name:="Лист1" 'Диаграмма будет помещена на первом листе.
```

```
    End With 'Конец установки свойств диаграммы.
```


With ActiveChart.Axes(xlValue) 'Установим свойства осей диаграммы.

.MinimumScale = -10 'Минимальное значение шкалы равно -10.

.MaximumScale = 10 'Максимальное значение шкалы равно 10.

.HasMajorGridlines = False 'Уберем линии сетки.

End With 'Конец установки свойств осей диаграммы.

End Sub 'Конец программы.

5.3. Проверьте выполнение программы.

6. Работа с элементом **Полоса прокрутки**.

Постановка задачи. С помощью элементов управления можно управлять совместной работой таких объектов, как ячейки Excel и диаграмма. Это позволяет оживить диаграмму, наглядно представить описываемый процесс в динамике.

Добавьте в предыдущем задании **Полосу прокрутки**, с помощью которой можно управлять графиком спроса и предложений. Чтобы управлять графиком, например, спроса, замените в программе для **Кнопки**:
Range("B2:B11") = "=RC[-1]" на *Range("B2:B11") = "=RC[-1]+R1C4"*. Это означает, что график спроса будет смещаться в зависимости от значения ячейки D1.

Выполнение:

6.1. Установите на рабочем поле документа элемент **Полоса прокрутки**.

6.2. Щелкните правой кнопкой мыши на элементе управления **Полоса прокрутки**, выберите команду **Просмотр кода** и впишите код:

Private Sub ScrollBar1_Change()

ScrollBar1.Min = -10 'Установка минимального значения Полосы прокрутки.

ScrollBar1.Max = 10 'Установка максимального значения Полосы прокрутки.

ScrollBar1.SmallChange = 1 'При щелчке Мыши на стрелке полосы прокрутки значение изменится на единицу.

ScrollBar1.LargeChange = 2 'При щелчке мыши между стрелками полосы прокрутки значение изменится на две единицы.

Range("D1") = ScrollBar1.Value 'Значение полосы прокрутки отразится в ячейке D1.

End Sub

6.3. Проверьте выполнение программы.

7. Работа с элементом **Список (Поле со списком)**.

Постановка задачи. Создайте **Список** или **Поле со списком**. для выбора из списка названий фирм (например, ЧФ Рубин, ЧП Григорьев А.П., ООО Салют и т.п.), которые выведутся в специальном поле и вставятся в документ Word.

Выполнение:

7.1. Установите на рабочем поле документа элементы *Кнопка*, *Поле* и *Список (Поле со списком)*.

7.2. Щелкните правой кнопкой мыши на элементе управления *Список (Поле со списком)*, выберите команду *Исходный текст* и впишите код:

```
Private Sub CommandButton1_Click()
```

'Применение инструкции позволяет *With* избежать многократного повторения имени объекта

```
With ListBox1
```

'Добавим в *Список* названия фирм. Кавычки ставятся с помощью функции *Chr()*, вызывающей соответствующий символ таблицы *ASCII* кодов.

```
.AddItem "ЧФ" &Chr(34) & " Рубин" & Chr(34)
```

```
.AddItem "ЧПГригорьевА.П."
```

```
.AddItem "ООО" &Chr(34) & " Салют" & Chr(34)
```

```
End With' Концециструкции With
```

```
End Sub
```

' Выбранный щелчком мыши элемент списка выведется в поле и вставится в документ.

```
Private Sub ListBox1_Click()
```

```
TextBox1.Value = ListBox1.Text
```

```
ActiveDocument.Range.InsertBefore ListBox1.Text
```

```
ActiveDocument.Range.InsertParagraphBefore
```

```
End Sub
```

7.3. Проверьте выполнение программы.

ТЕМА 6. ПОЛЬЗОВАТЕЛЬСКАЯ ФОРМА

Пользовательская форма. Свойства, методы и события форм. Вызов формы. Элементы управления Набор вкладок и Набор страниц.

В деятельности любой фирмы, учреждения часто требуется получить нужную информацию или обработать поступившую как можно оперативнее. Грамотно сконструированное диалоговое окно, снабженное целесообразными элементами управления, открывающие доступ к заранее упорядоченным данным, к предусмотренным механизмам их анализа позволит эффективно решить эту задачу.

Диалоговые окна (формы) с элементами управления составляют основу интерфейса современных программ. Они обеспечивают пользователю доступ к необходимой ему информации, средства ввода данных в компьютер, запуск программ их обработки и анализа.

Разрабатываемый интерфейс должен соответствовать установившимся соглашениям и быть как можно проще и понятнее для пользователя.

Разработка приложения в целом, и интерфейса, в частности, опирается на анализ процессов бизнеса и призвана повысить продуктивность работы.

6.1. Пользовательская форма

Форма (диалоговое окно, окно интерфейса программы) – это прямоугольная область экрана с элементами управления.

С помощью форм можно сообщать пользователю информацию и получать информацию от пользователя.

Для создания форм используются средства редактора VBA.

Чтобы создать пользовательскую форму (**UserForm**) нужно запустить редактор VBA и вызвать окно **UserForm** (см. Рис. 6 Окно **UserForm**) в открывшемся окне редактора VBA.

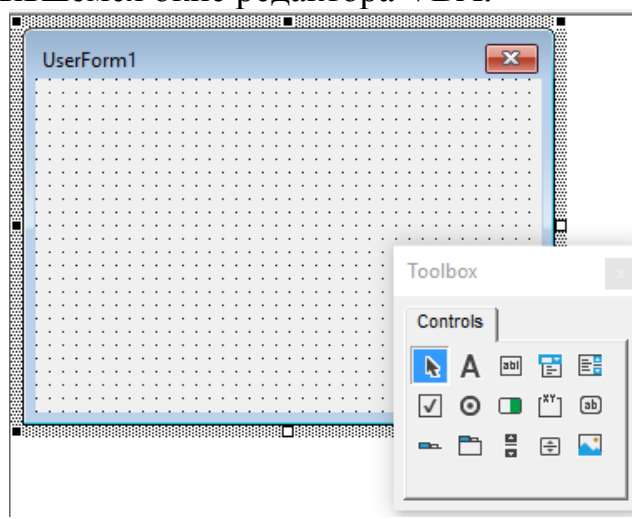


Рис. 6 Окно *UserForm*

Окно **UserForm** позволяет создавать в проекте окна диалога. Допускается рисование и просмотр элементов управления в форме. Возможен просмотр и изменение размеров сетки формы на вкладке (**Tools–Options...–General**). Кнопки **Controls** панели элементов **Toolbox** служат для вставки в форму элементов управления. Можно задать выравнивание элементов по линиям сетки (**Format – Align – toGrid**).

Способы вызова окна **UserForm**:

с помощью кнопки **Insert(Вставить) UserForm** панели инструментов (см. Рис. 7);

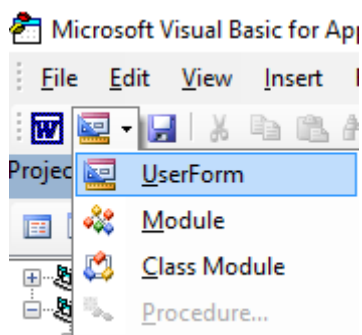


Рис. 7. Кнопка Вставить *UserForm*

командой меню: ***Insert (Вставить) – UserForm*** (см. Рис. 8);

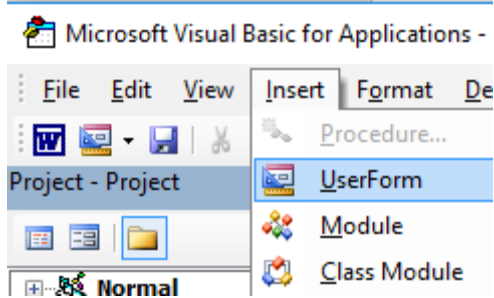


Рис. 8. Команда меню: ***Вставка – UserForm***

выбором команды ***UserForm*** из контекстного меню ***окна Проекта*** (см. Рис. 9, Рис. 10, Рис. 11).

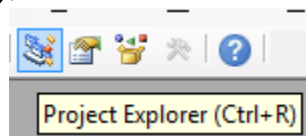


Рис. 9. Кнопка Окно проекта

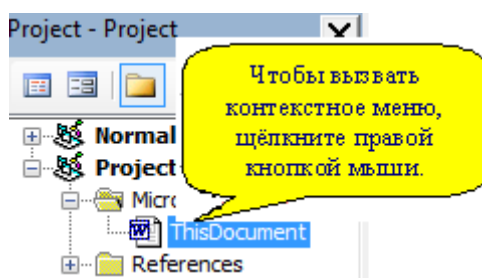


Рис. 10. Вызов контекстного меню

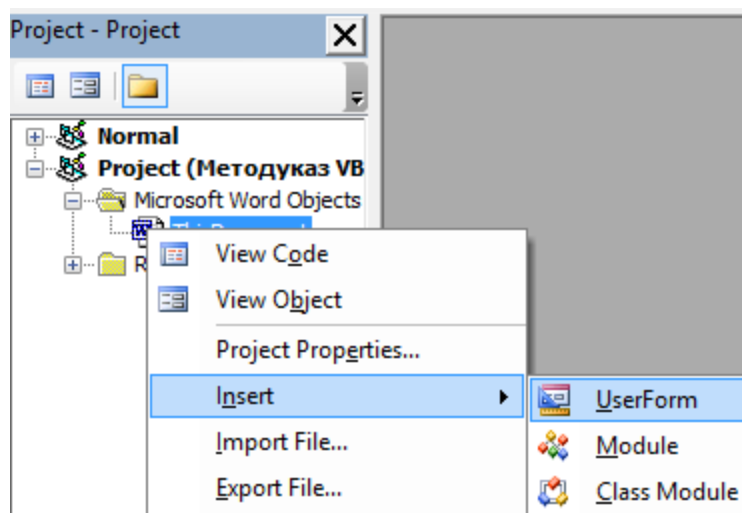


Рис. 11. Выбор команды *UserForm*

6.2. Свойства, методы и события форм

Форма, как объект VBA, обладает свойствами, методами и событиями. Некоторые свойства формы могут быть установлены программно, а некоторые выбраны в окне свойств.

Некоторые свойства форм:

<i>Name</i>	устанавливает имя, используемое при обращении к форме в программе;
<i>Caption</i>	устанавливает текст в строке заголовка формы.
<i>BorderStyle</i>	устанавливает тип границы: 0 или <code>fmBorderStyleNone</code> – нет линии границы; 1 или <code>fmBorderStyleSingle</code> – линия границы: одиночная линия;
<i>BorderColor</i>	устанавливает цвет линии границы;
<i>BackColor</i>	устанавливает цвет фона;
<i>Picture</i>	устанавливает рисунок фона;
<i>Left, Top</i>	Координаты верхнего левого угла формы;
<i>Height</i>	Высота формы;
<i>Width</i>	Ширина формы.

Наиболее часто используемые методы формы:

<i>Show</i>	Отображает форму на экране
<i>Hide</i>	Скрывает форму
<i>Move</i>	Изменяет местоположение и размер формы

Наиболее часто используемые события формы:

<i>Initialize</i>	Происходит во время создания формы, но до её загрузки
<i>Resize</i>	Происходит при изменении размеров формы
<i>Terminate</i>	Происходит при закрытии формы
<i>Click</i>	Происходит при щелчке мышью
<i>DblClick</i>	Происходит при двойном щелчке мышью

MouseDown	Происходит при нажатии кнопки мыши
MouseUp	Происходит при отпускании кнопки мыши
KeyDown	Происходит при нажатии клавиши
KeyUp	Происходит при отпускании клавиши
KeyPress	Происходит при нажатии клавиши

6.3. Вызов формы

Для просмотра формы в режиме выполнения (то есть, как она будет выглядеть в готовой программе) надо использовать команду **Run** и все ее элементы управления будут активными.

Вызов формы на исполнение с помощью кнопки, строки меню или сочетания клавиш невозможен.

Поэтому для вызова формы необходимо в основной программе указать команду **ИмяФормы.Show**, а кнопки и сочетания клавиш назначать именно этой основной программе.

Можно создать специальную программу вызова формы, например:

```
Sub Вызов_формы()
UserForm1.Show
EndSub.
```

Этот макрос можно запускать:

- с ленты предварительно установив на ней соответствующую команду (**Файл – Параметры – Настроить ленту – Выбрать команды: Макросы – Создать вкладку – Добавить – Normal.NewMacros.Вызов_формы**);

- с панели быстрого доступа предварительно установив на ней соответствующую команду (**Файл – Параметры – Панель быстрого доступа – Выбрать команды: Макросы –Добавить – Normal.NewMacros.Вызов_формы**);

- с рабочей области документа, установив на ней элемент управления **Кнопка** для запуска соответствующего макроса.

Щёлкнув дважды на этом элементе перейти в окно модуля, где записать:

```
Private Sub CommandButton1_Click()
UserForm1.Show
End Sub;
```

- создать с написанной выше инструкцией макрос и установить для его вызова сочетание клавиш и кнопку.

6.4. Элементы управления.

Набор вкладок и набор страниц

Формы с элементами управления составляют основу визуального программирования. Снабдить незаполненную форму элементами управления можно с помощью панели инструментов **Toolbox**. Для этого

нужно щелкнуть левой кнопкой мыши на значке нужного элемента управления и установить его на поверхности формы. Некоторые свойства управляющих элементов (размеры, положение) можно изменять непосредственно мышью, другие – с помощью окна свойств *Properties*. Элементы можно копировать, вырезать и вставлять. Для удобства размещения и выравнивания элементов используется сетка. Взаимное расположение и размеры устанавливаются с помощью меню *Format* (**Формат**) или контекстного меню.

По сравнению с элементами управления документа (например, Word в форме добавлены управляющие элементы **Набор вкладок** и **Набор страниц**, которые позволяют создавать набор страниц-вкладок.

Набор вкладок при переключении на другую страницу-вкладку не затрагивает другие элементы формы и изменения в состоянии других элементов можно задать только программно. **Набор страниц** при переключении на другую страницу-вкладку скрывает элементы на одной своей странице-вкладке и показывает элементы на другой.

Вопросы для самоконтроля

1. Что называется Формой?
2. В чем различие между пользовательской формой и диалоговым окном?
3. Как создать пользовательскую форму в VBA?
4. Какими способами можно вызвать окно **UserForm**?
5. Какие основные свойства, методы и события форм?
6. Как можно осуществить вызов формы на экран?
7. Как скрыть форму?
8. Для чего предназначено событие формы *Initialize*?
9. Как снабдить форму элементами управления?
10. Для чего предназначены элементы управления **Набор вкладок** и **Набор страниц**?

Практическая работа к теме № 6

1. Работа с элементами управления **Подпись**, **Поле**, **Кнопка**, применение функции *IsNumeric*, свойств *Enabled* и *Locked*, метода *SetFocus*. На примере создания формы для расчета стоимости с учетом НДС.

Постановка задачи. Создайте на примере форму для расчета стоимости с учетом НДС. Позаботьтесь, чтобы пользователь не мог изменять значения результата.

Выполнение:

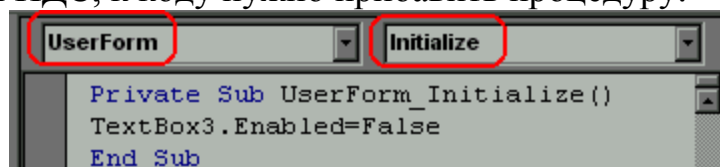
- 1.1. В редакторе Visual Basic в проект добавьте форму.
- 1.2. На форме разместите: три надписи, три поля ввода, две кнопки.
- 1.3. При помощи окна свойств задайте элементам управления свойства:
для формы установите заголовок "Расчёт стоимости";
для первой надписи – "Стоимость без учёта НДС";

- для второй надписи – "НДС";
- для третьей надписи – "Стоимость с учётом НДС";
- для кнопки 1: установите заголовок "ОК";
- для кнопки 2: установите заголовок "Выход".

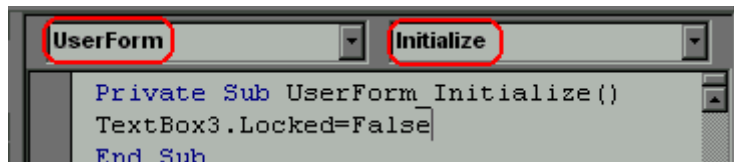
1.4. В модуле наберите код расчета стоимости с учетом НДС, производящегося после нажатия кнопки "ОК":

