МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Белорусский национальный технический университет

Кафедра «Технология и методика преподавания»

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ VBA

Лабораторный практикум

Минск
БНТУ
2013

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет

Кафедра «Технология и методика преподавания»

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ VBA

Лабораторный практикум
по дисциплине «Информационные системы и сети»
для студентов специальности
1-02 06 02 «Технология. Дополнительная специальность»

Минск
БНТУ
2013

УДК 004.42(076.5)(075.8)
ББК 22.18я7
И73

Составители:
Н. И. Астапчик, А. Ю.Зуёнок

Рецензенты:
С. А. Иващенко, В. М. Комаровская

Интегрированная среда разработки VBA : лабораторный практи-
И73 кум по дисциплине «Информационные системы и сети» для студентов специальности 1-02 06 02 «Технология. Дополнительная специальность» / сост.: Н. И. Астапчик, А. Ю. Зуёнок. – Минск : БНТУ, 2013. – 82 с.
ISBN 978-985-550-124-5.

В издании описаны наиболее важные элементы интегрированной среды разработки VBA, основные технические приемы при создании и использовании макросов.

Практикум содержит примеры выполнения заданий в интегрированной среде разработки VBA.

Издание предназначено для студентов дневной формы получения высшего образования специальности 1-02 06 02 «Технология. Дополнительная специальность»

УДК 004.42(076.5)(075.8)
ББК 22.18я7

ISBN 978-985-550-124-5

ВВЕДЕНИЕ

Изучение дисциплины «Информационные системы и сети» способствует формированию у будущих специалистов современных взглядов на роль и место компьютерных технологий в профессиональной деятельности.

Базой для изучения дисциплины «Информационные системы и сети» является дисциплина «Основы информационных технологий». Также содержание дисциплины связано с изучением дисциплин «Технологии программирования и методы алгоритмизации», «Компьютерная графика и мультимедиа».

В ходе выполнения лабораторных работ (практикума) студент должен изучить теоретические сведения, письменно ответить на контрольные вопросы, выполнить практические задания, приведенные в качестве примеров, индивидуальные практические задания (по вариантам), оформить отчет по приведенной ниже форме, защитить работу.

Форма отчета

Лабораторная работа № _____.
Тема лабораторной работы: «_____»
Цель работы: _____.
Краткие теоретические сведения (ответы на контрольные вопросы).
Вывод о проделанной работе.

Лабораторная работа № 1

МАКРОСЫ В MS WORD И MS EXCEL

Цель работы: изучить способы создания, запуска и удаления макросов.

Теоретические сведения

Общие сведения о макросах в MS Office

Макрос – это набор команд и инструкций, группируемых вместе в виде единой команды для автоматического выполнения задачи.

Их применение существенно экономит время пользователя и избавляет его от необходимости выполнения рутинных операций. Цель макросов – автоматизировать повторяющиеся действия и расширить функциональные возможности приложения. Пользователи сами имеют возможность создавать макросы, исходя из своих потребностей и характера выполняемых задач. Однажды написанный макрос может быть многократно использован.

MS Office предлагает два способа создания макропрограмм: непосредственно ввод ее текста (процедуры VBA) в редакторе VBA либо применение встроенного средства записи. Автоматическое создание макросов осуществляется подобно записи на обычный магнитофон, только вместо звука фиксируются нажатия клавиш и действия мыши. Последовательность следующая: включить запись, произвести все операции, которые должен будет выполнять макрос, и остановить запись. Предварительно макропрограмме необходимо дать уникальное имя, описание (необязательно) и указать параметры сохранения – они несколько отличаются в различных приложениях MS Office, но в общем случае их два: сохранение макроса в текущем документе (в других он будет недоступен) либо в собственной библиотеке макросов или шаблоне (станет доступен для всех файлов в этом приложении). Библиотека макросов может быть перенесена на другой компьютер и подключена к соответствующему приложению MS Office, а макропрограмма, сохраненная в файле, копируется и переносится вместе с ним.

Для запуска созданного макроса можно вынести кнопку на панель быстрого доступа или назначить комбинацию клавиш. При записи автоматически генерируется текст макропрограммы – процедура VBA.

В ходе записи макроса важно быть аккуратным при всех операциях, избегая лишних действий, чтобы макропрограмма не содержала ненужных команд.

Общие сведения о макросах в MS Word

В приложении MS Word часто выполняемые задачи можно автоматизировать путем создания макросов. Как правило, макросы используются:

- для ускорения часто выполняемых операций редактирования или форматирования;
- объединения нескольких команд, например, вставки таблицы с определенными размерами, границами и числом строк и столбцов;
- упрощения доступа к параметрам в диалоговых окнах;
- автоматизации обработки сложных последовательных действий в задачах;
- преобразования документов, созданных в других текстовых редакторах;
- выполнения различных действий с выделенными данными.

Запись макроса

1. На вкладке **Вид** выбрать команду **Макросы** – **Запись макроса** (на вкладке **Разработчик** в группе **Код** выбрать команду **Запись макроса)**.

2. Ввести имя макроса в поле **Имя макроса**. Оно должно быть уникальным, отражать смысл макроса и не содержать пробелов.

3. В списке **Макрос доступен для** щелкнуть шаблон. Если макрос создается только для конкретного документа, следует выбрать из списка название конкретного документа. Щелкнуть кнопку **ОК**.

4. В поле **Описание** ввести описание этого макроса.

5. Выполнить одно из следующих действий:

- чтобы *начать запись макроса, не связывая его с кнопкой на панели быстрого доступа или сочетанием клавиш*, нажать кнопку **ОК**;
- чтобы *связать макрос с панелью быстрого доступа*, необходимо выполнить следующие действия:

1) нажать кнопку **Кнопка;**

2) в группе **Настройка панели быстрого доступа** выбрать документ или все документы, для которых требуется добавить макрос на панель быстрого доступа.

3) в диалоговом окне **Выбрать команды из** выбрать макрос, который требуется записать, и нажать кнопку **Добавить**;

4) чтобы начать запись макроса, нажать кнопку **ОК**;

• чтобы *назначить макросу сочетание клавиш*, необходимо выполнить следующие действия:

1) нажать кнопку **Клавишам**;

2) в списке **Команды** выбрать макрос, который требуется записать;

3) в поле **Новое сочетание клавиш** ввести любую последовательность клавиш и нажать кнопку **Назначить**;

4) чтобы начать запись макроса, нажать кнопку **Закрыть**.

6. Выполнить действия, которые следует включить в макрос.

Примечание. При записи макроса можно использовать мышь для выбора команд и параметров, но не для выделения текста. Для выделения текста необходимо использовать клавиатуру.

7. Чтобы остановить запись действий, выбрать команду **Остановить запись** в группе **Код** на вкладке **Разработчик**.

Запуск макроса

1. На вкладке **Разработчик** в группе **Код** нажать кнопку **Макросы**.
2. В поле **Имя макроса** ввести имя макроса, который нужно выполнить.
3. Нажать кнопку **Выполнить**.

Запуск макроса нажатием кнопки на панели быстрого доступа

Чтобы добавить на панель быстрого доступа кнопку для запуска макроса, необходимо выполнить следующие действия:

1. Вызвать контекстное меню на **Панели быстрого доступа**.
2. Выбрать пункт **Настройка панели быстрого доступа**.
3. В списке **Выбрать команды из** выбрать элемент **Макросы**.
4. Выбрать в списке созданный макрос и нажать кнопку **Добавить**.
5. Чтобы изменить изображение на кнопке макроса, выбрать макрос в поле, в которое он был добавлен, и нажать кнопку **Изменить**.
6. В поле **Символ** выбрать нужное изображение для кнопки.

7. Чтобы изменить имя макроса, которое отображается при наведении указателя мыши на кнопку, в поле **Отображаемое имя** ввести имя, которое требуется использовать.

8. Нажать кнопку **ОК** – кнопка макроса будет добавлена на панель быстрого доступа.

9. На панели быстрого доступа нажать добавленную кнопку макроса.

Запуск макроса щелчком области графического объекта

Можно создать на графическом объекте активную точку, щелчок по которой будет запускать макрос:

1. Вставить на лист графический объект (рисунок, картинку, фигуру).

2. Для создания активной области на существующем объекте нажать кнопку **Фигуры** в группе **Иллюстрации** на вкладке **Вставка**, выбрать одну из фигур и нарисовать ее на существующем объекте.

3. Щелкнуть созданную активную точку указателя мыши правой кнопкой мыши, а затем выбрать пункт **Назначить макрос**.