```
Option Explicit
Private Sub CommandButton1_Click()
'Опишем переменную для стоимости без НДС как число
с плавающей точкой двойной точности.
Dim СтоимостьБезНДС As Double
'Опишем переменную для величины НДС как длинное целое.
Dim НДС As Long
'Опишем переменную для стоимости без НДС как число
с плавающей точкой двойной точности.
Dim СтоимостьСНДС As Double
'Тот текст, который будет помещен в первое окно ввода при-
своится переменной СтоимостьБезНДС.
СтоимостьБезНДС = Cdbl(TextBox1.Text)
'Тот текст, который будет помещен во второе окно ввода (ко-
личество процентов),присвоится переменной НДС.
НДС = CLng(TextBox2.Text)
'Расчет стоимости с учетом НДС по формуле.
СтоимостьСНДС = СтоимостьБезНДС * (1 + НДС / 100)
'Отформатируем стоимость с НДС с помощью стандартного
числового формата Fixed, который отображает, по крайней мере, од-
ну цифру слева и две цифры справа от десятичного разделителя.
СтоимостьСНДС = Format(СтоимостьСНДС, "Fixed")
'Результат вычисления (значение переменной СтоимостьС-
НДС)-это тот текст, который отобразится в окне TextBox3.
TextBox3.Text = CStr(СтоимостьСНДС)
End Sub
```

1.5. Чтобы пользователь не мог изменить значения в поле ввода Стои-
мость с учётом НДС, к коду нужно прибавить процедуру:



1.6. Чтобы разрешить пользователю только выделять данные и копировать в буфер для дальнейшего использования, следует воспользо-
ваться свойством **Locked** , а не **Enabled**:



1.7. Если пользователь нажмёт кнопку ОК до того, как были введены данные, то произойдёт аварийное прерывание выполнения программы. На экране появится сообщение "Несовпадение типов". Чтобы исправить этот недостаток, добавим в программу после описания переменных выражения:

If Not IsNumeric(TextBox1.Text) Then 'Если в поле ввода *TextBox1* нет текста, то появится сообщение об ошибке без прерывания программы и выхода в редактор.

MsgBox "Ошибка в поле Стоимость без учета НДС"

'Только один элемент управления находится в фокусе, т.е. принимать данные ввода. Если пользователь щелкнул по кнопке, забыв ввести данные, то следует вернуть фокус в поле ввода с помощью метода *SetFocus*:

TextBox1.SetFocus

Exit Sub 'Немедленно завершает выполнение процедуры Sub, в которой появляется эта инструкция. Выполнение продолжается с инструкции, следующей за инструкцией, в которой вызывалась процедура *Sub*.

End If

1.8. Аналогично и для второго поля ввода:

If Not IsNumeric(TextBox2.Text) Then

MsgBox "Ошибка в поле НДС"

TextBox2.SetFocus

Exit Sub

End If

1.9. Проверьте работу формы.

2. Работа с элементами управления **Переключатель** и **Рамка**.

Постановка задачи. Создать форму, включающую в себя три поля ввода и надписи к ним, кнопку и рамку, объединяющую два переключателя для выбора суммы или произведения двух чисел. Учтите возможность ошибки неправильного ввода информации.

Выполнение:

2.1. После создания формы запишите код::

Option Explicit

Private Sub CommandButton1_Click()

Dim ПервоеЧисло As Double 'Описаниепервогочисла.

Dim ВтороеЧисло As Double 'Описание второго числа.

'Предотвращение ошибки неправильного ввода информации в первое окно ввода.

If Not IsNumeric(TextBox1.Text) Then

MsgBox "Введитечисло!"

TextBox1.SetFocus

Exit Sub

End If

'Предотвращение ошибки неправильного ввода информации 'во второе окно ввода.

If Not IsNumeric(TextBox2.Text) Then

MsgBox "Введите число!"

TextBox2.SetFocus

Exit Sub

End If

'Присвоение переменной ПервоеЧисло информации из первого окна ввода.

ПервоеЧисло = CDb(TextBox1.Text)

'Присвоение переменной ВтороеЧисло информации из второго окна ввода.

ВтороеЧисло = CDb(TextBox2.Text)

If OptionButton1.Value Then 'Если Переключатель в первом положении, то осуществляется сложение

TextBox3.Text = ПервоеЧисло + ВтороеЧисло

End If 'Конец блока *If*

If OptionButton2.Value Then 'Если Переключатель во втором положении, то осуществляется умножение

*TextBox3.Text = ПервоеЧисло * ВтороеЧисло*

End If 'Конец блока *If*

EndSub

PrivateSubUserForm_Initialize()

'При появлении формы Переключатель будет находиться в первом положении

OptionButton1.Value = True

'Пользователю запрещено изменять результат.

TextBox3.Enabled = False

End Sub

2.2. Проверьте выполнение программы.

3. Работа с элементами управления **Флажок** и **Выключатель**.

Постановка задачи. Создать форму для расчёта стоимости ренты, воспользовавшись встроенной функцией *PV()*.

Выполнение:

3.1. Создайте форму, на которой расположите четыре надписи и четыре поля ввода, кнопку и флажок.

3.2. Программа:

Option Explicit

'Пусть при запуске формы Флажок будет установлен.

Private Sub UserForm_Initialize()

```

CheckBox1.Value = True
End Sub
'При нажатии кнопки (после ввода данных) произойдет расчет.
Private Sub CommandButton1_Click()
'Опишем переменные.
Dim d As Double, p As Double, n As Double, t As Byte, r As Double
'Данные вносятся в соответствующие поля.
d = Cdbl(TextBox1.Text)
p = Cdbl(TextBox2.Text)
n = Cdbl(TextBox3.Text)
'Если флажок установлен, то начисление будет производиться
в конце периода.
If CheckBox1.Value Then
t = 0
Else
'Если флажок снят, то начисление будет производиться в начале пе-
риода.
t = 1
End If
'Используется встроенная функция PV
r = -PV(d / (12 * 100), n * 12, p, , CheckBox1)
'Результат выводится в четвертом окне в форматированном виде,
чтобы, по крайней мере, была одна цифра после разделителя и две
после.
TextBox4.Text = CStr(Format(r, "Fixed"))
End Sub

```

3.3. Добавьте защиту полей и проверьте работу формы.

3.4. Измените форму так, чтобы функцию **Флажка** выполнял **Выключатель**.

4. Работа с элементами управления **Полоса прокрутки** и **Счетчик**.

Постановка задачи. Создать форму, показывающую сколько денег надо откладывать ежемесячно, чтобы накопить необходимую сумму за определенное количество лет при данном проценте годовых? Использовать встроенную финансовую функцию **Pmt**. Обеспечить возможность выбирать величину процента годовых с помощью **Счетчика** или **Полосы прокрутки**.

Выполнение:

4.1. Постройте форму:

The screenshot shows a Windows application window titled "Размер платежей". It contains four input fields, each with a spinner control. The first field is labeled "ИСКАОМАЯ СУММА" and has the value 10000. The second field is labeled "ПРОЦЕНТАЯ СТАВКА, годовых" and has the value 12. The third field is labeled "СРОК, лет" and has the value 5. The fourth field is labeled "РАЗМЕР ЕЖЕМЕСЯЧНЫХ ВЫПЛАТ" and has the value 122,44. At the bottom of the form is a button labeled "ВЫЧИСЛИТЬ".

4.2. Запишите код:

Option Explicit

'При инициализации формы установим максимальное значение счетчика равное 100.

```
Private Sub UserForm_Initialize()  
ScrollBar1.Max = 10000000  
ScrollBar2.Max = 100  
ScrollBar3.Max = 100End Sub
```

'Расчеты будут производиться после нажатия кнопки

```
Private Sub CommandButton1_Click()
```

'Объявим все переменные: БЗ – искомая сумма (будущее значение), Ставка – процентная ставка, Срок – количество периодов начисления процентов, Результат – искомая величина – размер ежемесячных выплат.

```
Dim БЗAs Double, Ставка As Double
```

```
Dim СрокAs Double, Результат As Double
```

'Переменные примут значения соответствующих полей ввода данных.

```
БЗ = CDbI(TextBox1.Text)
```

```
Ставка = CDbI(TextBox2.Text)
```

```
Срок = CDbI(TextBox3.Text)
```

'Результат вычисляется с помощью встроенной функции Pmt().

```
Результат = -Pmt(Ставка / (12 * 100), Срок * 12, 0, БЗ)
```

'Отформатированное значение размера ежемесячных выплат выводится в поле TextBox4.

```
TextBox4.Text = CStr(Format(Результат, "Fixed"))
```

```
EndSub
```

'Выбираемые значения счетчиков отображается в соответствующих полях TextBox для расчетов.

```
Private Sub ScrollBar1_Change()
```

```
TextBox1.Text = CStr(ScrollBar1.Value)
```

```
End Sub
```

```
Private Sub ScrollBar2_Change()
```

```

    TextBox2.Text = CStr(ScrollBar2.Value)
End Sub
Private Sub ScrollBar3_Change()
    TextBox3.Text = CStr(ScrollBar3.Value)
End Sub

```

4.3. Проверьте работу формы.

5. Работа с элементами управления **Набор страниц**

Постановка задачи. Создать форму для анализа операций с элементарными денежными потоками, позволяющую в одном диалоговом окне, но на разных страницах рассчитывать будущую величина (*FV*) и начальное значение (*PV*) вкладов.

Выполнение:

5.1. В Excel создайте форму, содержащую элемент управления: **Набор страниц**.

5.2. На первой странице установите заголовок "*FV*", **Кнопку**, пять **Полей** и **Надписей** к ним: "Годовая процентная ставка", "Количество начислений в году", "Срок проведения операций (лет)", "Начальное значение PV", "Будущая величина FV".

5.3. Составьте код:

```
Option Explicit
```

```
Private Sub CommandButton1_Click()
```

```
Dim r As Double, m As Double, n As Double, pv As Currency, fv As Currency
```

```
    r = TextBox1.Text
```

```
    m = TextBox2.Text
```

```
    n = TextBox3.Text
```

```
    pv = TextBox4.Text
```

```
    fv = Application.fv(r / m, n * m, , -pv)
```

```
    TextBox5.Text = fv
```

```
End Sub
```

5.4. На второй странице установите заголовок "*PV*", **Кнопку**, пять **Полей** и **Надписей** к ним: "Годовая процентная ставка", "Количество начислений

в году", "Срок проведения операций (лет)", "Будущая величина FV", "Начальное значение PV".

5.5. Составьте код:

```
Private Sub CommandButton2_Click()  
Dim r As Double, m As Double, n As Double, fv As Currency, pv As Cur-  
rency  
  
    r = TextBox6.Value  
    m = TextBox7.Text  
    n = TextBox8.Text  
    fv = TextBox9.Text  
    pv = Application.pv(r / m, n * m, , -fv)  
    TextBox10.Text = pv  
  
End Sub
```

5.6. Проверьте выполнение программы.

ТЕМА 7. ПРОЦЕДУРЫ И ФУНКЦИИ ЯЗЫКА VBA. СТРУКТУРА И ТИПЫ ПРОЦЕДУР

Программный проект. Модули. Процедуры. Органи-
зация данных в программе. Массивы.

Цель офисного программирования: создание сложного электронного документа или системы документов. При этом используются как мощные стандартные средства MS Office, так и настройка документов с помощью VBA на специфику реально решаемых бизнес-задач.

7.1. Программный проект

Для создания документа (системы документов) используется специальное приложение *Редактор Visual Basic (Сервис – Макрос – Редактор Visual Basic)*.

Создаваемый документ строится на основе автоматически подключаемого каркаса документов – совокупности объектов, необходимых для построения документов данного типа. Их можно изменять в зависимости от решаемой задачи (см. *Окно свойств*). Для подключения к документу объектов другого приложения (например, можно подключить библиотеку объектов Excel к документу Word) используется команда *Сервис – Ссылки*. В появившемся диалоговом окне *Ссылки-Project* в списке *Доступные ссылки* можно выбрать соответствующую библиотеку объектов (см. Рис. 13).

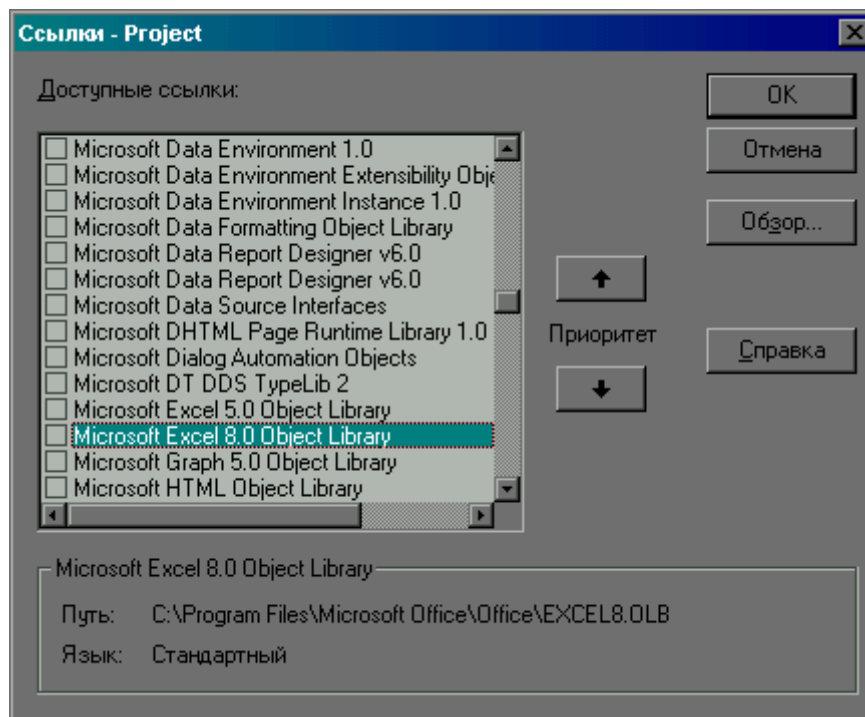


Рис. 12. Диалоговое окно *Ссылки-Project*

Архитектура документа определяется приложением, в котором документ создаётся (объект *Application*). Программный проект создаётся как часть архитектуры документа.

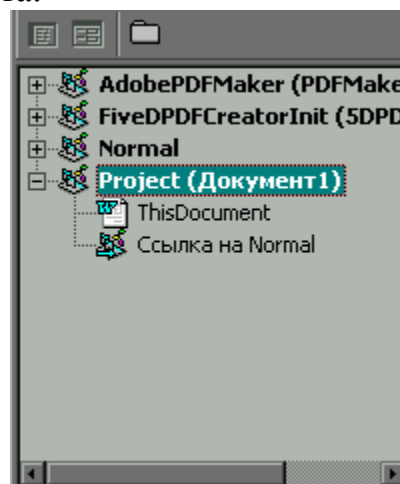


Рис. 13. Окно проекта

Для управления объектами документа записывается программа автоматически (*Macrorecorder*) или вручную (см. *Окно программы*).

Программа состоит из модулей.

7.2. Модули

Модуль – это файл с текстом программы, вставленный в документ. Программа на языке VBA состоит из одного или нескольких модулей.

Модули могут быть следующих типов:

модули, связанные с объектами приложения, реагирующие на (модули – обработчики событий);

программные (стандартные)
модули классов, создаваемые программистом;
модули макросов, создаваемые *Macrorecorder*.

Модули-обработчики событий всегда связаны с объектами, которые реагируют на события.

В них следует помещать только те процедуры, и функции, которые непосредственно обрабатывают события.

Программные (стандартные) – основной вид модулей. Создается программистом.

Большую часть всех процедур и функций следует размещать в стандартные модули.

Практика показывает, что целесообразнее создавать не один большой модуль, а несколько маленьких. В один модуль следует помещать набор функций, совместно вызывающих друг друга, объединённых общей темой и объединённых общей функциональной направленностью.

Реально всегда выделяется главный модуль, где сосредоточено описание глобальных переменных.

Модуль класса – специальный тип модуля, который используется при создании классов, разрабатываемых программистом для решения своих задач.

В VBA предусмотрена возможность создания пользовательских объектов, которые являются экземплярами классов. Все объекты одного и того же класса используют одинаковые методы в ответ на одинаковые сообщения. Классы конструируются в модулях классов, которые создаются командой **Вставка – модуль класса** и записываются в появившемся окне. Нажав клавишу **F4**, можно выбрать имя класса.

Имя модуля класса является именем класса объектов.

В разделе описания модуля объявляются переменные уровня модуля, которые используются как "значения свойств".

Класс инициализируется при помощи процедуры:

Class_Initialaize.

Имена свойств, значениями которых являются числовые данные объявляются при помощи процедуры ***Property Let***.

Процедурой ***Property Set*** объявляются имена свойств, значениями которых являются объекты.

При помощи процедуры ***Property Get*** устанавливается возможность считывания значений свойств. Методы создаются при помощи обычных процедур и функций.

Отличие модулей макросов от стандартных модулей в том, что они создаются автоматически при вызове ***Macrorecorder***.

В Excel каждый новый макрос записывается в новый модуль.

В Word все созданные ***Macrorecorder*** макросы размещаются в модуле с именем ***NewMacros***.

При создании макроса требуется указать в модуль, какого проекта он должен быть записан:

связанного с документом;

Normal.dot.

7.3. Процедуры

Процедура – это совокупность операторов, выполняющих определённые действия.

Различают процедуры (подпрограммы) и процедуры-функции.

Процедуры (подпрограммы) имеют стандартное оформление:

Public {[*Private*],[*Static*]}*Sub* Имя_Процедуры(Список параметров)
тело процедуры

End Sub

Public – глобальная процедура, доступна для всех других процедур во всех модулях проекта;

Private – процедура модуля, доступна для всех других процедур в данном модуле;

Static – служебное слово, которое говорит о том, что локальные переменные процедуры сохраняются в промежутках времени между вызовами этой процедуры

Name – имя процедуры, удовлетворяющее стандартным правилам написания имен в VBA. Имя процедуры обработки события состоит из имени объекта и имени события.

Список аргументов – список формальных параметров (аргументов) процедуры *Sub* (имен переменных, которые должны быть переданы в процедуру при обращении к ней).

Синтаксис элемента структуры **Список аргументов**:

[*Optional*] [*ByVal*, *ByRef*] [*ParamArray*] Имяпеременной *As* Тип

Optional – ключевое слово, указывающее, что аргумент не является обязательным. Все аргументы, описанные как **Optional** должны быть типа **Variant**

ByVal – указывает на то, что этот аргумент передается по значению.

ByRef – указывает на то, что этот аргумент передается по ссылке на его адрес в памяти. Описание **ByRef** используется по умолчанию.

ParamArray – используется только в качестве последнего элемента списке аргументов для указания о том, что конечным аргументом в списке параметров процедуры является описанный как типа **Variant** – массив значений, т.е. это слово позволяет задавать произвольное количество аргументов в процедуре.

Процедуры-функции имеют вид:

Function Имя_Функции [(Список аргументов)] *As* тип тело функции
End Function.

7.4. Организация данных в программе. Массивы.

Модули помещают в себе процедуры, включающие операторы, обрабатывающие данные. По способу организации различают простые и структурированные данные.

Простые данные состоят из одного элемента. К ним относятся константы и простые переменные. В памяти ЭВМ для них выделяется по одной ячейке.

Структурированные данные состоят из нескольких связанных друг с другом элементов. К ним относятся массивы, записи, файлы.

Массив – это поименованное упорядоченное множество однородных данных определённого типа. Каждый элемент массива имеет свой порядковый номер(индекс).

Запись – это поименованное упорядоченное множество разнородных данных.

Файл – это поименованное множество данных на внешней памяти.

Для их размещения в памяти ЭВМ выделяется массив ячеек памяти.

Все данные, обрабатываемые программой, должны быть в ней описаны.

В описаниях указываются имена и типы данных. Эта информация необходима для размещения данных в ячейках памяти и организации их обработки.

Массив имеет имя, а его элементы различаются номерами (индексами). Начало индексации массива с 0 или 1 определяется параметрами инструкции **Option Base**. Если не указано **Option Base 1**, нижняя граница индексов массива равняется нулю.

Массивы могут быть динамическими и статическими.

Статические массивы (массивом фиксированного размера) – это массивы с заданным размером. Количество размерностей и границы статических массивов определяются при объявлении.

Массив с переменным размером называется динамическим.

Количество размерностей и границы динамического массива определяются впоследствии в коде.

Примеры описания статических массивов:

Одномерный массив: $A=(1\ 4\ 5\ 6\ 1\ 8\ 6\ 5\ 3\ 7)$ -массив из 10 целых чисел

Описание: $Dim\ A(9)\ As\ Integer$.

Элементы: $A(0)=1; A(1)=4; A(2)=5; A(3)=6; A(4)=1; A(5)=8; A(6)=6; A(7)=5; A(8)=3; A(9)=7$.

Двумерный массив из 3 строк и 3 столбцов (первый аргумент представляет строки, а второй – столбцы):

$$B = \begin{matrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0. \end{matrix}$$

Описание: $Dim\ B(2, 2)\ As\ Boolean$.

Элементы: $B(0, 0)=1$; $B(0, 1)=0$; $B(0, 2)=1$; $B(1, 0)=1$; $B(1, 1)=0$; $B(1, 2)=0$; $B(2,0)=0$; $B(2,1)=0$; $B(2, 2)=0$.

Двумерный массив из 3 строк и 2 столбцов:

Описание: *Dim C(1 to 3, 1 to 2)* — индекс массива начинается с 1.

Для описания динамического массива используются инструкции с пустыми скобками.

Пример описания динамического массива:

Dim A() As Integer

Для изменения числа размерностей, определения числа элементов и задания верхних и нижних границ индексов для каждой размерности используется инструкция **ReDim**. Инструкцию **ReDim** можно применять для изменения динамического массива столько раз, сколько потребуется.

ReDim(N, N) As Integer

При каждом применении **ReDim** данные, содержащиеся в массиве, теряются. Для увеличения размера массива с сохранением его содержимое используется инструкция **ReDim Preserve**.

Функция **Array** позволяет задать список значений для занесения в массив. Например:

Dim Неделя, День

Неделя = Array("Пн", "Вт", "Ср", "Чт", "Пт", "Сб", "Вс")

' Предполагается, что нижняя граница индексов массива установлена равной 1

' (с помощью инструкции Option Base).

День = Неделя(2) ' День содержит "Вт".

День = Неделя(4) ' День содержит "Чт".

Вопросы для самоконтроля

1. Какова цель офисного программирования?
2. Что называется программным проектом?
3. Что называется программным модулем?
4. Какие типы модулей применяются в VBA?
5. Почему считается целесообразным создавать не один большой модуль, а несколько малых модулей?
6. Что называется модулем класса?
7. В чем различие между стандартным модулем и макросом?
8. В чем различие между процедурой-подпрограммой и процедурой-функцией?
9. В каких случаях применяют массивы?
10. В каких случаях применяют инструкцию **Option Base 1**?
11. В чем различие между статическими и динамическими массивами?
12. В чем различие между циклом с предусловием и циклом с постусловием?

Практическая работа к теме № 7

Часто деловая информация приводится в виде списков (товаров, клиентов и т.п.). Удобно выводить такой список в диалоговом окне. Для заполнения таких списков используются массивы.

1. Использование массивов для заполнения одноколоночного списка.

Постановка задачи. Создать одноколоночный *Список*, заполняемый программно.

Выполнение:

1.1. Установите на рабочем поле Документа Word элементы *Кнопку* и *Список*.

1.2. После двойного щелчка мышью на *Кнопке* войдите в редактор VBA и запишите код:

'После нажатия *Кнопки* произойдет заполнение *Списка* массивом.

```
Private Sub CommandButton1_Click()
```

```
With ListBox1
```

```
'Заполнение Списка массивом.
```

```
.List = Array("Киев", "Харьков", "Богодухов")'
```

.ListIndex = 1 Возвращает номер текущего элемента списка. (Нумерация списка начинается с нуля).

```
End With
```

```
End Sub
```

1.3. Проверьте работу программы.

2. Использование массивов для заполнения двух колоночного списка.

Постановка задачи. Создать двухколоночный *Список*, заполняемый программно.

Выполнение:

2.1. Установите на рабочем поле Документа Word элементы *Кнопку* и *Список*.

2.2. После двойного щелчка мышью на *Кнопке* войдите в редактор VBA и запишите код:

' После нажатия Кнопки произойдет заполнение списка

```
Private Sub CommandButton1_Click()
```

' Объявление массива из трех строк и двух столбцов со значениями типа строка

```
Dim A(2, 1) As String
```

```
'Заполнение элементов массива
```

```
A(0, 0) = "Завтрак"Первая строка, левый столбец.
```

```
A(0, 1) = "Съест"Первая строка, второй столбец.
```

```
A(1, 0) = "Обед"Вторая строка, левый столбец.
```

```
A(1, 1) = "Половину"Вторая строка, второй столбец.
```

```
A(2, 0) = "Ужин"Третья строка строка, левый столбец.
```

```
A(2, 1) = "Врагу"Третья строка строка, второй столбец.
```

Инструкция *With* позволяет избежать многократного повторения имени объекта.

```
With ListBox1
```

```
.ColumnCount = 2 'ColumnCount устанавливает количество столбцов списка.
```

```
.List = A 'Заполнение списка массивом A.
```

```
End With 'Конец инструкции With.
```

```
End Sub
```

2.3. Проверьте работу программы.

3. Трансформация элементов массива.

Постановка задачи. Элементы массива можно трансформировать в нужных целях. Пусть дан массив фамилий, имен и отчеств клиентов ("Иванов Петр Андреевич"), но в документе требуется сокращенная запись ("П. А. Иванов"). Создать одноклончатый **Список**, заполняемый программно Ф.И.О. клиентов. По щелчку мыши на выбранном элементе списка в документ должна вставляться сокращенная запись.

Выполнение:

3.1. Установите на рабочем поле Документа Word элементы **Кнопку** и **Список**.

3.2. После двойного щелчка мышью на **Кнопке** войдите в редактор VBA и запишите код:

```
Option Explicit
```

```
Dim k1 As Long, k2 As Long
```

```
Dim s1 As String, s2 As String, s3 As String, t As String
```

```
Dim N(6) As String, l(6) As String
```

```
Dim myRange As Range
```

```
Private Sub CommandButton1_Click()
```

```
'Пусть дан массив имен клиентов.
```

```
N(0) = "Иванов Петр Андреевич"
```

```
N(1) = "Александров Николай Михайлович"
```

```
N(2) = "Рогодин Владимир Никифорович"
```

```
N(3) = "Сидоренко Роман Станиславович"
```

```
N(4) = "Куприянова Наталья Константиновна"
```

```
N(5) = "Яковенко Любовь Петровна"
```

```
ListBox1.List = N 'Заполнение списка массивом N.
```

```
End Sub
```

'Выбранный щелчком мыши элемент списка вставится в документ в измененном виде.

```
Private Sub ListBox1_Click()
```

ActiveDocument.Range.InsertParagraphBefore 'Вставка пустого параграфа, который впоследствии заполнится выбранным элементом списка.

```
t = ListBox1.Text
```

k1 = InStr(t, " ") 'Указывается позиция первого вхождения разделителя (пробела) внутри выбранной строки.

k2 = InStr(k1 + 1, t, " ") 'Указывается позиция нового вхождения разделителя (пробела) внутри выбранной строки.

s1 = Mid(t, 1, k1 - 1) 'Выбираются символы с первого по последний перед разделителем и присваиваются строковой переменной s1.

s2 = Mid(t, k1 + 1, 1) 'Выбирается символ после первого вхождения разделителя и присваиваются строковой переменной s2.

s3 = Mid(t, k2 + 1, 1) 'Выбирается символ после второго вхождения разделителя и присваиваются строковой переменной s3.

'Отдельные части выбранной строки (подстроки) перегруппируются и между ними вставляются знаки препинания с пробелами, так чтобы запись приобрела нужный вид.

ActiveDocument.Range.InsertBefore s2 + ". " + s3 + ". " + s1

End Sub

3.3. Проверьте работу программы.

ТЕМА 8. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Оператор условного перехода If ... Then ... Else.

Оператор выбора варианта Select Case

Характерной чертой большинства экономических задач является множественность возможных решений. Управление экономическими объектами заключается в выборе наиболее приемлемого решения в зависимости от выполнения определенных условий. Заранее предусмотреть возможные варианты, определить порядок выполнения программы можно, используя, управляющие операторы (структуры управления).

8.1. Оператор условного перехода If ... Then ... Else

Оператор условного перехода (условный оператор) задаёт выполнение определённых групп инструкций (кодов) в зависимости от итога проверки условия.

Существует два вида условного оператора:

1. *If <условие> Then (не полный оператор) <Код> End If.*

2. *If <условие 1> Then <Код – 1>*

ElseIf <условие 2> Then <Код – 2>

...

ElseIf <условие N> Then <Код – N>

Else <Код – N + 1> End If.

Здесь: *<условие >* – логическое выражение, принимающее значения **True** или **False**, т.е. результат выражения всегда имеет булевский тип.

If (если), **Then** (то), **Else** (иначе) и **ElseIf** (еще если)- ключевые слова языка;

< *Kod* > – любые операторы, в том числе и условные.

Порядок выполнения:

если значение выражения – **True** (условие соблюдается), то выполняется *Kod -1*,

если **False**(условие не соблюдается), то выполняется *Kod -2*.

Затем в обоих случаях управление передается дальше, к следующему оператору программы, следующему после оператора **End If**.

Если ветвь **Else** отсутствует (не полный оператор) и условие не соблюдается, то никакие действия не выполняются.

8.2. Оператор выбора варианта **Select Case**

Оператор выбора **Select Case** производит разбор случаев и в зависимости от значений анализируемого выражения выбирает и использует одну из последовательностей операторов.

Когда требуется проверить только одно условие, предпочтительнее использовать оператор выбора варианта **Select Case**.

Синтаксис:

Select Case выражение

[**Case** списокВыражений-n

[инструкции-n]]

[**Case Else**

[инструкции_else]]

End Select

Здесь **выражение** может быть произвольным выражением (например, переменной).с числовым или строковым значением. **Инструкции-n** – инструкции, выполняемые в случае совпадения **выражения** с любым компонентом списка **списокВыражений-n**. Необязательная инструкция **Case Else** выполняется, если **Select Case** не находит подходящего значения ни в одной из инструкций **Case**.

Вопросы для самоконтроля

1. Для чего предназначен оператор условного перехода **If ... Then ... Else**?
2. Каков порядок выполнения неполного условного оператора?
3. Каков порядок выполнения полного условного оператора?
4. Для чего предназначен оператор выбора варианта **Select Case**?
5. Каков порядок выполнения оператора выбора варианта **Select Case**?
6. Когда предпочтительнее использовать оператор выбора варианта **Select Case**?

Практическая работа к теме № 8

1. Элементы управления **Выключатель**, **Флажок**, **Переключатель** предполагают использование ветвлений.

Постановка задачи. Установите на рабочем поле документа элемент **Выключатель**, который устанавливает определенный формат шрифта (полужирный, 16 пт,), например, в заголовке самого элемента управления, и **Флажок**, заголовок которого указывает явно снят он или установлен.

Выполнение:

1.1. Вызовите панель инструментов Элементы управления и установите на рабочем поле документа элементы управления **Выключатель** и **Флажок**.

1.2. Щелкните правой кнопкой мыши на элементе управления **Выключатель**, выберите команду **Исходный текст** и впишите код:

```
Private Sub ToggleButton1_Click()  
If ToggleButton1.Value = True Then  
ToggleButton1.Font.Bold = True  
ToggleButton1.Caption = "Полужирный"  
ToggleButton1.Font.Size = 16  
Else  
ToggleButton1.Font.Bold = False  
ToggleButton1.Caption = "Обычный"  
ToggleButton1.Font.Size = 16  
End If  
End Sub
```

1.3. Щелкните правой кнопкой мыши на элементе управления **Флажок**, выберите команду **Исходный текст** и впишите код:

```
Private Sub CheckBox1_Click()  
If CheckBox1.Value = True Then  
CheckBox1.Caption = "Флажок установлен"  
Else  
CheckBox1.Caption = "Флажок снят"  
End If  
End Sub
```

1.4. Проверьте выполнение программ.

2. Число прописью. Использование оператора выбора **Select Case**.

Постановка задачи. Во многих платежных документах (платежных ведомостях и др.) наряду с числовыми выражениями сумм, дат и т.п. применяются и записи прописью. Создать процедуру, выводящую в специальное поле строку числа прописью при вводе в другое поле числового значения от 0 до 10.

Выполнение:

2.1. Установите на рабочем поле Документа Word элементы **Кнопку** и два **Поля**.

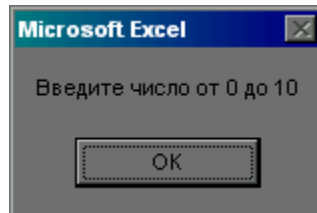
Число прописью

2.2. После двойного щелчка мышью на **Кнопке** войдите в редактор VBA и запишите код

Dim Число As Integer 'Объявление числовой переменной переменной для ввода в первое поле.

Dim Слово As String 'Объявление строковой переменной переменной для ввода 'во второе поле.

If Not IsNumeric(TextBox1.Text) Then 'Проверка правильности ввода числа.



MsgBox "Введите число от 0 до 10" 'В случае ошибки ввода появится окно сообщений.

Exit Sub 'Инструкция, которая немедленно завершает выполнение процедуры Sub, в которой появляется эта инструкция. Выполнение продолжается с инструкции, следующей за инструкцией, в которой вызывалась процедура Sub.

End If 'Конец блока If.

Число = CInt(TextBox1.Text) 'Текст, который вводится в первое поле ввода, преобразуется (если возможно) в данные типа Integer и присваивается переменной Число.

Select Case Число 'Начало инструкции, которая выполняет одну из нескольких инструкций в зависимости от значения переменной Число.

Case "0": Слово = "ноль" ' Если переменная Число примет значение 0, то присвоить 'переменной Слово значение "ноль", которое отобразится во втором поле.

Case "1": Слово = "один"

Case "3": Слово = "три"

Case "4": Слово = "четыре"

Case "5": Слово = "пять"

Case "6": Слово = "шесть"

Case "7": Слово = "семь"

Case "8": Слово = "восемь"

Case "9": Слово = "девять"

Case "10": Слово = "десять"

Case Else 'В иных случаях присвоить переменной Слово значение "Я считаю только до десяти".

Слово = "Я считаю только до десяти"

End Select 'Конец инструкции

TextBox2.Text = Слово 'Отобразить значение переменной Слово во втором поле.

2.3. Проверьте работу программы.

2.4. Измените текст программы так, чтобы при вводе в первое поле числа от 1 до 7, во втором выведется день недели прописью.

2.5. Измените текст программы так, чтобы при вводе в первое поле числа от 1 до 12, во втором выведется название месяца прописью.

3. Применение ветвящихся процессов для создания пользовательских функций.

Постановка задачи.Предлагается следующая системаскидок:

Количество продаваемого товара	Скидка на цену товара
от 0 до 9	0%
от 10 до 29	2%
от 30 до 49	5%
от 50 до 100	7%
Более 100	10%

Создать пользовательскую функцию для расчета скидки на товар в зависимости от его количества, используя инструкцию *Select Case*.

Выполнение:

Напишите код пользовательской функции Скидка_на_товар. Для этого:

— откройте редактор VBA;

— в модуле напишите функцию:

Public Function Скидка_на_товар(Цена As Currency, Количество As Integer) As Currency

Select Case Количество

Case 0 To 9

Скидка_на_товар = 0

Case 10 To 29

*Скидка_на_товар = CCur(Цена * 2 / 100)*

Case 30 To 49

*Скидка_на_товар = CCur(Цена * 5 / 100)*

Case 50 To 100

*Скидка_на_товар = CCur(Цена * 7 / 100)*

Case Is > 100

*Скидка_на_товар = CCur(Цена * 10 / 100)*

End Select End Function

Проверьте выполнение функции.

4. Использование готовой пользовательской функции при создании другой пользовательской функции.

Постановка задачи. Дана пользовательская функция *Скидка_на_товар(Цена, Количество)* для расчета скидки на цену товара в зависимости от его количества (см. предыдущий пункт).

Требуется создать пользовательскую функцию для расчета стоимости товара с учетом скидки.

Выполнение:

Напишите код пользовательской функции *Стоимость_со_скидкой* в том же модуле, что и для предыдущего пункта, напишите функцию:

Public Function Стоимость_со_скидкой(Цена As Currency, Количество As Integer) As Currency

*Стоимость_со_скидкой = (Цена - Скидка_на_товар(Цена, Количество)) * Количество*

End Function

Проверьте выполнение функции.

5. Применение ветвящихся процессов для расчета стоимости покупки с использованием инструкции *IfThenElse*.

Постановка задачи. Для увеличения спроса на товар предлагается следующая система цен, если покупатель берет напитки упаковками:

<i>Количество упаковок (одна упаковка – 9)</i>	<i>Цена упаковки товара 1 (грн)</i>	<i>Цена упаковки товара 2 (грн)</i>
1 – 2	75	70
3 – 6	73	68
7 – 9	71	66
10 и более	70	65

При этом, если количество товара меньше упаковки, то цена товара 1 равна 8 грн, а товара 2 – 7,5 грн.

Создать форму для расчета стоимости покупки товаров.

Выполнение:

5.1. Создайте форму, содержащую элементы управления: **Кнопку** для запуска программы. **Поля** для ввода количества соответствующих напитков и вывода конечного результата и соответствующие надписи к элементам управления.

5.2. Запишите код:

Option Explicit

Private Sub UserForm_Initialize()

'При инициализации формы в полях ввода установятся нулевые значения.

TextBox1.Value = 0

TextBox2.Value = 0

End Sub

Private Sub CommandButton1_Click()

Dim NT1 As Long 'Числоупаковоктовара 1.

Dim MT1 As Long 'Количествотовара 1меньшееупаковки.

Dim kT1 As Double 'Цена товара 1 в рознице.

Dim NT2 As Long 'Число упаковок товара 2.

Dim MT2 As Long 'Количество товара 2 меньшее упаковки.

Dim kT2 As Double 'Цена товара 1 в рознице.

Dim RT1 As Double 'Количество товара 1.

Dim RT2 As Double 'Количество товара 2.

Dim R As Double 'Общая стоимость покупки.

RT1 = TextBox1.Value 'Ввод количества товара 1 в первое поле.

RT2 = TextBox2.Value 'Ввод количества товара 2 во второе поле.

NT1 = Fix (Cdbl (RT1 / 9)) 'Определение числа упаковок товара 1.

*MT1 = CLng (RT1 - 9 * NT1) 'Определение числа товара 1, вне упаковки.*

NT2 = Fix (Cdbl (RT2 / 9)) 'Определение числа упаковок товара 2.

*MT2 = CLng (RT2 - 9 * NT2) 'Определение числа товара 2, вне упаковки.*

'Перебор вариантов оплаты товара 1 в зависимости от числа упаковок.

If NT1 < 3 Then '1 – 2 упаковки.

kT1 = 75

Else 'Иначе.

If NT1 < 7 Then '3 – 6 упаковок.

```

kT1 = 73
Else 'Иначе.
If NT1 < 10 Then '7 – 9 упаковок.
kT1 = 71
Else 'Иначе.
'Более 10 упаковок.
kT1 = 70
End If
End If
End If
'Перебор вариантов оплаты товара 2.
If NT2 < 3 Then '1 – 2 упаковки.
kT2 = 7
Else 'Иначе.
If NT2 < 7 Then '3 – 6 упаковок.
kT2 = 68
Else 'Иначе.
If NT2 < 10 Then '7 – 9 упаковок.
kT2 = 66
Else 'Иначе.
'Более 10 упаковок.
kT2 = 65
End If
End If
End If
R = NT1 * 9 * kT1 + MT1 * 8 + NT2 * 9 * kT2 + MT2 * 75 'Расчет об-
щей стоимости.
TextBox3.Value = R 'Вывод значения общей стоимости в третьем
поле.
End Sub

```

5.3. Проверьте работу программы.

ТЕМА 9. ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Циклические процессы. Структура и порядок выполнения оператора For .Next . Структура и порядок выполнения оператора цикла Do...Loop. Оператор цикла For-Each-Next.

К группе операторов организации управления вычислительным процессом принадлежат и операторы цикла.

Основой решения на ЭВМ большинства прикладных задач являются повторения, в частности, многократные повторения расчетов по одним и тем же формулам, но при различных исходных данных. Такие вычислительные процессы называются циклическими, а повторяющиеся участки – циклами.

Главными вопросами, которые необходимо уяснить при изучении программирования циклических ВП, являются:

1. Как организовать повторение циклов при решении той или иной задачи;
2. Сколько раз нужно повторять цикл.

В одних задачах число повторений можно определить заранее до начала решения задачи по исходным данным, в других этого сделать нельзя.

Исходя из этого, программы решения задач называют циклическими программами с заранее известным и заранее неизвестным числом выполнения циклов.

9.1. Структура и порядок выполнения оператора

For.Next

Цикл ***For.Next*** позволяет повторять группу операторов заданное число раз.

Структура оператора:

For ***СчетчикЦикла*** = *Начало* *To* *Конец* *Step* *Шаг*

[оператор 1]

[оператор 2]

.....

[оператор N]

[Exit For]

[оператор N + 1]

[оператор N + 2]

Next ***СчетчикЦикла***

Здесь: ***СчетчикЦикла*** – параметр цикла (переменная, с помощью которой организуется счетчик количества повторений цикла.);

Step (шаг – необязательный аргумент в структуре)– значение, на которое изменяется счетчик при каждом выполнении тела цикла. Если это значение не задано, по умолчанию шаг равен единице.

Начало, Конец – начальное и конечное значения параметра цикла;

Операторы 1...N+2(операторы тела цикла) – любые операторы языка;

Exit For- оператор досрочного выхода из цикла.

Порядок выполнения оператора

1. Присваивание параметру цикла начального значения ***Счетчик-Цикла = Начало***;

2.Проверка условия, например, *СчетчикЦикла* <= *Конец* – перед выполнением цикла;

3.Выполнение операторов тела цикла, если условие *СчетчикЦикла* <= *Конец* выполнено (*True*), и выход из цикла, если условие *СчетчикЦикла* <= *Конец* не выполнено (*False*);

4.Увеличение значения параметра цикла на величину шага *СчетчикЦикла* = *СчетчикЦикла* + *Шаг*;

5.Возврат к началу цикла

В VBA возможна организация вложенных циклов *For..Next* (когда один цикл *For..Next* располагается внутри другого). При этом счетчик каждого цикла должен иметь уникальное имя. Например, для тройного цикла возможна следующая структура программы:

```
For I = A1 To A2 Step H
  For J = 1 To 10
    For K = 1 To 20
      P1: P2: .....PN
    Next K
  Next J
Next I
```

Здесь *I, J, K* – параметры циклов соответственно внешнего, среднего и внутреннего циклов. Операторы тела цикла *P1, P2, ... PN* выполняются $\{[(A2-A1)/H]+1\} * 10 * 20$ раз, т.е. для каждого из значений параметров *I, J, K*.

9.2. Структура и порядок выполнения оператора цикла *Do...Loop*

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

В зависимости от позиции условия выхода из цикла различают два варианта оператора цикла *Do..Loop*: цикл с предусловием и цикл с постусловием.

1) Цикл с предусловием *Do. .While.Loop; Do. . Until. Loop*.

Структура и порядок выполнения:

<i>Do. .While</i> “условие”	<i>Do. . .Until</i> “условие”
<i>P1</i>	<i>P1</i>
<i>P2</i>	<i>P2</i>
...	...
<i>PN</i>	<i>PN</i>
<i>Loop</i>	<i>Loop</i>

Здесь *Do* (выполнять), *While* (пока), *Until* (до тех пор), *Loop* (петля) -служебные слова;

P1...PN – любые операторы языка;

“условие” – булево выражение либо выражение типа сравнение, принимающие значения *True* (*False*).

Порядок выполнения:

While -тело цикла (операторы *P1...PN*) выполняется до тех пор, пока условие имеет значение *True*

Until тело цикла (операторы *P1...PN*) выполняется до тех пор, пока условие имеет значение *False*

2). Цикл с постусловием *Do.Loop.While; Do. Loop. Until*.

Структура и порядок выполнения

<i>Do.</i>	<i>Do.</i>
<i>P1</i>	<i>P1</i>
<i>P2</i>	<i>P2</i>
<i>PN</i>	<i>PN</i>
<i>Loop While</i> “условие”	<i>Loop Until</i> “условие”

Здесь отличие от выше рассмотренной структуры заключается в том, что проверка условия выхода из цикла осуществляется не в начале, а в конце цикла.

В этом случае тело цикла выполняется хотя бы один раз при любом значении условия выхода из цикла.

9.3. Оператор цикла *For-Each-Next*

Для перебора объектов из группы подобных объектов, например, ячеек из диапазона или элементов массива, удобно использовать оператор цикла *For-Each-Next*.

Синтаксис:

For Each Элемент *In* Группа

Блок_операторов

[*Exit For*]

Блок_операторов *Next* Элемент

Вопросы для самоконтроля

1. Что называется циклическим процессом?
2. В каких случаях применяется оператор *For .Next*?
3. Каков синтаксис оператора *For .Next*?
4. Каков порядок выполнения оператора *For .Next*?
5. Что называется вложенными циклами?
6. В каких случаях применяется оператор *Do...Loop*?
7. Каков синтаксис оператора *Do...Loop*?
8. Каков порядок выполнения оператора *Do...Loop*?
9. В каких случаях применяется оператор *For-Each-Next*?
10. Каков синтаксис оператора *For-Each-Next*?

Практическая работа к теме № 9

1. Использование цикла *For ... Next* для заполнения ячеек листа Excel.

Постановка задачи. Использование циклов позволяет заполнять программно ячейки рабочего листа значениями, вычисляемыми по формулам. Например, заполним ячейки 10^{10} значениями валового дохода, характеризующегося функцией: $y=(200c+50)+(5-30c)q + cq^2$, где y – валовый доход, q – выпуск продукции ($10 \leq q \leq 50$). Пусть для определенности $c=0,5$. По данным построим диаграмму.

Выполнение:

1.1. Установите на рабочем листе Excel элемент управления **Кнопку**.

1.2. После двойного щелчка мышью на **Кнопке** войдите в редактор VBA и запишите код:

Option Explicit

Private Sub CommandButton1_Click() 'После нажатия на кнопку произойдет выполнение программы.

Dim c As Single, q As Integer 'Определим переменные: c – параметр, q – количество продукции.

c = 0.5 'Задается значение параметра.

'Цикл для заполнения ячеек активного листа значениями аргумента q .

For q = 10 To 55 Step 5 'Цикл меняется от 10 до 55 с шагом 0,5.

ActiveCell(q / 5).Value = q 'Соответствующие ячейки заполняются значениями q .

Next q 'Конец цикла.

ActiveCell.Offset(0, 1).Activate 'Активизируется смежная ячейка.

ActiveCell.Value = "Y(q)" 'В эту ячейку заносится название будущей диаграммы.

For q = 10 To 55 Step 5 'Цикл для заполнения смежных ячеек.

'В эти ячейки заносятся значения валового дохода, вычисленные по формуле.

*ActiveCell(q / 5).Value = (200 * c + 50) + (5 - 30 * c) * q + c * q ^ 2*

Next q 'Конец цикла.

Selection.CurrentRegion.Select 'Выделяется текущая область со значениями, необходимыми для построения диаграммы.

Charts.Add 'Добавляется объект диаграмма.

ActiveChart.ChartType = xlColumnClustered 'Тип диаграммы – "Гистограмма".

ActiveChart.Location Where:=xlLocationAsObject, Name:="Лист1" 'Диаграмма располагается на активном листе.

End Sub

1.3. Проверьте работу программы.

2. Анимационные эффекты.

Постановка задачи. Использование циклов позволяет "оживить" диаграмму, представить процесс в динамике более наглядно. Пусть в предыдущем задании параметр q с меняется от 0,5 до 0,75. Построить с помощью цикла движущуюся диаграмму.

Выполнение:

Чтобы эффективнее применить анимационные эффекты используем API (Application Programming Interface — прикладной программный интерфейс) – набор функций и процедур, наиболее часто используемых программами. Для этого достаточно вызвать нужную функцию из библиотек kernel32.dll или user32.dll, содержащих функции и процедуры API, а она уже сама все сделает.

2.1. Объявим функцию задержки времени (в миллисекундах) между кадрами *Sleep* из библиотеки *kernel32*. Без такой задержки смена кадров будет происходить слишком быстро.

```
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
```

```
Option Explicit
```

```
Private Sub CommandButton1_Click() 'После нажатия на кнопку произойдет выполнение программы.
```

```
Dim r As Integer 'Переменная цикла заполнения ячеек значениями q.
```

```
For r = 1 To 10 'Переменная r изменяется от 1 до 10 с шагом по умолчанию 1.
```

```
Worksheets("Лист1").Cells(r, 1).Value = 10 + (r - 1) * 5 'Ячейки первого столбца рабочего листа заполняются значениями от 10 (при r=1) до 55 (при r=10).
```

```
Next r 'Следующее значения r.
```

```
Range("B1").Select 'Выделяется ячейка B1, в которую будет вставлена формула расчета валового дохода.
```

```
ActiveCell.FormulaR1C1 = "(200*R1C3+50)+(5-30*R1C3)*RC[-1]+R1C3*RC[-1]^2" 'В активную ячейку вставляется формула. Значения параметра передается по абсолютной ссылке R1C3 ("C1"). Значения q передаются относительными ссылками RC[-1] в той же строке, но на столбец левее.
```

```
Selection.AutoFill Destination:=Range("B1:B10"), Type:=xlFillDefault 'Автозаполнение диапазона "B1:B10".
```

```
Range("B1:B10").Select 'Диапазон "B1:B10" выделяется для построения диаграммы.
```

```
Charts.Add 'Добавляется объект диаграмма.
```

```
With ActiveChart 'Перечисляются необходимые свойства объекта диаграмма.
```

```
.ChartType = xlColumnClustered 'Тип гистограмма.
```

```

.SetSourceData          Source:=Sheets("Лист1").Range("B1:B10"),
PlotBy:=xlColumns'Значения диаграммы выбираются из диапазона
"B1:B10".
.SeriesCollection(1).XValues = "=Лист1!R1C1:R10C1" "Для подписи
оси x выбирается диапазон "A1:A10".
.Location Where:=xlLocationAsObject, Name:="Лист1" 'Диаграмма
располагается на листе1.
End With 'Конец перечисления общих свойств диаграммы.
With ActiveChart.Axes(xlValue) 'Перечисления необходимых свойств
осей диаграммы.
.MinimumScale = 0 'Выбираются минимальное значение для отоб-
ражения диаграммой.
.MaximumScale = 2000 'Выбирается максимальное значение для
отображения диаграммой.
End With 'Конец перечисления свойств оси.
End Sub
'Программа демонстрации анимационных эффектов.
Private Sub CommandButton2_Click() 'После нажатия на кнопку про-
изойдет выполнение программы.
Dim i As Single 'Переменная изменения параметра с.
For i = -1 To 1 Step 0.005 'Пределы изменения и шаг переменной i.
ActiveSheet.Cells(1, 3).Value = i 'Значения переменной i помещаются
в ячейку "C1", содержимое которой используется при расчете валового
дохода и соответственно строится диаграмма.
Sleep (144) ' API функция задержки. В скобках интервал времени
между кадрами – 144 миллисекунд.
DoEvents 'Управление процессом передается компьютеру.
Next i 'Следующее значение i.
End Sub

```

3. Использование цикла для заполнения элемента управления **Список**.

Постановка задачи. Создать форму для исчисления будущей величины депозита (**FV**) по известным величинам первоначальной суммы вклада (**PV**) и срока депозита **n**. Результат выполнения программы отобразить на листе Excel в виде зависимости **FV** от величины ставки по депозиту, изменяющейся от 0 до 1 с шагом 0,01. Данные представятся также в виде графика. В форме эта зависимость отобразится в элементе управления **Список**.

Выполнение:

3.1. В Excel создайте форму, содержащую элементы управления: **Кнопка**, **Список**, два **Поля** и **Надписи** к ним.

3.2. Составьте код:

```
Option Explicit
```

'Расчеты будут производиться после нажатия кнопки

```

Private Sub CommandButton1_Click()
'Объявимвсепеременные.
Dim PV As Currency 'Первоначальная сумма вклада.
Dim срок As Integer 'Срок депозита.
Dim i As Integer 'Переменная для указания строк активного листа
и элементов списка формы.
Dim n As Integer '
Dim A() As Currency 'Массив будущих величин депозита для заполне-
ния ячеек активного листа.
Dim ЭлементыСписка() As Double 'Массив будущих величин депози-
та для заполнения элементов списка формы.
Dim Проценты() As Double 'Массив значений процентов.
Dim Area As Object
PV = CCur(TextBox1.Text) 'Данные (типа Currency), введенные
в поле TextBox1 (исходная сумма) присвоятся переменной PV.
срок = CInt(TextBox2.Text) 'Данные (типа Integer), введенные в поле
TextBox2 (срок депозита) присвоятся срок.
'Работа с элементами рабочего листа.
ActiveSheet.Cells.Clear 'Предварительная очистка ячеек перед авто-
матическим заполнением программой
'Уточнение размеров массивов.
ReDimA(1 To 101)
ReDimПроценты(1 To 101)
'Заполнение ячеек активного листа.
With ActiveSheet
'Цикл для заполнения 100 ячеек активного листа значениями про-
центов и будущих значений депозита.
For i = 1 To 101 Step 1
Проценты(i) = (i - 1) / 100 'Заполнение массива Проценты () значе-
ниями от 0 до 1 с шагом 0,1.
A(i) = Application.FV(Проценты(i), срок, -PV) 'Использование встро-
енной функции FV (будущее значение) для заполнения массива A().
.Cells(i + 1, 1).Value = Проценты(i) 'Строки первого столбца запол-
няются значениями процентов
.Cells(i + 1, 2).Value = A(i) 'Строки второго столбца заполняются
значениями будущих значений депозита.
Next i
End With
'Работа с элементом формы Список.
ReDim ЭлементыСписка(1 To 101, 0 To 1) 'Уточнение размеров дву-
мерного массива элементов списка.
For i = 1 To 101 Step 1 'Цикл для заполнения 100 строк списка значе-
ниями процентов и будущих значений депозита.

```

ЭлементыСписка(i, 0) = Проценты(i) 'Строки первого столбца заполняются значениями процентов.

ЭлементыСписка(i, 1) = A(i) "Строки второго столбца заполняются значениями будущих значений депозита.

Next i

With ListBox1

.Clear 'Предварительная очистка списка.

.ColumnCount = 2 'Объявляется количество столбцов списка (2 столбца).

.List = ЭлементыСписка() 'Заполнение списка значениями элементов двумерного массива с величинами процентов и соответствующих им будущих значений депозита.

.ListIndex = 0 ' Возвращает номер текущего элемента списка. (Нумерация списка начинается с нуля).

End With

'Работа с объектом Chart (диаграмма).

ActiveSheet.ChartObjects.Delete 'Предварительная очистка листа от ранее созданных диаграмм.

Set Area = ActiveSheet.Cells(1, 2).CurrentRegion

n = Area.Columns.Count

ActiveSheet.ChartObjects.Add(195, 30, 200, 190).Select

ActiveChart.ChartWizard Source:=Range(Cells(1, 2), Cells(2, n)), Gallery:=xlColumn, Format:=6, PlotBy:=xlRows, CategoryLabels:=1, SeriesLabels:=0, HasLegend:=False

End Sub

4. Использование цикла для программирования таблицы подстановки расчета будущих значений вкладов.

Постановка задачи. Создать программно таблицу подстановки расчета будущих значений вкладов на диапазоне A1:K11 с ячейками ввода A12 и A13, в которые вводятся расчетные значения срока и ставки, вместо значений которых программа будет вводить реальные значения.

Выполнение:

4.1. Создайте на рабочем листе модуль, в который впишите код.

Option Explicit

Sub ТаблицаПодстановки()

Dim i As Integer 'Определениецикла

With Worksheets("Лист1") 'Начало инструкции With

'Перечислим свойства объекта Рабочий лист

'В ячейку A1 введем рабочую формулу: встроенную функцию для расчета будущего значения вклада

.Range("A1").Formula = "=FV(A12, A13, , -B14)"

'В ячейку B14 будем вводить начальное значение вклада

.Range("A14").Value = "Введите сумму в B14" '

```

For i = 2 To 11 'Цикл примет значения от 2 до 11
'В ячейки на пересечении первого столбца и строк со второй до
одинадцатой поместим значения соответственно от 1 до 10 – ко-
личество лет
.Cells(i, 1) = i - 1
'В ячейки на пересечении первой строки 'и столбцов со второго до
одинадцатого поместим значения процентной ставки от 1 до 10
процентов
.Cells(1, i) = (i - 1) / 100
Next i 'Следующее значение цикла
'Создадим таблицу подстановки на диапазоне A1:K11 с ячейками
ввода A12 и A13, в которые вводятся расчетные значения срока
и ставки, вместо значений которых программа будет вводить ре-
альные значения.
.Range("A1:K11").Table .Range("A12"), .Range("A13")
End With 'Конструкция With
End Sub

```

4.2. Вводя различные значения начального вклада в ячейку A14, проверьте работу сводной таблицы.

5. Пример использования оператора цикла **For-Each-Next** для форматирования ячеек.

Постановка задачи. Создать процедуру, которая в выделенном диапазоне ячеек в Excel положительные числа отображают синим цветом, а отрицательные числа — красным. А также, в зависимости от содержимого ячейки выделенной области изменяет цвет ее фона, изменяет цвет, размер и тип шрифта.

Выполнение:

5.1. Запишите программу:

```
Option Explicit
```

```
Sub Знак()
```

```
Dim c As Object 'Объявляется объект (в данном случае – диапазон
ячеек), который возвращается методом Selection
```

```
For Each c In Selection
```

```
If IsNumeric(c.Value) Then 'Функция IsNumeric(выражение) возвра-
щает True, если выражение может быть описано как числовое, и False —
в противном случае.
```

```
If c.Value > 0 Then c.Value = "+" & c.Interior.ColorIndex =
8'Interior является объектом, характеризующим фон указанного диа-
пазона. 8 – голубой цвет фона. c.Font.Bold = True 'Font — шрифт
диапазона ячеек — является объектом. Этот объект имеет следую-
щие свойства: Bold – Жирный шрифт c.Font.Italic = True Курсивный
Шрифт. c.Font.ColorIndex = 5 'Синий цвет символов.
```

```
c.Font.Size = 20 'Размер шрифта в пт.
```

```

End If If IsNumeric(c.Value) Then If c.Value < 0 Then c.Value = "-
"c.Interior.ColorIndex = 4 'Зеленый цвет фона.c.Font.ColorIndex = 3
'Красный цвет символов.

```

```

End If If IsNumeric(c.Value) Then
If c.Value = 0 Then c.Value = 0
c.Interior.ColorIndex = 6 'Желтый цвет фона
c.Font.ColorIndex = 7 'Фиолетовый цвет символов

```

```

End If
Next c
End Sub

```

5.2. Введите различные числовые значения (положительные, отрицательные, равные нулю) в какой-либо диапазон ячеек. Выделите диапазон и проверьте работу программы.

6. Повторение инструкций **Do. .While.Loop** (условие проверяется до входа в цикл).

Постановка задачи. Создать процедуру, которая заполнит ячейки значениями функции спроса $D(P)=50/(0,5 \times P+5)$ повторением инструкций **Do. .While.Loop** пока условие, что цена $P < 100$ имеет значение **True**.

Выполнение:

6.1. Установите на рабочем листе Excel элемент управления **Кнопку**.

6.2. Запишите код:

```
Option Explicit
```

'После нажатия Кнопки ячейки заполняются значениями функции спроса $D(P)$.

```
Private Sub CommandButton1_Click()
```

```
'Объявим переменную-счетчик, показывающую число итераций.
```

```
Dim i As Integer
```

```
'Объявим переменную p, показывающую значение цены от 0 до 100.
```

```
Dim P As Integer
```

```
'Объявим переменную D, показывающую значение функции спроса.
```

```
Dim D As Double
```

```
'Объявим объектную переменную Ячейка как диапазон.
```

```
Dim Ячейка As Range
```

```
'Установим, что Ячейка – это ячейка, где установлен курсор.
```

```
Set Ячейка = ActiveCell
```

```
'Установим начальные значения переменных.
```

```
i = 0
```

```
P = 0
```

'Сразу же (до входа в цикл) ограничим процесс, так чтобы продолжался пока значение P больше 100.