4. Выполнить одно из указанных ниже действий:

- чтобы назначить графическому объекту существующий макрос, следует дважды щелкнуть мышью по имени нужного макроса или ввести его имя в поле **Имя макроса**;
- чтобы записать новый макрос для назначения выделенному графическому объекту, нажать кнопку **Записать**, ввести имя макроса в диалоговом окне **Запись макроса** и нажать кнопку **ОК**, чтобы начать запись. Завершив запись макроса, нажать кнопку **Остановить запись** на вкладке **Разработчик** в группе **Код**;
- для редактирования существующего макроса – щелкнуть его имя в поле **Имя макроса**, а затем нажать кнопку **Изменить**.

5. Нажать кнопку **ОК**.

6. Выделить активную точку – появится область **Средства рисования** и вкладка **Формат**.

7. На вкладке **Формат** в группе **Стили фигур** щелкнуть стрелку рядом с командой **Заливка фигуры** и выберать пункт **Нет заливки**.

8. Щелкнуть стрелку возле кнопки **Контур фигуры** и выбрать вариант **Нет контура**.

Удаление макроса

На вкладке **Разработчик** в группе **Код** выбрать команду **Макросы**. Появившееся окно диалога **Макрос** содержит список макросов текущего документа. После выбора макроса, который предполагается удалить, станут доступными кнопки окна диалога **Макрос.** Нажать кнопку **Удалить**.

Общие сведения о макросах в MS Excel

В MS Excel макропрограммы могут быть использованы для создания и печати отчетов, применения специального форматирования к выбранным диапазонам ячеек, импортирования данных, автоматического построения диаграмм, написания собственных программ табличных вычислений и др.

В MS Excel запись, сохранение, быстрый запуск и удаление созданных макросов осуществляется по аналогии с MS Word.

Запуск макроса производится нажатием клавиши CTRL в сочетании с клавишей быстрого вызова.

1. На вкладке **Разработчик** в группе **Код** нажать кнопку **Макросы**.
2. В поле **Имя макроса** выбрать макрос, которому нужно назначить сочетание клавиши **CTRL** с клавишей быстрого вызова.
3. Нажать кнопку **Параметры** – отобразится диалоговое окно **Параметры макроса**.
4. В поле **Сочетание клавиш** ввести любую прописную или строчную букву для использования с клавишей **CTRL**.

Примечание. Выбранное сочетание клавиш заменяет все совпадающие стандартные сочетания клавиш на то время, пока открыта книга, содержащая данный макрос.

5. Введите описание макроса в поле **Описание**.
6. Нажмите кнопку **ОК**, чтобы сохранить изменения, а затем кнопку **Отмена**, чтобы закрыть диалоговое окно **Макрос**.

Настройка автоматического запуска макроса при открытии книги

Если макрос записан и сохранен с именем «Авто_открыть», он будет запускаться при каждом открытии содержащей его книги.

Создание макроса «Авто_открыть»

1. На вкладке **Разработчик** в группе **Код** выбрать команду **Безопасность макросов**.

2. В категории **Параметры макросов** в группе **Параметры макросов** нажать переключатель **Включить все макросы – ОК**.

3. Для сохранения макроса с конкретным документом сначала нужно открыть этот документ.

4. На вкладке **Разработчик** в группе **Код** нажать кнопку **Запись макроса**.

5. В поле **Имя макроса** ввести **Авто_открыть**.

6. В списке **Сохранить** выбрать документ, в котором нужно сохранить макрос. Нажать кнопку **ОК**, а затем выполнить действия, которые нужно записать.

7. На вкладке **Разработчик** в группе **Код** нажать кнопку **Остановить запись**.

Чтобы предотвратить автоматическое выполнение макроса «Авто_открыть» при запуске приложения, во время запуска удерживайте нажатой клавишу **SHIFT**.

Включение и отключение макросов в центре управления безопасностью MS Excel и MS Word

Параметры безопасности макросов доступны в центре управления безопасностью. При изменении параметров макроса в центре управления безопасностью они изменяются только для текущего приложения MS Office данного пользователя, а не для всех приложений.

Чтобы включить или отключить макросы в центре управления безопасностью нужно:

1. Нажать **кнопку Microsoft Office**, а затем кнопку **Параметры Excel (Параметры Word)**.

2. Выбрать категорию **Центр управления безопасностью**, нажать кнопку **Параметры центра управления безопасностью** и щелкнутье **Параметры макросов**.

3. Выбрать нужные параметры:

- **Отключить все макросы без уведомления**. Данный параметр следует выбирать при отсутствии доверия к макросам. В результате отключаются все макросы в документах и связанные с ними оповещения системы безопасности;

• **Отключить все макросы с уведомлением**. Данный параметр установлен по умолчанию. Выберите этот параметр, если нужно отключить макросы, но при их наличии необходимо получать оповещения системы безопасности. Это позволит включать макросы только в случаях, когда это требуется;

• **Отключить все макросы, кроме макросов с цифровой подписью**. Данный параметр идентичен параметру **Отключить все макросы с уведомлением,** за исключением того, что при наличии цифровой подписи надежного издателя макрос запускается только в случае, если данный издатель уже внесен в список надежных. Если издатель не внесен в этот список, появится уведомление. Таким образом, можно выбрать нужный вариант: включить макрос, содержащий цифровую подпись или занести издателя в список надежных. Все макросы, не содержащие цифровой подписи, отключаются без уведомления;

• **Включить все макросы** (***не рекомендуется, возможен запуск опасной программы***)**.** Данный параметр разрешает выполнение всех макросов. Компьютер становится уязвимым для потенциально опасного кода, поэтому использовать этот параметр не рекомендуется;

• **Доверять доступ к объектной модели проектов VBA.** Этот параметр предназначен для разработчиков и позволяет явно заблокировать или разрешить программный доступ к объектной модели VBA от любого клиента автоматизации. Другими словами, он позволяет защитить код, созданный для автоматизации программ MS Office и программного управления средой Microsoft Visual Basic для приложений (VBA) и объектной моделью. Этот параметр можно установить для отдельного пользователя или приложения. По умолчанию доступ запрещен. Этот параметр безопасности затрудняет несанкционированным программам создание самореплицирующегося кода, который может причинить вред системе пользователя. Чтобы любой клиент автоматизации смог получить программный доступ к объектной модели VBA, пользователь, выполняющий код, должен явным образом предоставить его. Чтобы включить доступ, следует установить имеющийся флажок.

Функции, необходимые для выполнения работы

- **СЧЁТЕСЛИ (диапазон; критерий)**

Диапазон – это одна или несколько ячеек подряд, включающие числа или имена, массивы или ссылки, содержащие числа. Пустые ячейки и текстовые значения не учитываются.

Критерий – критерий в форме числа, выражения, текста или ссылки на ячейку, который определяет, какие ячейки нужно подсчитывать. Например, критерий может быть выражен следующим образом: *32*, *«32»*, *«>32»*, *«яблоки»* или *B4*;

- **СЕГОДНЯ()**

Возвращает текущую дату в числовом формате. Числовой формат даты – это код дата и времени, с помощью которого в Microsoft Excel производятся вычисления над датами и промежутками времени. Если до ввода этой функции для ячейки был задан формат **Общий**, результат будет отформатирован как дата.

Контрольные вопросы

1. Что такое макрос?
2. Перечислите способы создания макросов.
3. Для чего используются макросы в MS Word?
4. Для чего используются макросы в MS Excel?
5. Какие требования предъявляются к имени макроса?
6. Как записать макрос?
7. Как назначить макрос кнопке?
8. Как назначить макросу сочетание клавиш?
9. Перечислите способы запуска макроса.
10. Как создать макрос «Авто_открыть»? В чем его особенность?
11. Как удалить макрос?
12. Перечислите параметры макросов, которые можно задать в центре управления безопасностью.

Пример

Заменить сочетания двух–четырех стоящих подряд пробелов одним.

1. Ввести текст, следующего содержания:

Информатика – молодая научная дисциплина, изучающая вопросы, связанные с поиском, сбором, хранением, преобразованием и использованием информации в самых различных сферах человеческой деятельности. Генетически информатика связана с вычислительной техникой, компьютерными системами и сетями, так как именно компьютеры позволяют порождать, хранить и автоматически перерабатывать информацию в таких количествах, что научный подход к информационным процессам становится одновременно необходимым и возможным.

После второй мировой войны возникла и начала бурно развиваться **кибернетика** как наука об общих закономерностях в управлении и связи в различных системах: искусственных, биологических, социальных.

Впервые термин «кибернетика» ввел французский физик Андре Мари Ампер в первой половине XIX в. Он занимался разработкой единой системы классификации всех наук и обозначил этим термином гипотетическую науку об управлении, которой в то время не существовало, но которая, по его мнению, должна была существовать. Само название происходит от греческого слова *kybeneticos* – искусный в управлении.

Текст, должен содержать последовательности из двух–четырех пробелов.

2. Активизировать команду записи макроса.

3. В появившемся диалоговом окне **Запись макроса** ввести имя создаваемого макроса («ЛишниеПробелы»). В графе **Макрос доступен для** выбрать **Для всех документов**. В поле **Описание** кратко указать, для чего предназначен макрос.

4. В том же диалоговом окне нажать кнопку **Назначить макрос клавишам.** Указать желаемую комбинацию (например, **Ctrl + Alt + + Пробел**) и нажать **Назначить,** затем **Закрыть.**

5. Запись макроса начата (о чем свидетельствует типичный значок рядом с курсором). Вызвать диалоговое окно **Найти и заменить** (на вкладке **Главная** в меню **Редактирование** нажать кнопку **Заменить**).

6. В окне **Найти и заменить** в поле **Найти** ввести два пробела. В поле **Заменить на** поставить один пробел. После этого нажать на кнопку **Больше** для указания особых параметров поиска. Указать **Направление** («везде») и снять галочки со всех опций, кроме **Подстановочные знаки**. Нажимать кнопку **Заменить все** до тех пор, пока не появится сообщение о том, что произведено ноль замен.

7. Остановить запись макропрограммы – нажать на кнопку **Остановить запись** (вкладка **Вид**, команда **Макросы**).

Макрос записан. Теперь при нажатии комбинации клавиш **Ctrl** + + **Alt** + **Пробел** из текущего документа будут автоматически удаляться лишние пробелы.

Задания

Задание 1

Создать макрос Word, который будет автоматически преобразовывать слова и фразы определенного стиля в сноски.

При наборе текста некоторые слова и фразы необходимо выделять специальным стилем (например, **Примечание**). Затем, по окончании работы, макрос автоматически преобразует все подобные участки в сноски.

Исходный текст – из примера 1. Несколько участков надо определить как имеющие стиль **Примечание**. Для этого необходимо этот стиль создать (рис. 1.1).

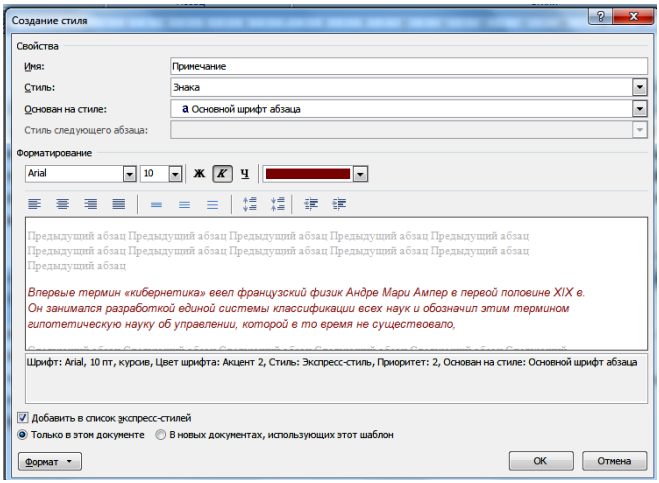

Рис. 1.1. Создание стиля «Примечание»

Информатика – молодая научная дисциплина, изучающая вопросы, связанные с поиском, сбором, хранением, преобразованием и использованием информации в самых различных сферах человеческой деятельности. Генетически информатика связана с вычислительной техникой, компьютерными системами и сетями, так как именно компьютеры позволяют порождать, хранить и автоматически перерабатывать информацию.

После второй мировой войны возникла и начала бурно развиваться **кибернетика** – *наука об общих закономерностях в управлении и связи в различных системах: искусственных, биологических, социальных.*

Впервые термин «кибернетика» ввел французский физик Андре Мари Ампер в первой половине XIX в. Он занимался разработкой единой системы классификации всех наук и обозначил этим термином гипотетическую науку об управлении. Само название происходит от греческого слова *kybeneticos – искусный в управлении*.

Последовательность действий при записи макроса:

- установить курсор в начале документа;
- найти текст со стилем **Примечание** и выделить его;
- вырезать найденный текст в буфер обмена;
- создать в месте расположения курсора сноску (курсор немного сместить, чтобы сноска выглядела должным образом);
- курсор перейдет в панель сносок;
- вставить вырезанный ранее текст из буфера обмена в сноску;
- из панели сносок перейти обратно в текст.

Для обработки всего текста потребуется запустить макрос несколько раз.

После проверки задания преподавателем макрос следует удалить.

Задание 2

Исходные данные см. в табл. 1.

Таблица 1

Исходные данные

Сведения о студентах					
№	Фамилия, имя	Дата рождения	Возраст	Средний балл	Стипендия
1	Александров Сергей	12.02.1993		5,4	
2	Власов Константин	03.04.1992		9,4	

Окончание табл. 1

Сведения о студентах					
№	Фамилия, имя	Дата рождения	Возраст	Средний балл	Стипендия
3	Волосюк Светлана	25.06.1990		8,2	
4	Иванова Наталья	27.04.1994		7,3	
5	Кириллов Николай	15.03.1993		6,1	
6	Королев Алексей	12.05.1993		7,9	
7	Николаева Юлия	01.10.1992		8,4	
8	Ничипорук Владислав	30.08.1991		6,5	
9	Прищепа Ирина	06.01.1990		4,2	
10	Степанов Илья	14.02.1995		8,6	

1. Необходимо определить возраст студента и размер стипендии. Заполнить соответствующие столбцы.

Подсказка. Для определения возраста студента создать формулу с использованием функции **СЕГОДНЯ()** из категории **Дата и время.** Для правильного отображения возраста – создать свой пользовательский числовой формат. Для определения размера стипендии – использовать логическую функцию **ЕСЛИ**.

Если средний балл студента меньше 5,5 баллов, то стипендия не начисляется; если средний балл студента меньше 7, но больше 5 баллов то стипендия будет равна 250 000 руб.; если средний балл студента больше 7, но меньше 8,7 баллов то стипендия будет составлять 300 000 руб.; если средний балл более 8,7 баллов, то стипендия будет равна 360 000 руб.

2. Создать макрос, форматирующий таблицу (см. рис. 1.2):

Сведения о студентах					
	Фамилия, имя студента	**Дата рождения**	**Возраст**	**Средний балл**	**Стипендия**
1	*Александров Сергей*	12.02.1993	19	5,4	**- р.**
2	*Власов Константин*	03.04.1992	19	9,4	**360 000 р.**
3	*Волосюк Светлана*	25.06.1990	21	8,2	**300 000 р.**
4	*Иванова Наталья*	27.04.1994	17	7,3	**300 000 р.**
5	*Кириллов Николай*	15.03.1993	18	6,1	**250 000 р.**
6	*Королев Алексей*	12.05.1993	18	7,9	**300 000 р.**
7	*Николаева Юлия*	01.10.1992	19	8,4	**300 000 р.**
8	*Ничипорук Владислав*	30.08.1991	20	6,5	**250 000 р.**
9	*Прищепа Ирина*	06.01.1990	22	4,2	**- р.**
10	*Степанов Илья*	14.02.1995	17	8,6	**300 000р.**

Рис. 1.2. Результаты макроса

После проверки задания преподавателем макрос следует удалить.

Задание 3

1. Создать таблицу (см. рис. 1.3).

Календарь погоды																													
Дата	01.02.2012	02.02.2012	03.02.2012	04.02.2012	05.02.2012	06.02.2012	07.02.2012	08.02.2012	09.02.2012	10.02.2012	11.02.2012	12.02.2012	13.02.2012	14.02.2012	15.02.2012	16.02.2012	17.02.2012	18.02.2012	19.02.2012	20.02.2012	21.02.2012	22.02.2012	23.02.2012	24.02.2012	25.02.2012	26.02.2012	27.02.2012	28.02.2012	29.02.2012
Температура, °С	-5	-7	-9	-6	-3	0	0	-2	-10	-9	-8	-7	-2	-1	0	1	3	3	2	1	-2	-1	-7	-3	-2	0	1	5	4
Осадки	с	с	н	н	н	н	н	д	д	н	н	с	с	с	д	д	д	н	н	н	н	д	с	д	с	н	д	д	с

Рис. 1.3. Исходные данные:
с – снег; *д* – дождь; *н* – без осадков

2. Создать макрос, который, используя условное форматирование, будет определять: самый холодный день (синий цвет), самый теплый день (красный цвет), дни с нулевой температурой (желтый цвет).