```
Do While P < 100
```

```
'Следующее значение цены уменьшается на единицу
```

```
P = P + 1
```

'С точностью до двух десятичных знаков вычисляется функция спроса по заданной формуле.

*$D = \text{Format}(50 / (0.5 * P + 5), "###0.00")$*

'При этом значение счетчика итераций увеличивается на единицу.

$i = i + 1$

'Следующая ячейка заполнится следующим значением переменной D.

Ячейка(i).Value = D

' Повторение процесса.

Loop

'В окне сообщений выведется число итераций

MsgBox "Выполнено " & i & " итераций цикла."

End Sub

6.3. Проверьте выполнение программы.

7. Цикл с предусловием **Do . Loop . While**.

Постановка задачи. Создать процедуру, которая заполнит ячейки значениями функции спроса $D(P)=50/(0,5 \times P+5)$ повторением инструкций **Do . Loop . While**, когда условие, что $D > 4$ имеет значение **True**.

Выполнение:

7.1. В предыдущей программе замените:

Do While P > 10 на Do;

Loop на Loop While D > 4

8. Проверка условия в инструкции **Do..Loop** с помощью ключевого слова **Until**: до входа в цикл.

Постановка задачи. Создать процедуру, которая заполнит ячейки значениями функции предложений $S(P)=0,05(P^2-5)$ повторением инструкций **Do . Loop** с помощью ключевого слова **Until**: до входа в цикл.

Выполнение:

8.1. В предыдущей программе:

предварительно укажите начальное значение $S=0$;

замените переменную D на S;

замените формулу $50 / (0.5 * P + 5)$ на $0.05 * (P ^ 2 - 5)$;

замените **Do** на **Do Until S = 10**.

8.2. Проверьте выполнение программы.

9. Проверка условия в инструкции **Do . Loop . Until**

Постановка задачи. Создать процедуру, которая заполнит ячейки значениями функции предложений $S(P)=0,05(P^2-5)$ повторением инструкций **Do . Loop** с помощью ключевого слова **Until**: после в цикл.

Выполнение:

9.1. В предыдущей программе замените

Do Until S < 10 на Do

Loop на Loop Until S < 10

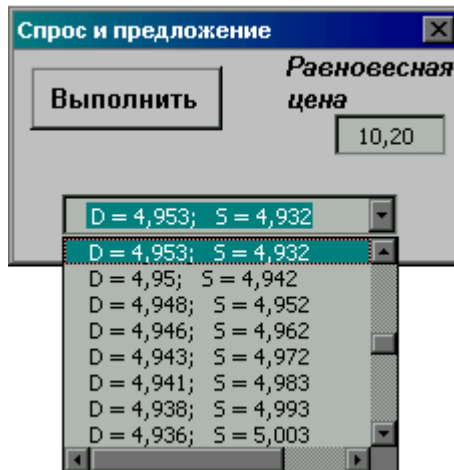
9.2. Проверьте выполнение программы.

10. Итерационные вычисления.

Постановка задачи. Для документа Word создать форму со списком. Заполнить список значениями функций спроса и предложений при цене, изменяющейся от 0 до 100 с шагом 1. Вычислить цену равновесия с точностью до 0,1. Вывести результат о отдельном поле формы.

Выполнение:

10.1. Создадим форму с элементами управления: *Кнопку*, *Поле* и *Поле со списком*.



10.2. Запишем код:

```
Option Explicit
Private Sub CommandButton1_Click()
Dim P As Double, D As Double, S As Double
With ComboBox1
.ColumnCount = 2 'Установим два столбца в списке
.ColumnWidths = "100;50" 'Установим ширину столбца.
End With
P = 0
Do
S = Format(0.05 * (P ^ 2 - 5), "###0.000")
P = P + 0.01
D = Format(50 / (0.5 * P + 5), "###0.000")
ComboBox1.AddItem " D = " & D & "; " & " S = " & S 'Заполняем список
форматированными данными
If S - D < 0.001 Then
TextBox1.Value = Format(P, "###0.00")
End If
Loop Until P > 20
End Sub
```

10.3. Измените программу так, чтобы найденные значения отразились в ячейках Excel.

ТЕМА 10. ОТЛАДКА ПРОГРАММ,

ПРОЦЕДУР И ФУНКЦИЙ

Интеллектуальные возможности редактора VBA. Отладка программы. Панели системы контроля. Установка контрольных точек Прерывание по ошибке

Успешность бизнеса во многом зависит от безошибочности действий. Использование компьютера при принятии решений стала повседневной практикой во многих фирмах. Однако при работе с такими сложными системами возможны ошибки, как связанные с "компьютерной неграмотностью" – неправильным использованием встроенных функций и процедур, так и ошибками программирования при написании программ пользователь независимо от его опыта допускает те или иные ошибки. Для создания надежных программ в *редакторе Visual Basic* предусмотрен ряд мер, к которым относятся средства отладки программ:

10.1. Средства отладки программ

К средствам отладки программ прежде всего относится панель инструментов *Отладка*.

Вызвать панель инструментов *Отладка* можно командой меню *Вид – Панели инструментов – Отладка*:

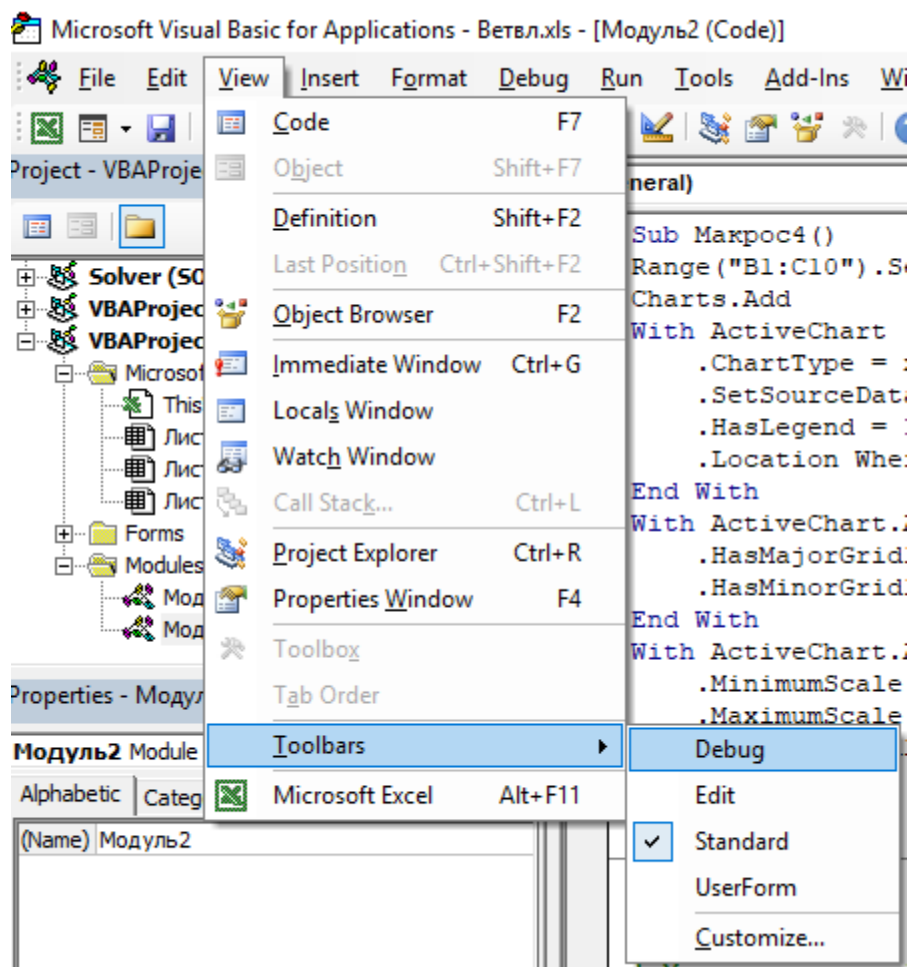


Рис. 14. Окно проекта.

В результате появится панель инструментов **Отладка** (Рис. 16).



Рис. 15. Окно проекта.

Инструменты отладки программы:

1. **Конструктор** включает и выключает состояние проектирования.
2. **Запуск подпрограммы/UseRTIorm** запускает программу на выполнение.
3. **Прервать** прерывает выполнение программы и переводит программу в режим прерывания.
4. **Сброс** приостанавливает выполнение и производит сброс проекта.
5. **Точка останова** позволяет работать с т.н. **точками прерываниями**, с помощью которых размечают промежуточные финиши выполнения программы.
6. **Шаг с заходом** задает пооператорный режим выполнения программы.

7. **Шаг с обходом** осуществляет вызов процедур и функций за один шаг.
8. **Шаг с выходом** прерывает пошаговое выполнение процедуры и позволяет вернуться к этому режиму в вызывающей процедуре.
9. **Окно локальных переменных** позволяет наблюдать значения локальных переменных программы.
10. **Окно отладки** отображает информацию, получаемую от отладочных инструкций программы или от команд, вводимых непосредственно в окне.
11. **Окно контрольного значения** позволяет следить за состоянием некоторых переменных и выражений программы.
12. **Контрольное значение** предназначено для быстрого просмотра контрольного выражения.
13. **Стек вызова** позволяет отладить рекурсивные процедуры или процедуры с вложенностью вызовов.

Правильное использование интеллектуальных возможностей **редактора VBA** позволяет исправить ошибки программ.

10.2. Интеллектуальные возможности редактора VBA

Чтобы запустить программу из редактора Visual Basic, следует нажать клавишу **F5**. Если при ее выполнении обнаружатся ошибки в записи или обращении к неизвестным командам, то появится сообщение о том, что команда редактору неизвестна и надо изменить (исправить) текст программы.

При составлении программ возможны ошибки следующих типов:

ошибки компиляции, которые проявляются в процессе составления программ и являются следствием синтаксических ошибок, неправильного использования свойств и т.п.;

ошибки выполнения, которые проявляются после запуска программы, и является результатом попытки выполнить недопустимую операцию типа ввода несоответствующих данных, некорректности вычислений и т.п.;

логические ошибки, в результате которых программа выдает неверные результаты.

При обнаружении ошибки компиляции VBA выделяет строку, в которой содержится ошибка, красным цветом, и на экране отображается диалоговое окно с сообщением о возможной причине, вызвавшей ошибку.

При обнаружении ошибки выполнения на экране отображается диалоговое окно с сообщением о номере ошибки и возможной причине ее появления, а строка с ошибкой помечается стрелкой желтого цвета.

Обнаружение и устранение логических ошибок связано с тщательным анализом алгоритма программы с привлечением средств отладки VBA.

10.3. Отладка программы

В VBA для поиска алгоритмических ошибок заложены большие возможности для отладки программ. К ним относятся:

1. Использование инструкции *Option Explicit*- для явного описания переменных.
2. Пошаговое выполнение программ. (Команды меню *Отладка*).
3. Использование точек останова (команда *Отладка, Точка останова*).
4. Вывод значений свойств и переменных. (Для этого достаточно расположить указатель мыши на имени свойства или переменной).
5. Другим способом отслеживания текущих значений данных является использование диалогового окна *Контрольные значения*, отображаемого на экране с помощью команды *Вид, Окно контрольного значения*. Это окно применяется для одновременного отображения нескольких переменных.
6. Команда *Отладка, Добавить контрольные значения* позволяет добавить новые контрольные значения в диалоговое окно.
7. Окно *Локальные переменные*, отображаемое на экране командой *Вид, Окно локальных переменных*, выводит значения всех переменных текущей процедуры, а не только специально выбранных, как это происходит в окне *Контрольные значения*.
8. Окно *Проверка*, отображаемое на экране командой *Вид, Окно отладки*, предоставляет пользователю возможность:

набирать и вычислять отдельные операторы (оператор должен быть набран в одной строке);

определять текущие значения переменных и свойств. Для этого в окне *Проверка* надо набрать вопросительный знак, имя переменной или свойства и нажать клавишу *Ввод*.

Устанавливать текущие значения переменных. Для этого в окне *Проверка* надо набрать имя переменной, знак присвоения и новое значение переменной.

Таким образом, если после неправильного использования команд возникает ошибка, то Visual Basic в диалоговом окне кратко опишет ее.

10.4. Панели системы контроля. Установка контрольных точек Прерывание по ошибке


В режиме отладки доступно следующее.

1. Пошаговое исполнение программы.
Все команды выполняются последовательно, но после каждой из них нужно нажать кнопку *F8* (следующая команда будет подсвечиваться желтым цветом). Здесь можно увидеть, где возникает ошибка.
2. Просмотр значений переменных.


Подведя курсор в режиме отладки к имени любой переменной, можно увидеть ее значение. А значения их всех имеются в окне **Локальные переменные**.

3. Точки останова. Если программа большого объема, то перемещаться по всем ее строчкам с помощью кнопки **F8** довольно обременительно. Лучше использовать точки останова, т.е. те отметки в тексте программы, где ее исполнение остановится, а она сама перейдет в режим отладки. Чтобы установить такую точку, следует щелкнуть кнопкой мыши в левом сером поле окна модуля – тогда появится коричневая строка с кружком на нужном месте.

Начиная с этой точки, каждый раз при нажатии кнопки **F8** программа будет выполняться в пошаговом режиме. Но можно отказаться от такой работы, нажав кнопку **F5**. Точки останова полезны тогда, когда требуется локализовать ошибку в программе, особенно такую, которая находится в середине текста или не единична.

4. 4.Окно контрольного значения .

В этом окне можно задать определенное выражение, значение которого будет подсчитываться при выполнении программы. Кроме того, выделив в режиме отладки какое-либо выражение, нажав комбинацию клавиш **Shift +F9** можно увидеть его значение, а также добавить это выражение в окно контрольного значения и отследить, как оно будет изменяться. Это необходимо, например, при отладке программы с множеством арифметических выражений, одно из которых выдает ошибку. С помощью этого окна можно отследить, при изменении какого параметра значение выражения станет неприемлемым.

5. Окно проверки или отладки .

В данное окно можно попасть, выбрав пункты **Вид – Окно отладки**. Если в текст программы вставить специальную команду – **Debug.Print**, то при выполнении последней в окне отладки будет находиться то, что задано в параметрах команды.

Возможности этого окна многообразны. Так, в него можно вводить сообщения при выполнении какого-либо условия или отображать промежуточные результаты вычислений для контроля их правильности, а также использовать его как маленький калькулятор или командную строку.

6. Стек вызова .

Отображается структура программы (она состоит из нескольких подпрограмм – процедур или функций) и показываются переходы между ее подпрограммами. Команда доступна только в режиме прерывания.

Окна контрольного значения, локальных переменных, проверки или отладки, стека вызова могут быть получены как из меню **Вид** (или соответ-

ствующими сочетаниями клавиш), так и нажатием кнопок на панели **Отладка**.

В любой момент можно прервать работу макрокоманды, нажав клавиши **Ctrl+Break**. Затем можно перейти в режим отладки или окончания работы. Если требуется выйти из режима отладки, следует выбрать из меню **Запуск** функцию **Сброс** или нажать на панели **Отладка** кнопку **Сброс**.

Чтобы выполнение программы началось с какой-либо команды, надо в режиме отладки просто перетащить мышью по серой полосе слева на требуемое место указатель в виде желтой стрелки. Если при просмотре текста вы забыли, где программа была остановлена, то выберите из меню **Отладка** команду **Показать следующую инструкцию**. Помните, всегда можно получить подробную справку по любой команде, поставив на нее курсор и нажав клавишу **F1**.

Под исключительными ситуациями в дальнейшем будем понимать ситуации, при возникновении которых в вычислениях возникают непонятные на первый взгляд ситуации и процесс решения прерывается.

При разработке программ важно предусмотреть анализ возможных ошибок, возникающих по вине пользователя.

При этом возможны два подхода:

предотвращение ошибок: программно анализировать вводимые или вычисляемые данные и в случае, если они могут приводить к ошибке, обеспечить, чтобы программа информировала пользователя о необходимости корректного задания данных;

обработка ошибок: в случае появления ошибки, перехватить ее, обработать и программно откликнуться на возникшую ошибку.

При создании приложений надо сочетать оба подхода, применяя в каждом конкретном случае и для каждой возможной ошибки тот, который кажется разработчику наиболее эффективным.

Вопросы для самоконтроля

1. Каких типов возможны ошибки при составлении программ?
2. В чем проявляются ошибки компиляции?
3. В чем проявляются ошибки выполнения?
4. В чем проявляются логические ошибки?
5. Какие возможности предусмотрены в ВВА для поиска алгоритмических ошибок?
6. Каковы подходы для предотвращения ошибок, возникающих по вине пользователя?

Практическая работа к теме № 10

Предотвращение ошибок.

Постановка задачи. В данном разделе рассмотрим процесс создания приложения, в котором предотвращается появление ошибок на примере формы для конвертации валют, использующей нахождение частного от деления двух чисел.

Выполнение:

1. Создайте диалоговое окно, содержащее:
 - поле для ввода исходной суммы;
 - поле со списком, из которого можно выбрать тип валюты для этого поля;
 - поле для вывода результата;
 - поле со списком, из которого можно выбрать тип валюты, в которую конвертируется исходная сумма;
 - поля для ввода банковского курса валют;
 - кнопка, после нажатия на которую произойдут вычисления;
 - соответствующие надписи к элементам управления.

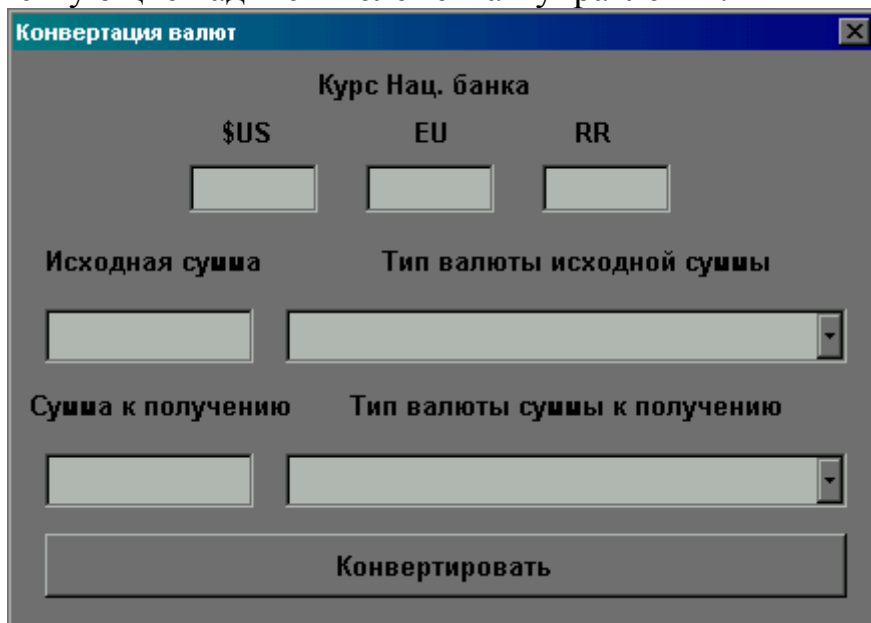


Рис. 16. Форма конвертации валют.

2. Чтобы предотвратить ошибки, связанные с неверным употреблением типов данных, воспользуйтесь инструкцией *Option Explicit* и функциями, преобразующими выражения к соответствующему типу данных (*Cdbl*(выражение), *CStr*(выражение) и т.п.).

3. Контроль ввода именно числовых данных в поля обеспечьте функцией *IsNumeric*(выражение):