Подсказка. Функция **Условное форматирование** в MS Excel позволяет применять формат к ячейке только в том случае, если содержащиеся в ней данные удовлетворяют определенным условиям.

Для того, чтобы воспользоваться функцией **Условное форматирование,** необходимо:

- выделить необходимый диапазон данных;
- выполнить команду **Главная** – **Условное форматирование** – **Создать правило**;
- в открывшемся диалоговом окне **Создание правила форматирования** произвести следующие настройки: **Стиль формата**: трехцветная шкала; **Среднее значение:** тип – число; значение – 0 (см. рис. 1.4).

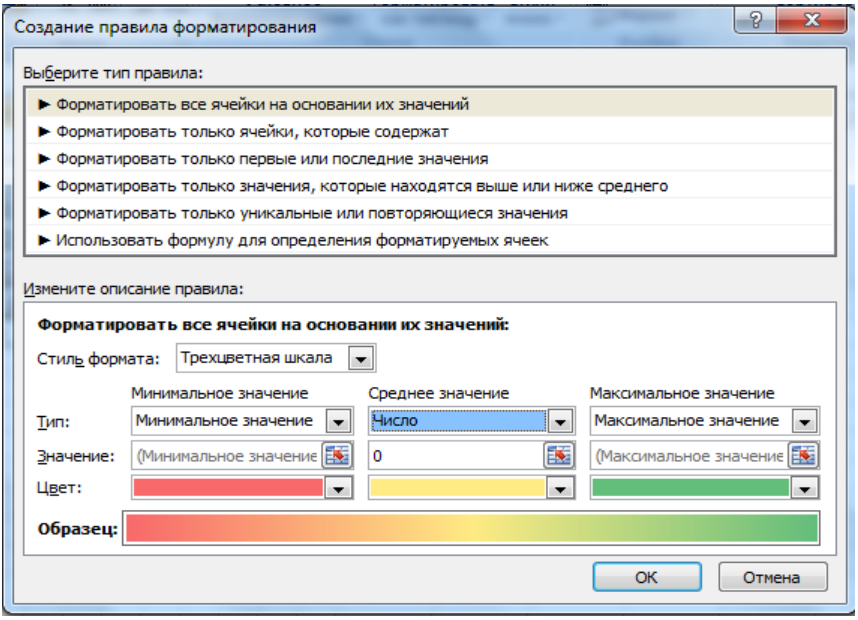

Рис. 1.4. Диалоговое окно «Создание правила форматирования»

Для удаления условного форматирования необходимо выбрать команду **Главная–Условное форматирование–Удалить правила–Удалить правила из выделенных ячеек**.

3. Назначить макросу сочетание клавиш и графический объект.

Подсчитать:

- среднемесячную температуру;
- количество дней, когда шел снег (для этого нужно в свободную ячейку ввести формулу: **=СЧЕТЕСЛИ(B5:AC5; «с»)**);
- количество дождливых дней;
- каких дней было больше: с положительной или с отрицательной температурой;
- количество дней за месяц, в которых температура была ниже средней;
- количество дней с отрицательной температурой в период с 5 по 20 февраля.

После проверки задания преподавателем макрос следует удалить.

Лабораторная работа № 2

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ VBA ОРГАНИЗАЦИЯ ВВОДА / ВЫВОДА ДАННЫХ

Цель работы: получить навыки работы в среде разработки **Visual Basic for Application**, изучить способы ввода / вывода информации.

Теоретические сведения

Способы запуска редактора Visual Basic

Прежде чем начать работать с редактором Visual Basic, нужно его открыть. Во всех приложениях MS Office это делается одинаково:

- В меню **Разработчик** выбрать **Visual Basic**. Нажать **Alt + F11**;
- можно вызвать редактор при возникновении ошибки в макросе;
- можно открыть готовый макрос для редактирования в диалоговом окне **Макрос**. В окне редактора Visual Basic можно работать одновременно с работой в приложении, откуда этот редактор был вызван. Переход – через **Alt + Tab** (в редактор также можно вернуться, повторно нажав **Alt + F11**).

Интерфейс редактора VBA

IDE (интегрированная среда разработки) состоит из нескольких компонентов (рис. 2.1).

Главное меню – это строка текста, расположенная в верхней части окна Visual Basic и состоящая из нескольких пунктов:

- меню **File** (**Файл**) – предназначено для работы с файлами, из которых образуются приложения. В нем можно создавать, сохранять и печатать проекты;
- меню **Edit** (**Правка**) – выполняет стандартные операции с буфером обмена – вырезание, копирование и вставка. Они применяются не только к фрагментам программы, но и к управляющим элементам;

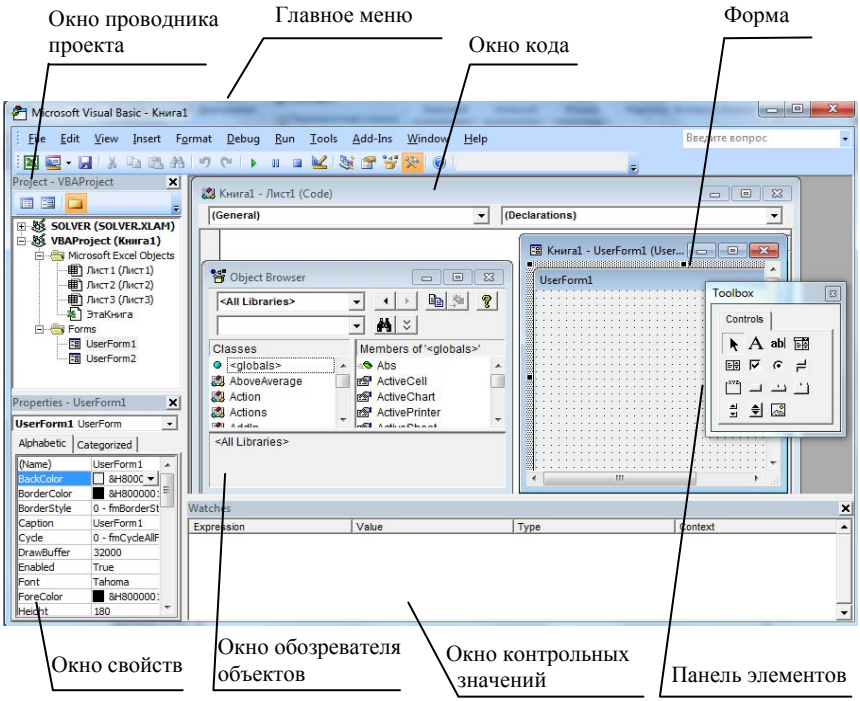

Рис. 2.1. Интерфейс интегрированной среды разработки VBA

- в меню **View** (**Вид**) включаются режимы просмотра различных компонентов и инструментов. Можно просматривать формы и программные модули;
- меню **Insert** (**Вставка**) позволяет добавлять процедуры, формы, модули и модули класса;
- команды меню **Format** (**Формат**) определяют расположение и размеры элементов и форм;
- при помощи команд меню **Debug** (**Отладка**) можно запустить и остановить приложение, расставить точки прерывания и выбрать просматриваемые объекты, а также выполнить другие операции, помогающие следить за работой приложения;

- команды меню **Run** (**Запуск**) – запускают и останавливают приложение, прерывают и возобновляют выполнение программы, что особенно удобно в процессе отладки;
- меню **Tools** (**Сервис**) – позволяет включить дополнительные элементы, запустить макросы и настроить параметры редактора;
- меню **Window** (**Окно**) – позволяет выстроить окна **IDE** (установить каскадное или мозаичное расположение), упорядочить значки свернутых форм, а также создает список, позволяющий быстро перейти к одному из открытых окон **IDE**;
- меню **Help** – помощь пользователю.

Для быстрого вызова главного меню необходимо нажать клавишу **F10**.

Панель инструментов находится под главным меню (см. табл. 2.1). Если она отсутствует, необходимо выполнить команду **View–Toolbars–Standard**.

Таблица 2.1

Назначение кнопок панели инструментов Standard

Кнопки панели инструментов Standard	Назначение
	Переход в приложение (в данном случае в Excel)
	Вставить объект
	Запуск проекта
	Прерывание программы
	Сброс
	Режим конструктора

Окончание табл. 2.1

Кнопки панели инструментов Standard	Назначение
	Окно проводника проекта
	Окно свойств
	Окно обозревателя объектов
	Панель элементов
	Справка

Окно проводника проекта по внешнему виду напоминает собой окно Проводника Windows и предназначено для быстрого просмотра составляющих проекта, который объединяет в себе все объекты, составляющие приложение. Это стандартные объекты открытого приложения MS Office (документ редактора MS Word, книга MS Excel и ее листы), формы, модули и классы.

Окно свойств отображает различные атрибуты выделенного объекта. Все объекты (формы, управляющие элементы и т. д.) имеют атрибуты, которые изменяют не только внешний вид объекта, но и его поведение. Все эти атрибуты называются свойствами. Следовательно, каждый объект обладает набором свойств.

Окно контрольных значений позволяет просматривать значения контрольных переменных в процессе проверки правильности работы (отладки) проекта, что позволяет находить ошибки в логике работы программ.

Конструктор форм расположен в центре экрана редактора VBA. Здесь выводится либо изображение формы, что позволяет производить визуальное конструирование макета формы и расположенных на ней элементов, либо окно программы.

Окно формы используется для создания диалоговых окон, разрабатываемых приложений в VBA. Форма в проект добавляется с помощью команды Вставка–Форма (Insert–Form) или нажатием кнопки **Вставить объект–UserForm.**

Объекты, их свойства и методы

Объект – комбинация кода и данных, которая может рассматриваться как единое целое, например, элемент управления, форма и компонент приложения. Каждый объект определяется по принадлежности к классу. Все визуальные элементы, такие как рабочий лист (Worksheet), диапазон (Range), диаграмма (Chart), форма (UserForm), являются объектами.

Объект обладает определенными свойствами и методами.

Свойства – это характеристики объекта такие, как размер, цвет, положение на экране или состояние объекта, например доступность или видимость.

Методы – это действия, выполняемые над объектом. Например:

Worksheets("Лист1").Visible = False	С помощью установки свойству «Видимость» значения «Ложь» скрывается рабочий лист «Лист 1»
Worksheets ("Лист 2").Delete	При помощи метода «Delete»» удаляется рабочий лист «Лист 2»

Программный объект может являться частью другого, большего программного объекта. Для доступа к свойствам и методам объекта, являющегося составной частью более крупного объекта, нужно определить каждый из сборных объектов, начиная с самого левого (большего) объекта, а затем через точку указать остальные составные объекты, один за одним, пока не будет определен объект, к свойствам и методам которого необходимо получить доступ. Например, с помощью кода

Workbooks("Книга1").Worksheets(“Лucm1").Range("A14").Font

можно получить доступ к рабочей книге «Книга1», рабочему листу «Лист1», шрифту ячейки «А14».

Объектами Excel являются таблицы, рабочие книги, диаграммы, области ячеек и др. Семейство представляет собой объект, содержащий несколько других объектов, как правило, одного и того же типа. Например, семейство **Workbooks** объединяет все открытые рабочие книги.

Обратиться к элементу семейства можно по имени или номеру. Например:

Worksheets(“Лист1”) или Worksheets(1)

Сохранение программы

1. Если программа сохраняется впервые или уже сохранена, а существующее имя не требует изменений, то сохранять можно как в VBA, так и в Excel;

2. Если уже существующее имя нужно изменить, то сохранять необходимо таким образом: выйти в Excel, выбрать меню **Файл–Сохранить как**.

3. В поле **Тип файла** необходимо указать **Книга Excel с поддержкой макросов.**

Правила записи операторов

При записи операторов необходимо придерживаться следующих правил:

- каждый новый оператор записывать с новой строки.
- чтобы записать несколько операторов на одной строке нужно, разделить их между собой двоеточием «:».
- если оператор не помещается в одной строке, то необходимо поставить в конце строки пробел и знак подчеркивания « _», а затем продолжить не поместивший текст на следующей строке.

Оператор присваивания присваивает значение выражения переменной, константе или свойству объекта. Оператор присвоения всегда включает знак равенства «=».

Синтаксис:

Переменная = Выражение

или

Постоянная = Выражение

Оператор присваивания предписывает выполнить выражение, заданное в его правой части, и присвоить результат переменной, имя которой указано в левой части.

Организация ввода / вывода

Решение любой задачи имеет три части – ввод данных, обработка данных, вывод результата.

Под **вводом данных** понимается описание всех переменных, констант и массивов, используемых в программе, а также код, обеспечивающий присвоение этим переменным вводимых данных.

Под **обработкой данных** понимается код, состоящий из математических выражений, которые приводят к получению результата.

Вывод результата – это код программы, который позволяет отобразить полученный результат в необходимом виде: на экране (лист Excel, форма), на принтере и т. д.

Ввод/вывод данных можно осуществлять несколькими способами:

- записью и чтением данных из ячеек Excel;
- с помощью диалоговых окон **(InputBox–MsgBox)**;
- конструирование собственных диалоговых окон **(UserForm)**.

Запись и чтение данных из ячеек Excel

Производить запись и чтение данных из ячеек Excel можно, используя два свойства: **Range** и **Cells**.

Свойство **Range** требует один аргумент – имя ячейки, записанное при помощи относительной или абсолютной адресации. Например:

Range(“A1”); **Range(“A1”)**

Свойству **Cells** требуется два аргумента, (номер строки и номер столбца), на пересечении которых находится ячейка. Например: **Cells(2,4)** (обращение к ячейке D2)

Ввод / вывод данных с использованием диалоговых окон InputBox и MsgBox

Наиболее часто в программах VBA встречаются две разновидности диалоговых окон: ***окна сообщений*** и ***окна ввода***.

Окно ввода создается и выводится на экран с помощью функции **InputBox**.

Функция **InputBox.** Выводит на экран диалоговое окно, содержащее сообщение, поле ввода и две кнопки – **OK** и **Cancel**. Уста-

навливает режим ожидания ввода текста пользователем или нажатия кнопки, а затем возвращает значение типа **String** при нажатию кнопки **OK**, содержащее текст, введенный в поле. Возвращает пустую строку при нажатии кнопки **Cancel**.

Синтаксис:

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

Обозначения:

- **prompt** – строковое выражение, отображаемое как сообщение в диалоговом окне. Строковое значение prompt может содержать несколько строк. Для разделения строк допускается использование символа возврата каретки **(chr (13))**, символа перевода строки **(chr(10))** или комбинацию этих символов **(chr(13) & Chr (10))**;
- **title** – строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку заголовка помещается имя приложения;
- **default** – строковое выражение, отображаемое в поле ввода как используемое по умолчанию, если пользователь не введет другую строку. Если этот аргумент опущен, поле ввода изображается пустым;
- **xpos** – числовое выражение, задающее расстояние по горизонтали между левой границей диалогового окна и левым краем экрана. Если этот аргумент опущен, диалоговое окно выравнивается по центру экрана по горизонтали;
- **ypos** – числовое выражение, задающее расстояние по вертикали между верхней границей диалогового окна и верхним краем экрана. Если этот аргумент опущен, диалоговое окно помещается по вертикали примерно на одну треть высоты экрана;
- **helpfile** – строковое выражение, определяющее имя файла справки, содержащего справочные сведения о данном диалоговом окне. Если этот аргумент указан, необходимо также наличие аргумента **Context**;
- **context** – числовое выражение, определяющее номер соответствующего раздела справочной системы. Если этот аргумент указан, необходимо также наличие аргумента **helpfile**.

Например:

Name=InputBox(“Введите Ваше имя”, “Пример окна ввода”)

На экране появится окно (рис. 2.2).

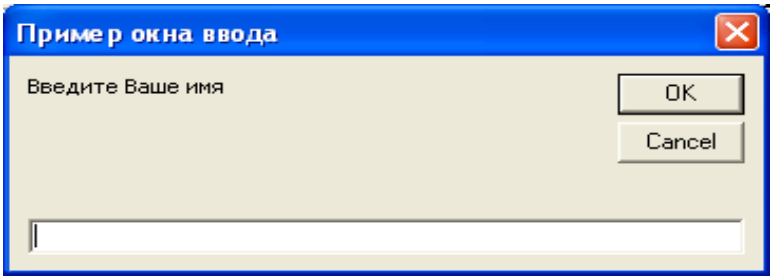

Рис. 2.2. Пример окна ввода

Переменной **Name** будет присвоено значение типа **String**, введенное пользователем.

Так как введенные пользователем данные считаются текстом, при вводе числовых значений необходимо преобразовать их к одному из числовых типов данных с помощью функции преобразования типа. Например:

X = CDbl(InputBox(“Введите значение X”, “Пример окна ввода”, “1,678”))

Введенное пользователем значение будет преобразовано к типу **Double** и присвоено переменной ***X***. Если пользователь не будет вводить значение, а просто нажмет кнопку **OK**, переменной ***X*** будет присвоено значение по умолчанию – 1,678.