```
If IsNumeric ( TextBox1. Text) = False Then  
Exit Sub
```


TextBox1.SetFocus 'Курсор устанавливается в нужном поле для ввода или изменения данных.

End If

4. Поскольку в программе используется операция деления, то необходимо обеспечить проверку, не является ли знаменатель нулем. Например, так:

If Cdbl(TextBox1.Text) = 0 Then

TextBox1.SetFocus 'Курсор устанавливается в нужном поле для ввода или изменения данных.

Exit Sub

5. Обеспечьте появление инструкций в случае неправильного пользования формой, с помощью окон информационных сообщений. Например: *MsgBox "Введите числовое значение больше нуля" & Chr(10) & "(десятичный разделитель – ЗАПЯТАЯ)", _*

vbInformation, "Неправильно заполнены поля!"

6. Проанализируйте пример программы, которая обеспечивает некоторую защиту от возможных ошибок при работе с данной формой:

Option Explicit

'Объявим переменные уровня модуля.

Dim i As Integer 'Целочисленная переменная для заполнения списка.

Dim КурсИсходнойВалюты As Double 'КурсИсходнойВалюты

Dim КурсПокупаемойВалюты As Double 'КурсПокупаемойВалюты

Dim Валюта(0 To 3) As String 'Массив типов валют.

Dim Курс(0 To 3) As Double 'Массив для банковских курсов валют.

Private Sub UseRT1orm_Initialize() 'Устанавливаются параметры формы на этапе инициализации.

TextBox2.Locked = False 'Запретим вносить изменения в поле результата, но оставим возможность копировать результат.

'Заполнение массива наименований валют.

Валюта(0) = "Гривня"

Валюта(1) = "Американский доллар"

Валюта(2) = "Единая европейская валюта"

Валюта(3) = "Российский рубль"

'Установим поле с одно колончатыми списками.

ComboBox1.BoundColumn = 0

ComboBox2.BoundColumn = 0

'Использование цикла для заполнения полей со списками наименованиями валют.

For i = 0 To 3

ComboBox1.AddItem Валюта(i) 'Заполнение первого поля со списком.

ComboBox2.AddItem Валюта(i) 'Заполнение второго поля со списком.

Next i

End Sub