Окно сообщения используется для предоставления информации пользователю. Представляет возможность задать вывод на экран окна сообщения, в котором пользователь должен щелкнуть на одной из кнопок, прежде чем продолжить работу.

Процедура **MsgBox.** Выводит на экран диалоговое окно, содержащее сообщение, устанавливает режим ожидания нажатия кнопки пользователем, а затем возвращает значение типа **integer**, указывающее, какая кнопка была нажата. Синтаксис:

MsgBox(prompt[, buttons] [, title])

Пояснения:

- **prompt** – строковое выражение, отображаемое как сообщение в диалоговом окне;
- **buttons** – числовое выражение, представляющее сумму значений, которые указывают число и тип отображаемых кнопок, тип используемого значка, основную кнопку
- **title** – строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку заголовка помещается имя приложения.

Значения параметра **Buttons** процедуры **MsgBox**, определяющие отображаемые в диалоговом окне кнопки, приведены в табл. 2.2.

Таблица 2.2

Значения параметра **Buttons** процедуры **MsgBox**

Константа	Значение	Отображаются кнопки
vbOKOnly	0	OK
VbOKCancel	1	OK, Отмена
VbAbortRetryIgnore	2	Стоп, Повтор, Пропустить
VbYesNoCancel	3	Да, Нет, Отмена
VbYesNo	4	Да, Нет
VbRetryCancel	5	Повтор, Отмена

При написании программ с откликом, когда нужно знать, какая кнопка диалогового окна была нажата, вместо возвращаемых значений удобнее использовать константы VBA, которые делают код программы более читаемым и, к тому же, их легко запомнить (табл. 2.3, 2.4, 2.5).

Таблица 2.3

Значение констант VBA

Константа	Значение	Отображаются кнопки
vbOK	1	OK
VbCancel	2	Отмена

Окончание табл. 2.3

Константа	Значение	Отображаются кнопки
VbAbort	3	Прервать
vbRetry	4	Повторить
vbIgnore	5	Пропустить
vbYes	6	Да
vbNo	7	Нет

Таблица 2.4

Значения параметра Buttons процедуры MsgBox, определяющие основную кнопку

Константа	Значение	Описание
vbDefaultButton1	0	First button is default (default)
vbDefaultButton2	256	Second button is default
vbDefaultButton3	512	Third button is default
vbDefaultButton4	768	Fourth button is default

Таблица 2.5

Значения параметра Buttons процедуры MsgBox, определяющие отображаемые в диалоговом окне используемые информационные значки

Константа	Значение	Описание	Вид значка
vbCritical	16	Критическое сообщение	
vbQuestion	32	Предупреждение запроса	
vbExclamation	48	Предупреждающее сообщение	
vbInformation	64	Информационное сообщение	

Например:

N = MsgBox (“Значение переменной X=” & X & Chr(10) & “Продолжить вычисления?”, VbYesNo, “Пример окна MsgBox”)

Пример окна MsgBox, если к моменту выполнения данного оператора переменная ***X*** равнялась числу 2,14587895, приведен на рис. 2.3.

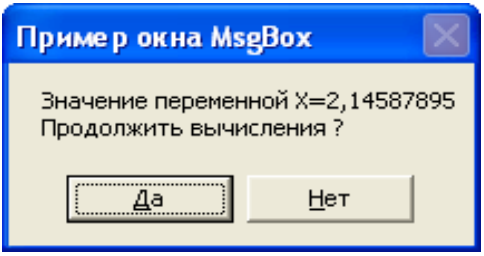

Рис. 2.3. Пример окна MsgBox с кнопками «Да» и «Нет»

Пользователь может нажать одну из кнопок: «Да» или «Нет». Если будет нажата кнопка «Да», переменной ***N*** будет присвоено значение 6, если будет нажата кнопка «Нет» – 7. Проанализировав в дальнейшем это значение, можно выбрать одну из ветвей выполнения программы.

Часто процедура MsgBox используется в «минимальном» варианте – только для вывода сообщения с одной кнопкой – **OK**. В этом случае аргументы не берутся в скобки. Например (рис. 2.4):

MsgBox (“Значение переменной X=” & X)

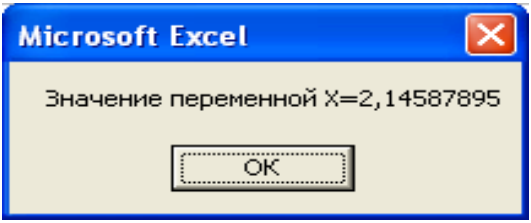

Рис. 2.4. Пример окна MsgBox с кнопкой «OK»

Например (рис. 2.5):

MsgBox ("Внимание!!", 5 + 256 + 48)

MsgBox ("Внимание!!", vbRetryCancel + vbDefaultButton2 + +vbExclamation)

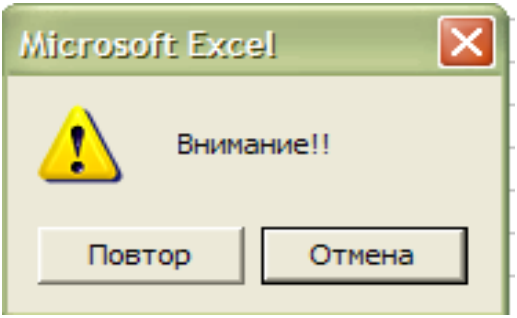

Рис. 2.5. Пример окна MsgBox с предупреждающим сообщением

Контрольные вопросы

1. Перечислите способы запуска редактора VBA.
2. Перечислите основные элементы интерфейса IDE VBA.
3. Перечислите основные пункты главного меню.
4. Какие кнопки содержит панель инструментов Standard?
5. Как сохранить созданную в редакторе VBA программу?
6. Дайте определение понятиям объект, свойство, метод.
7. Что является объектами MS Excel?
8. Перечислите правила записи операторов.
9. Каков синтаксис оператора присваивания?
10. Перечислите способы организации ввода/вывода данных.
11. С помощью каких свойств осуществляется запись и чтение данных из ячеек MS Excel? Приведите примеры.
12. Приведите пример синтаксиса функции InputBox. Каков смысл переменных, используемых в данной функции?
13. Приведите синтаксиса процедуры MsgBox. Каков смысл переменных, используемых в данной процедуре?

Примеры

Пример 1

Создать программу для вычисления выражения $c = \sin a / \cos b$.
Порядок выполнения:
1. Ввести в ячейках A1, B1 числа.
2. Вызвать редактор VBA.
3. Выполнить команду **Insert–Module.**
4. Выполнить команду **Insert–Procedure.**
5. Ввести имя процедуры (по правилу именования).
6. Ввести текст программы:

```
Public Sub prog1()                          'заголовок процедуры
Dim a As Double, b As Double, c             'раздел описания переменных
As Double
'исполняемая часть программы
a = Work-                                   'присваивание идентификатору
sheets(1).Range("a1").Value                 a числового значения
' ячейки a1 рабочего листа
b = Work-                                   'присваивание идентификатору
sheets(1).Range("b1").Value                 b  числового значения
' ячейки b1 рабочего листа
c = Sin(a) / Cos(b)                         'вычисление арифметического
                                            выражения
Worksheets(1).Range("c1").Value = c         'передача вычисленного значе-
                                            ния в ячейку c1 рабочего листа 1
End Sub                                     'конец процедуры
```

7. Запустить программу на выполнение (**F5** или **Кнопка**).
8. Просмотреть результаты выполнения на рабочем листе.

Пример 2

Использование диалоговых окон.
1. В редакторе VBA выполнить команду **Insert–Module.**
2. Ввести текст программы:

```
Sub Тест()
    Dim msg As String, style As Integer, title As String, x
    Dim default
    msg = «Мои первые шаги в VB!»
    title = “Сообщение с одной кнопкой”
    MsgBox msg, , title
    title = “Сообщение с двумя кнопками”
    style = vbYesNo
    MsgBox msg, style, title
    title = “Сообщение с двумя кнопками и значком”
    style = vbYesNo + vbCritical
    MsgBox msg, style, title
    msg = “Введите число, кратное 16, не больше 64!”
    title = ”Формируем окно для ввода информации ”
    default = 16
    x = InputBox (msg, title, default,1000,500)
    msg = “ Мои первые шаги в VB!”
    title = “ Мое сообщение”
    style = vbYesNo + x
    MsgBox msg, style, title
End Sub
```

3. Запустить программу на выполнение.

Пример 3

Использование диалоговых окон.

1. В редакторе VBA выполнить команду **Insert–Module.**
2. Ввести текст программы. Приведенная в примере 2 программа может быть переписана следующим образом:

```
Sub Тест1()
    Dim x as Byte
    MsgBox «Мои первые шаги в VB!», , «Сообщение с одной
            кнопкой»
    MsgBox «Мои первые шаги в VB!», vbYesNo, «Сообщение с
            двумя кнопками»
```

```
    MsgBox «Мои первые шаги в VB!», vbYesNo + vbCritical,
        «Сообщение с двумя кнопками и значком»
    x = InputBox («Введите число, кратное 16, не больше 64!»,
        «Формируем окно для ввода информации», 16, 500,
        1000)
    MsgBox «Мои первые шаги в VB!», vbYesNo + x, «Мое со-
        общение»
End Sub
```

3. Запустить программу на выполнение.

Пример 4

Создать программу для вычисления выражения c = sin *a* / cos *b* с использованием диалоговых окон.

1. В редакторе VBA последовательно выполнить команду **Insert–Module; Insert–Procedure** ввести имя процедуры (по правилу именования), ввести текст программы:

```
Public Sub Dialog()
    Dim a As Double, b As Double, c As Double
    a = CDbl(InputBox("Введите a", "Задача", 2.4))
    b = CDbl(InputBox("Введите b", "Задача", 1.4))
    c = Sin(a) / Cos(b)
    MsgBox "Значение выражения c = Sin(a) / Cos(b) получи-
лось=" & c, vbInformation, "Ответ"
End Sub
```

2. Запустить программу на выполнение.

Задания

Задание 1

Организовать ввод с клавиатуры двух чисел. Окна ввода должны иметь различные заголовки, появляться в разных частях экрана, содержать разные сообщения и неравные значения в поле ввода по умолчанию.

Требуется найти *сумму*, *произведение*, *частное* и *остаток от деления* первого числа на второе (**Mod**). Результаты операций вывести на экран. Все окна сообщений должны иметь различное количество кнопок и разные значки.

Задание 2

Написать программу, осуществляющую простейший диалог с пользователем. Вначале должно появиться окно ввода, значением по умолчанию в котором является строка, содержащая Ваше имя. После обработки полученных данных, на экране должно появиться окно сообщения с текстом приветствия, содержащего обращение по имени, введенному ранее.

Задание 3

Создать программу для вычисления выражения (по вариантам. Номер варианта выдает преподаватель). При этом:

1. Ввод данных осуществляется через окно **InputBox**.
2. Задать различные начальные значения переменных.
3. Вывод результата осуществляться в окно **MsgBox**.
4. Введенные данные и результат должны сохраняться в таблице Excel (см. табл. 2.6):

Таблица 2.6

Результаты вычислений задания 3

Значение A	Значение B	Значение выражения
5	25	0,2

5. Использовать свойства Range и Cells.
6. Предусмотреть в окне MsgBox и в ячейках таблицы перенос текста на новую строку.

Лабораторная работа № 3

ТИПЫ ДАННЫХ. ВЫРАЖЕНИЯ

Цель работы: изучить типы данных, виды операций и выражений, приобрести навыки записи математических функций на основе линейных алгоритмов.

Теоретические сведения

Описание данных

Все объекты (данные), которыми оперирует язык программирования VBA, относятся к определенному типу. Тип данных определяет область возможных значений переменной, структуру организации данных, операции, определенные над данными этого типа. Типы данных подразделяются на *простые* (скалярные) и *сложные* (структурированные). У простых типов данных возможные значения данных едины и неделимы. Сложные же типы имеют структуру, в которую входят различные простые типы данных. Скалярные типы данных представлены в табл. 3.1.

Таблица 3.1

Скалярные типы VBA

Имя типа		Возможные значения
Boolean	Логический	True, False
Byte	Байтовый	от 0 до 255
Integer	Целое	от –32768 до +32767
Long	Длинное целое	от –2147483648 до +2147483647
Single	Число с плавающей точкой	от –3,4E38 до –1,4E – 45 – для отрицательных значений; от 1,4E – 45 до 3,4E38 – для положительных значений
Double	Число с плавающей точкой двойной точности	от –1,7E308 до –4,9E – 324 – для отрицательных значений; от 4,9E – 324 до 1,7E308 – для положительных значений

Окончание табл. 3.1

Имя типа		Возможные значения
Currency	Денежный	Десятичные числа с фиксированной позицией запятой: допускается до пятнадцати цифер до запятой и до четырех – после запятой
String	Строковый	В зависимости от два вида строк: строки фиксированной длины (до 2^{16} символов) и строки переменной длины (до 2^{31} символов); данные записываются в кавычках
Date	Дата	Даты изменяются в диапазоне от 1.01.100 года до 31.12.9999 года
Object	Объект	Ссылка на объект (указатель)
Variant	Вариант	Универсальный тип, значением которого могут быть данные любого из перечисленных выше типов, объекты, значения NULL и значения ошибок ERROR

Переменные в программе можно описывать или не описывать. В последнем случае будет присвоен тип **Variant**. Явно описывать переменную можно как в начале блока, так и в любом месте, где возникла необходимость использовать новую переменную. Лучше все переменные описывать явно и, как правило, в начале блока. Для запрета использования переменных, которые не были описаны явно, в начало программы необходимо вставить оператор **Option Explicit**.

Описание простых переменных

Переменные – это поименованные области в памяти компьютера. После вычисления какого-либо значения оно записывается в память, чтобы затем можно было к нему возвращаться. Использование переменных дает VBA возможность создавать прямое соответствие между областями памяти и заданным именем. Затем можно использовать это имя в программе.

Имя переменной может содержать цифры, буквы и знак подчеркивания, но обязательно должно начинаться с буквы.

Описание простых переменных имеет следующий синтаксис:

Dim ИМЯ_ПЕРЕМЕННОЙ As ИМЯ_ТИПА

Одним оператором **Dim** можно описать произвольное число переменных, но конструкция **As** должна быть указана для каждой из них, иначе переменным без **As** будет присвоен тип **Variant**.

Например:

Dim X As Byte, Z As Integer, C, Слово As String

Здесь *X* – это переменная байтового типа; *Z* – целого типа; *C* – типа вариант (по умолчанию); *Слово* – переменнвая строкового типа.

Описание констант

Константы, в отличие от переменных, не могут изменять свои значения. Использование констант делает программы легче читаемыми и позволяет проще вносить исправления – достаточно ввести новое значение при определении константы. Синтаксис должен быть следующим:

Const ИМЯ_КОНСТАНТЫ As
ИМЯ_ТИПА=ПОСТОЯННОЕ_ВЫРАЖЕНИЕ

Например:
Const Stavka As Single = 0.2
Const g = 9.8

Описание массивов

Массив – это структурированный тип данных, который представляет собой последовательность ячеек памяти, имеющих общее имя и хранящих данные одного типа. Каждый элемент массива определяется **индексом** (номером). Количество элементов в массиве называется **размерностью массива**. Массив описывается следующей конструкцией:

Dim ИМЯ_МАССИВА(СПИСОК_РАЗМЕРНОСТЕЙ) As
ИМЯ_ТИПА

В списке размерностей массива каждое измерение отделяется запятой и определяется заданием нижней и верхней границ изменения индексов.

Например:

Dim X(1 TO 5) As Integer, Y(1 To 10, 1 To 20) As Double

Здесь *X* – одномерный массив, состоящий из 5 элементов целого типа; *Y* – двумерный массив, стоящий из 10 строк и 20 столбцов с элементами числового типа двойной точности.

Имена в VBA

В VBA пользователь определяет имена переменных, функций, процедур, постоянных и других объектов. Вводимые пользователем имена должны отражать суть обозначаемого объекта так, чтобы делать программу легко читаемой. В VBA имеются следующие ограничения на имена:

1. Длина имени не должна превышать 255 символов.
2. Имя не может содержать стандартные разделители (точку, запятую, двоеточие, дефисов, пробелов и т. п.) и следующие символы: %, &, !, @, #, $.
3. Имя может содержать любую комбинацию не запрещенных символов, но начинаться должно с буквы.
4. Имена должны быть уникальны внутри области, в которой они определены.
5. Запрещено использовать имена, совпадающие с ключевыми словами VBA и именами встроенных функций и процедур.

Хотя регистр букв (верхний или нижний) в имени не имеет значения, его умелое использование может существенно облегчить понимание содержательной стороны переменной. Вместо плоских и невыразительных имен предпочтительнее использовать имена, которые легче воспринимаются благодаря выделению некоторых входящих в них символов разумным использованием верхнего регистра. Например, представляется более удачным вместо имен «процентнаяставка», «х-начзнач» использовать «ПроцентнаяСтавка», «х-НачЗнач».