```

Private Sub ComboBox1_Click() 'Щелчок мыши на выбранном элементе
первого списка приводит к следующим действиям.
КурсИсходнойВалюты = CDbI(Курс(ComboBox1.ListIndex)) 'Переменная
КурсИсходнойВалюты получает значение курса по порядковому номеру
элемента, выбранного в первом списке.
CommandButton1.Enabled = True 'Разблокируемкнопку.
ComboBox2.SetFocus 'Курсор переносится во второй список для выбора
типа получаемой валюты.
End Sub
Private Sub ComboBox2_Click () 'Щелчок мыши на выбранном элементе
второго списка приводит к следующим действиям.
КурсПокупаемойВалюты = CDbI(Курс(ComboBox2.ListIndex)) 'Переменная
КурсПокупаемойВалюты получает значение курса по порядковому номеру
элемента, выбранного во втором списке.
CommandButton1.Enabled = True 'Расблокируемкнопку.
End Sub
Private Sub CommandButton1_Click() 'При нажатии кнопки произойдет пе-
ресчет суммы из одной валюты в другую.
'Предварительная проверка правильности заполнения полей курсов валют.
If IsNumeric(TextBox3.Text) = False Then 'Если в поле введено неверное зна-
чение, то появится предупреждающее окно сообщений.
Call ИнструкцияЗаполненияПоля 'Вызов подпрограммы с инструкцией за-
полнения полей ввода данных.
TextBox3.SetFocus 'Курсор устанавливается в нужном поле для ввода или
изменения данных.
Exit Sub 'Выполнение программы приостанавливается.
End If
If IsNumeric(TextBox4.Text) = False Then 'Если в поле введено неверное чис-
ловое значение, то появится предупреждающее окно сообщений.
Call ИнструкцияЗаполненияПоля 'Вызов подпрограммы с инструкцией за-
полнения полей ввода данных.
TextBox4.SetFocus 'Курсор устанавливается в нужном поле для ввода или
изменения данных.
Exit Sub 'Выполнение программы приостанавливается.
End If
If IsNumeric(TextBox5.Text) = False Then 'Если в поле введено неверное чис-
ловое значение, то появится предупреждающее окно сообщений.
Call ИнструкцияЗаполненияПоля 'Вызов подпрограммы с инструкцией за-
полнения полей ввода данных.
TextBox5.SetFocus 'Курсор устанавливается в нужном поле для ввода или
изменения данных.
Exit Sub 'Выполнение программы приостанавливается.
End If

```

'Проверка наличия нулевого значения.
If CDb1(TextBox3.Text) = 0 Then 'Если в поле введено неверное числовое значение, то появится предупреждающее окно сообщений.
Call ИнструкцияЗаполненияПоля 'Вызов подпрограммы с инструкцией заполнения полей ввода данных.
TextBox3.SetFocus 'Курсор устанавливается в нужном поле для ввода или изменения данных.
Exit Sub 'Выполнение программы приостанавливается.
End If

If CDb1(TextBox4.Text) = 0 Then 'Если в поле введено неверное числовое значение, то появится предупреждающее окно сообщений.
Call ИнструкцияЗаполненияПоля 'Вызов подпрограммы с инструкцией заполнения полей ввода данных.
TextBox4.SetFocus 'Курсор устанавливается в нужном поле для ввода или изменения данных.
Exit Sub 'Выполнение программы приостанавливается.
End If

If CDb1(TextBox5.Text) = 0 Then 'Если в поле введено нечисловое значение, то появится предупреждающее окно сообщений.
Call ИнструкцияЗаполненияПоля 'Вызов подпрограммы с инструкцией заполнения полей ввода данных.
TextBox5.SetFocus 'Курсор устанавливается в нужном поле для ввода или изменения данных.
Exit Sub 'Выполнение программы приостанавливается.
End If

'Заполнение массива банковского курса валют.
Курс(0) = 1 'Курс гривни.
Курс(1) = TextBox3.Value 'Ввод курса доллара.
Курс(2) = TextBox4.Value 'Ввод курса евро.
Курс(3) = TextBox5.Value 'Ввод курса российского рубля.
'Проверка правильности ввода числа в первое поле.
If IsNumeric(TextBox1.Text) = False Then 'Если в поле введено нечисловое значение, то появится предупреждающее окно сообщений.
Call ИнструкцияЗаполненияПоля
TextBox1.SetFocus 'Курсор устанавливается в нужном поле для ввода или изменения данных.
Exit Sub 'Выполнение программы приостанавливается.
End If

If CDb1(TextBox1.Text) = 0 Then 'Если в поле введено нулевое значение, то появится предупреждающее окно сообщений.
Call ИнструкцияЗаполненияПоля 'Вызов подпрограммы с инструкцией заполнения полей ввода данных.

TextBox1.SetFocus 'Курсор устанавливается в нужном поле для ввода или изменения данных.

Exit Sub 'Выполнение программы приостанавливается.

End If

'Проверка заполнения полей со списками.

If ComboBox1.Text = "" Then 'Если в поле введено нечисловое значение, то появится предупреждающее окно сообщений.

CommandButton1.Enabled = False 'Кнопка блокируется.

ComboBox1.SetFocus 'Курсор устанавливается в первое поле со списком для выбора типа валюты.

Exit Sub 'Выполнение программы приостанавливается.

End If

If ComboBox2.Text = "" Then 'Если в поле введено нечисловое значение, то появится предупреждающее окно сообщений.

CommandButton1.Enabled = False 'Кнопка блокируется.

ComboBox2.SetFocus 'Курсор устанавливается в первое поле со списком для выбора типа валюты.

Exit Sub 'Выполнение программы приостанавливается.

End If

'Расчет.

*TextBox2.Value = CStr(Format(CDbl(TextBox1.Value) * КурсИсходнойВалюты / КурсПокупаемойВалюты, "Fixed"))*

End Sub

Sub ИнструкцияЗаполненияПоля() ' Подпрограмма с инструкцией заполнения полей ввода данных.

*MsgBox "Введите числовое значение больше нуля" & Chr(10) & "(десятичный разделитель – ЗАПЯТАЯ}", _
vbInformation, "Неправильно заполнены поля!"*

End Sub

7. Проверьте работоспособность программы, вводя заведомо ошибочные данные.

8. Для предотвращения других ошибок воспользуйтесь обработчиком ошибок типа:

On Error GoTo Инструкция

Exit Sub

Инструкция:

MsgBox "Введите положительное число.", vbInformation, "Неправильно заполнены поля!" 'Реакция на ошибку выполнения.

ТЕМА 11. СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ОБЪЕКТОВ

Созданиемодулейкласса. Процедуры Property Let, Property и Get Property Set.

В VBA предусмотрена возможность создания пользовательских объектов, использование которых позволяет сократить текст программы и сделать его более понятным. Пользовательские объекты являются элементами пользовательских классов. Пользовательские классы конструируются в модулях классов, которые создаются в редакторе Visual Basic выбором команды **Вставка – Модуль класса** (*Insert – Class Module*). При создании классов надо предусмотреть его инициализацию, описание свойств и методов, которыми будет наделен объект.

11.1. Создание модулей класса

Чтобы создать класс:

1. Выберите команду **Вставка – Модуль класса**. Откроется окно нового модуля класса.
2. Нажмите клавишу **F4** и присвойте в появившемся окне свойству **Name** имя класса. Имя модуля класса является именем класса объектов.
3. В разделе описания модуля объявите переменные уровня модуля, которые используются как «значения свойств».
4. Инициализируйте класс при помощи процедуры **Private Sub Class_Initialize**. В этой процедуре надо указать значения, принимаемые по умолчанию переменными уровня модуля, описывающими «значения свойств».
5. При помощи процедур **Property Let** объявите имена свойств, значениями которых являются числовые данные, а при помощи процедур **Property Set** объявите имена свойств, значениями которых являются объекты. Если какое-то свойство – только для чтения, то для него не надо составлять процедуру **Property Let**.
6. При помощи процедур **Property Get** установите возможность считывания значения свойств.
7. Создайте методы класса.

Методы создаются при помощи обычных процедур и функций. Если метод возвращает число, то для его конструирования используется функция, а в остальных случаях – процедура. Для удаления объекта из памяти по завершению работы с ним допустимо создание процедуры **Private Sub Class_Terminate**.

11.2. Процедуры *Property Let*, *Property u* *Get Property Set*

Процедура *Property Let* служит для объявления имен свойств, значениями которых являются числовые данные.

Процедура *Property Set* служит для объявления имен свойств, значениями которых являются объекты.

Процедура *Property Get* обеспечивает возможность считывания значения свойств.

Процедуры *Property Let*, *PropertyGet* и *Property Set* имеют такую же структуру, что и обычные процедуры. Они предназначены для специфических задач, описанных выше.

Вопросы для самоконтроля

1. Для каких целей предусмотрена возможность создания пользовательских объектов?
2. С помощью каких команд конструируются пользовательские классы?
3. Что необходимо предусмотреть при создании классов?
4. Каков порядок создания классов?
5. Для чего предназначена процедура *Property Let*?
6. Для чего предназначена процедура *Property Get*?
7. Для чего предназначена процедура *Property Set*?

Практическая работа к теме № 11

В экономике многие величины имеют однотипный вид. Например, производительность, рентабельность, Отношение и т.п. имеют вид отношения (дроби). Для работы с подобными величинами удобно создать пользовательский класс, снабдив его соответствующими свойствами и методами, что в дальнейшем значительно упростит процесс программирования.

1. Создание класса, объединяющего объекты, имеющих вид дроби.

Постановка задачи. Создать класс Отношение, определяющий тип данных, которые задаются парой положительных чисел (числителем и знаменателем, причем последний не может быть равным нулю) и операции над ними (сложение, вычитание, умножение и т.п.).

Выполнение:

- 1.1. Запустите Word или Excel, откройте документ со своим приложением и запустите редактор Visual Basic .
- 1.2. В окне проекта кликните правой кнопкой мыши и во всплывающем меню выберите пункт "**Вставить**" и далее "**Модуль класса**".

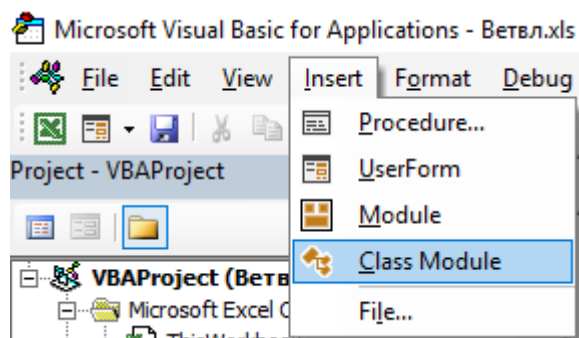


Рис. 17. Вставка модуля класса.

1.3. В дереве объектов выделите свой класс (по умолчанию *Класс1*) и в окне свойств значение свойства *Name* установите для него равным, например, *Отношение*. Это имя можно будет использовать в дальнейшем в программах для создания экземпляров класса — переменных типа *Отношение*. **Создание класса.**

1.4. Откройте двойным щелчком мыши на имени модуля класса в окне проекта окно для ввода исходного текста программы. Переменные, описанные на уровне модуля, будут являться свойствами класса. Процедуры и функции, описанные в модуле класса, будут являться методами этого класса.

1.5. Запишите код:

Option Explicit

Private u As Double, k As Double 'Переменные (свойства) для внутреннего использования в классе Отношение.

'Описание свойств при чтении значений которых должен вызываться метод вычисления.

Public Property Get Числитель() As Double

Числитель = u 'Значение свойства Числитель.

End Property

Public Property Get Знаменатель() As Double '

Знаменатель = k 'Значение свойства Знаменатель.

End Property 'Завершения конструкции.

Public Property Let Числитель(ByVal НовыйЧислитель As Double)

'Property Let применяется для передачи значений (ByVal) стандартных и определенных программистом типов.

If IsNumeric(НовыйЧислитель) = False Then 'Присвоение переменной НовыйЧислитель нечислового значения вызовет ошибку.

Call Инструкция 'Реакция на ошибку: вызов подпрограммы Инструкция.

Exit Property 'Приостановка выполнения программы в случае неправильного ввода данных.

ElseIf НовыйЧислитель < 0 Then 'Присвоение переменной НовыйЧислитель неположительного значения вызовет ошибку.

Call Инструкция 'Реакция на ошибку: вызов подпрограммы Инструкция.

Exit Property ' Приостановка выполнения программы в случае неправильного ввода данных.

End If 'Конецблока If.

и = НовыйЧислитель 'Новое значение внутренней переменной.

End Property 'Завершения конструкции.

Public Property Let Знаменатель (ByVal НовыйЗнаменатель As Double)

If IsNumeric(НовыйЗнаменатель) = False Then 'Присвоение переменной НовыйЗнаменатель нечислового значения вызовет ошибку.

Call Инструкция 'Реакция на ошибку: вызов подпрограммы Инструкция.

Exit Property ' Приостановка выполнения программы в случае неправильного ввода данных.

ElseIf НовыйЗнаменатель <= 0 Then 'Присвоение переменной НовыйЧислитель неположительного значения вызовет ошибку.

Call Инструкция 'Реакция на ошибку: вызов подпрограммы Инструкция.

Exit Property ' Приостановка выполнения программы в случае неправильного ввода данных.

End If 'Конецблока If.

k = НовыйЗнаменатель 'Новое значение внутренней переменной.

End Property 'Завершения конструкции.

'Определение действий над объектами типа Отношение.

Public Function Сложение (ByVal Слагаемое As Отношение) As Отношение 'Определение действия сложения двух объектов типа Отношение.

Dim Сумма As New Отношение 'Определение результата.

With Слагаемое 'Действие функции.

'Назначение свойств результату действия функции.

*Сумма.Числитель = и * .Знаменатель + k * .Числитель 'Реализация правила сложения дробей.*

*Сумма.Знаменатель = k * .Знаменатель*

End With

Set Сложение = Сумма 'Результат сложения – сумма.

End Function 'Завершения конструкции.

Public Function Умножение (ByVal Сомножитель As Отношение) As Отношение

Dim Произведение As New Отношение 'Определение результата.

With Сомножитель 'Действие функции.

'Назначение свойств результату действия функции.

*Произведение.Числитель = u * .Числитель 'Числитель одной дроби умножается на числитель другой, а результат передается соответствующей переменной.*

*Произведение.Знаменатель = k * .Знаменатель 'Знаменатель одной дроби умножается на Знаменатель другой, а результат передается соответствующей переменной.*

End With

Set Умножение = Произведение 'Результат умножения – произведение.

End Function 'Завершения конструкции.

'Определение функции, выводящей объект Отношение в виде десятичной дроби.

Public Function Величина(ByVal ОбъектОтношение As Отношение) As Отношение

Dim ДесятичнаяДробь As New Отношение 'Определение результата.

With ОбъектОтношение 'Действие функции.

'Назначение свойств результату действия функции.

ДесятичнаяДробь.Числитель = .Числитель / .Знаменатель 'Результат деления присваивается числителю.

ДесятичнаяДробь.Знаменатель = 1

End With

Set Величина = ДесятичнаяДробь

End Function 'Завершения конструкции.

Public Sub Инструкция() 'Появление информационного окна с инструкцией в случае ошибочного ввода данных.

MsgBox "Введите положительное число.", vbInformation, "Неправильно заполнены поля!" 'Реакция на ошибку выполнения.

End Sub

'Инициализация по умолчанию Отношения дробью 1/1.

Private Sub Class_Initialize()

u = 1

k = 1

End Sub

Таким образом, последовательно определяя необходимые свойства и методы объектов класса с помощью специальных процедур **Property Let**, **Property Get** и **Property Set** можно разработать необходимые Вам объекты.

2. Расчет маргинальной рентабельности.

Постановка задачи. Использование собственных пользовательских классов позволяет существенно упростить создание форм. В частности, снабдив класс свойствами защиты полей от некоторых ошибок ввода, в последствии уже можно не возвращаться к данному вопросу при создании различных форм, использующих данный пользовательский класс.

Создайте форму для расчета маргинальной рентабельности, используя созданный в предыдущем задании класс Отношение.

Выполнение:

2.1. . В Word создайте форму, вызываемую с панели инструментов и содержащую элементы управления:

поля ввода для ввода и вывода данных;

кнопку для запуска программы.

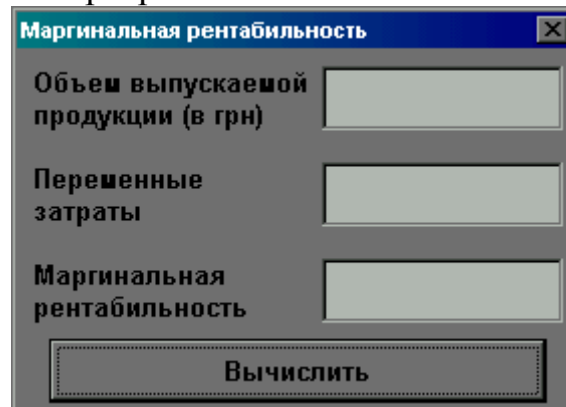


Рис. 18. Форма для расчета маргинальной рентабельности.

2.2. Маргинальная рентабельность MR находится по формуле: $MR = (P/V) - 1$, где P – объем выпускаемой продукции в ценовом выражении, V – переменные затраты. Программа расчета маргинальной рентабельности с учетом некоторой защиты полей от неправильного ввода имеет вид:

Option Explicit

Private Sub CommandButton1_Click()

On Error GoTo Инструкция 'В случае ввода нечисловых данных в поля производится переход по метке Инструкция.:

Dim MR As New Отношение 'Объявляется только один объект типа Отношение.

'В соответствующие поля вводятся значения свойств объекта Отношение.

MR.Числитель = TextBox1.Text

MR.Знаменатель = TextBox2.Text

Set MR = MR.Величина(MR) 'К объекту применяется определенный пользователем метод

TextBox3.Text = MR.Числитель - 1 'Вывод результата.

Exit Sub 'приостановка программы.

Инструкция: 'Метка, после которой определяются действия для предотвращения ошибки.

MsgBox "Введите положительное число." & Chr(10) & "(десятичный разделитель – ЗАПЯТАЯ)", vbInformation, "Неправильно заполнены поля!"

'Реакция на ошибку выполнения.

End Sub

2.3. Проверьте работу формы, вводя неверные данные. Обратите внимание, что реакция на ввод отрицательных значений входит в свойства пользовательского объекта.

ЗАКЛЮЧЕНИЕ

От современного пользователя, использующего ПК для решения проблем в области бизнеса, требуется постоянное совершенствование уже полученных навыков, освоение новых возможностей и технологий. Профессиональные программисты часто не могут уловить все нюансы проблем, требующих специальной экономической подготовки, опыта работы в бизнесе. Даже чтобы грамотно сформулировать задание для программиста, пользователь должен иметь некоторые представления о процессе программирования.

На примере приложений Word и Excel (как самых распространенных) рассмотрены некоторые пути применения элементов офисного программирования для создания системы электронных документов, настройки их с помощью VBA на специфику реально решаемых бизнес-задач. Полученные навыки позволят в дальнейшем применить VB-технологии при создании документов, использующих и другие приложения (Access, Power Point и др.), освоить технику создания интерактивных Web-документов, создавать уникальные приложения в других средах визуального программирования, глубже понять основные идеи и принципы построения и функционирования, как отдельных приложений, так и ОС в целом.

СПИСОК ЛІТЕРАТУРИ

1. Поденежко О. В. Офісне програмування VBA : навч. посіб. / О. В. Поденежко, Н. К. Сьомка ; Держ. податк. адмін. України, Нац. ун-т ДПС України. – Ірпінь, 2010. – 189 с.
2. Малачівський П. С. Програмування на VISUAL BASIC : навч. посіб. для студентів ВНЗ / П. С. Малачівський, Я. В. Пізюр. – Львів : Растр-7, 2014. – 407 с.
3. Уокенбах Дж. Excel 2013: профессиональное программирование на VBA = Excel 2013 Power Programming with VBA / Джозе Уокенбах. – М. : Диалектика, 2014. – 960 с.
4. Юрченко І. В. Інформатика та програмування. Частина 2 / І. В. Юрченко, В. С. Сікора. – Чернівці : Вид. Яворський С. Н., 2015. – 210 с.
5. Методичні вказівки до лабораторних, практичних, самостійних та контрольних робіт з дисципліни «Інформатика та комп'ютерна техніка» (для студентів 1 курсу денної і заоч. форми навчання за напрямками підгот. б. 170201 «Цивільний захист» і б. 170202 «Охорона праці» / Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова ; [уклад. : С. В. Дядюн, М. П. Пан, Г. В. Білогурова]. – Харків : ХНУМГ ім. О. М. Бекетова, 2015. – 113 с.

СОДЕРЖАНИЕ	
ВВЕДЕНИЕ.....	3
ТЕМА 1. ВВЕДЕНИЕ В VBA	4
1.1. Суть объектно-ориентированного программирования.....	4
1.2. Объект, свойство, метод, событие.....	5
1.3. Классы, основные объекты MS Office	6
ТЕМА 2. СОЗДАНИЕ И РЕДАКТИРОВАНИЕ МАКРОПРОЦЕДУР	8
2.1. MacroRecorder	8
2.2. Порядок работы с макросами в Word	8
2.3. Порядок записи макроса в Excel	11
2.4. Просмотр макроса	11
2.5. Создание макроса в редакторе VBA	12
ТЕМА 3. СОЗДАНИЕ ПРОСТЫХ ПРОГРАММ НА VBA	14
3.1. Иерархия объектов VBA	15
3.2. Некоторые объекты Word	16
3.3. Некоторые объекты Excel	18
3.4. Алфавит и основные конструкции VBA	19
3.5. Синтаксис	20
3.6. Типы данных.....	20
3.7. Переменные и константы.....	21
3.8. Преобразования.....	22
3.9. Область определения и видимости переменных и констант	23
3.10. Инструкция Option Explicit	23
3.11. Организация диалога	24
ТЕМА 4. ОПЕРАТОРЫ ЯЗЫКА VBA. ФУНКЦИИ	32
4.1. Операторы.....	33
4.2. Оператор присваивания	33
4.3. Функции языка VBA	34
ТЕМА 5. ЭЛЕМЕНТЫ УПРАВЛЕНИЯ.....	41
5.1. Элементы управленияActiveX.....	42
5.2. Создание и настройка элемента управления	43
5.3. Некоторые свойства элементов управления	43
ТЕМА 6. ПОЛЬЗОВАТЕЛЬСКАЯ ФОРМА	49
6.1. Пользовательская форма	50
6.2. Свойства, методы и события форм	52
6.3. Вызов формы	53
6.4. Элементы управления.....	53
Набор вкладок и набор страниц	53
ТЕМА 7. ПРОЦЕДУРЫ И ФУНКЦИИ ЯЗЫКА VBA. СТРУКТУРА И ТИПЫ ПРОЦЕДУР	61
7.1. Программный проект.....	61

7.2. Модули.....	62
7.3. Процедуры	64
7.4. Организация данных в программе. Массивы.....	65
ТЕМА 8. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ	69
8.1. ОПЕРАТОР УСЛОВНОГО ПЕРЕХОДА IF ... THEN ... ELSE	69
8.2. ОПЕРАТОР ВЫБОРА ВАРИАНТА SELECT CASE	70
ТЕМА 9. ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ	76
9.1. СТРУКТУРА И ПОРЯДОК ВЫПОЛНЕНИЯ ОПЕРАТОРА.....	77
FOR.NEXT.....	77
9.2. СТРУКТУРА И ПОРЯДОК ВЫПОЛНЕНИЯ ОПЕРАТОРА ЦИКЛА DO... LOOP.....	78
9.3. ОПЕРАТОР ЦИКЛА FOR-EACH-NEXT.....	79
ТЕМА 10. ОТЛАДКА ПРОГРАММ,.....	89
ПРОЦЕДУР И ФУНКЦИЙ	89
10.1. Средства отладки программ.....	89
10.2. Интеллектуальные возможности редактора VBA	91
10.3. Отладка программы	92
10.4. Панели системы контроля. Установка контрольных точек Прерывание по ошибке	92
ТЕМА 11. СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ОБЪЕКТОВ	100
11.1. Создание модулей класса	100
11.2. Процедуры PROPERTY LET, PROPERTY И	101
GETPROPERTYSET.....	101
ЗАКЛЮЧЕНИЕ	106
СПИСОК ЛИТЕРАТУРЫ	107

Навчальне видання

СУЧАСНІ ІНФОРМАЦІЙНІ ТЕХНОЛГІЇ В
ЕКОНОМІЦІ
ОСНОВИ ПРОГРАМУВАННЯ У СЕРЕДОВИЩІ
MS OFFICE

Навчальний посібник
Для студентів 4 курсу факультету «Бизнес-управління»,
які навчаються за спеціальністю – Економіка
(російською мовою)

В авторській редакції
Комп'ютерний набір С. Б. Данилевич

Підписано до друку 20.06.2017. Формат 60×84/16.
Папір офсетний. Гарнітура «Таймс».
Ум. друк. арк. 6,8. Обл.-вид. арк. 6,5
Тираж 300 пр. Зам. № /

План 2016/2017 навч. Р. поз. № 6 у переліку робіт кафедри

Видавництво
Народної української академії
Свідоцтво № 1153 від 16.12.2002.

Надруковано у видавництві
Народної української академії.

Україна, 61000, Харків, МСП, вул.. Лермонтовська, 27.