Выражения и операции

Операции бывают арифметические, отношения и логические (табл. 3.2, 3.3)

Таблица 3.2

Арифметические операции VBA

Выражение	Операция	Пример		
		A	B	Результат
A+B	Сложение	5	2,75	7,75
A – B	Вычитание	5	2,75	2,15
A * B	Умножение	2	6	12
A / B	Деление	7	2	3,5
A \ B	Целочисленное деление	7	2	3
AmodB	Остаток от деления по модулю	7	2	1
A^B	Возведение в степень	2	3	8

Таблица 3.3

Операции отношения

Выражение	Операция
<	меньше
>	больше
<=	меньше или равно
>=	больше или равно
=	равно
<>	не равно

Логические операции: Not – логическое отрицание; **And** – логическое «И», **Or** – логическое «ИЛИ»; **Xor** – исключающее «ИЛИ».

Выражения устанавливают порядок выполнения действий над элементами данных. Выражения состоят из **операндов** и **знаков**

операций. **Операндами** являются константы, переменные, указатели функций, выражения, взятые в скобки.

Выражения бывают арифметические, отношения и логические.

Арифметические выражения записываются с помощью операндов числовых типов и арифметических операций, а результатом таких выражений является числовое значение. В арифметическом выражении можно использовать стандартные математические функции, которые приведены в табл. 3.4.

Выражения отношения определяют истинность или ложность результата при сравнении двух операндов. Сравнивать можно данные любого одинакового типа. Результат операции отношения только логический: **True** – «Истина» или **False** – «Ложь».

Логические выражения. Результатом логического выражения является логическое значение **True** или **False**. Простейшими видами логических выражений являются: логическая константа, логическая переменная, логическая функция, выражение отношения. Логические операции выполняются только над операндами логического типа (табл. 3.4).

Таблица 3.4

Стандартные математические функции VBA

Математическая запись	Имя функции в VBA	Описание
$\|x\|$	Abs(число)	Возвращает значение, тип которого совпадает с типом переданного аргумента, равное абсолютному значению указанного числа
arctg$X$	Atn(X)	Возвращает значение типа Double, содержащее арктангенс числа
cos$X$	Cos(X)	Возвращает значение типа Double, содержащее косинус угла
$]X[$	Int(X)	Возвращает значение типа, совпадающего с типом аргумента, которое содержит целую часть числа
ln$X$	Log(X)	Возвращает значение типа Double, содержащее натуральный логарифм числа

Окончание табл. 3.4

Математическая запись	Имя функции в VBA	Описание
e^X	Exp(*X*)	Возвращает значение типа Double, содержащее результат возведения числа **e**
sign*X*	Sgn(*X*)	Возвращает значение типа Variant (Integer), соответствующее знаку указанного числа
sin*X*	Sin(*X*)	Возвращает значение типа Double, содержащее синус угла
$X^{1/2}$	Sqr(*X*)	Возвращает значение типа Double, содержащее квадратный корень указанного числа
tg*X*	Tan(число)	Возвращает значение типа Double, содержащее тангенс угла

Таблица 3.5

Результаты логических операций для различных значений операндов

A	B	not A	A and B	A or B	A xor B
T	T	F	T	T	F
T	F	F	F	T	T
F	F	T	F	F	F
F	T	T	F	T	T

Чтобы получить перечень всех математических функций, достаточно набрать имя любой известной математической функции (например, sin), а затем нажать клавишу **F1** и внизу описания выбранной функции выбрать ссылку **Математические функции**. В полученном перечне можно получить справку о назначении любой из встроенных математических функций и ее аргументе.

Чтобы получить перечень всех производных математических функций и правила их формирования, достаточно набрать имя любой известной математической функции (например, sin), а затем нажать клавишу **F1** и внизу описания выбранной функции выбрать ссылку **Производные математические функции**.

Приоритет выполнения операций

Если выражение содержит несколько операций, то приоритет их выполнения следующий:

1. Сначала выполняются арифметические операции в таком порядке, как они представлены в табл. 3.6.

Таблица 3.6

Приоритет арифметических операций

Описание операции	Обозначение в VBA
Унарный минус (изменение знака)	– (в начале выражения)
Возведение в степень	^
Умножение и деление	*, /
Деление нацело и остаток от деления	\, Mod
Сложение и вычитание	+, –

2. Далее выполняются операции отношения (они имеют одинаковый приоритет).

3. Последними выполняются логические операции в таком порядке, как они представлены в табл. 3.7.

Таблица 3.7

Приоритет логических операций

Описание операции	Обозначение в VBA
Логическое отрицание	Not
Логическое И	And
Логическое ИЛИ	Or
Исключающее ИЛИ	Xor

4. Если выражение содержит несколько операций одинакового приоритета, то порядок их выполнения слева направо. Чтобы изменить порядок действий в выражении используются круглые скобки.

Функции преобразования типов

При обработке выражений, введенных с помощью функции **InputBox**, и последующем выводе результата на экран иногда приходится прибегать к функциям преобразования форматов (типов).

Преобразования строки в число и обратно осуществляются следующими функциями:

- **Val(<строка>)** – возвращает число, содержащееся в строке, как значение числового типа;
- **Str(<число>)** – возвращает значение типа **Variant (String)**, являющееся строковым представлением числа.

В качестве допустимого десятичного разделителя функция **Str** воспринимает точку. Часто возникает потребность в выводе нескольких результатов в одном окне сообщений. Помимо операции конкатенации (сложения строк – &) используют функцию **Chr**.

- **Chr(<число>)** – возвращает строку, ASCII-код которой равен аргументу. Например, **Chr (13)** – возвращает символ «возврат каретки», т. е. осуществляет переход на новую строку.

Контрольные вопросы

1. Что определяет тип данных? На какие группы делятся типы данных?
2. Перечислите целочисленные типы данных. Чем они отличаются?
3. Перечислите вещественные типы данных. Чем они отличаются?
4. Назовите логический, строковый, денежный, универсальный типы данных, тип даты, объект.
5. Что такое переменная? Как описать переменную?
6. Что такое константа? Как ее описать?
7. Что такое массив? Как его описать? Что такое размерность массива?
8. Из чего строится выражение?
9. Перечислите виды операций.
10. Перечислите арифметические операции.
11. Перечислите операции отношения.

12. Перечислите логические операции.
13. Перечислите стандартные математические функции.
14. Что является результатом логических операций?
15. Перечислите порядок выполнения операции в выражении.
16. Перечислите функции преобразования типов. Каково их назначение?

Примеры

Пример 1

Даны две переменные *X* и *Y*. Требуется произвести между ними обмен значениями. Для решения поставленной задачи применяется следующий код VBA:

```
Sub Обмен_двух()
    Dim X As Integer, Y As Integer, Z As Integer
        'Осуществим ввод значений;
        'функция Val используется для преобразования строки в число
    X = Val(InputBox("Введите первое число", "Ввод числа"))
    Y = Val(InputBox("Введите второе число", "Ввод числа"))
        'Осуществим обмен значений переменных
    Z = X
    X = Y
    Y = Z
        'Выведем результат в окне сообщений;
        'функция Str применяется для преобразования строки в число
        'функция Chr (в данном случае) – для организации перевода строки
    MsgBox "Первое число " & Str(X) & Chr(13) & "Второе число " & Str(Y), ,"Результат"
End Sub
```

Пример 2

Составим программу, в результате выполнения которой будет выведено значение **True**, если точка с заданными координатами (*x*, *y*) лежит внутри заштрихованной области и **False** в противном случае (см. рис. 3).

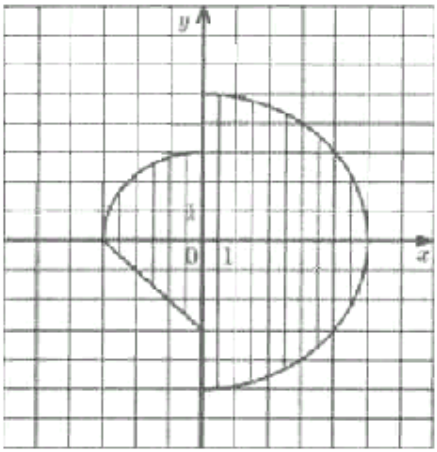

Рис. 3. Область для составления программы

Рассматриваемая область состоит из двух частей, каждая из которых описывается системой неравенств.

Первая часть: $x = 0$; $x^2 + y^2 = 9$; $y = -x - 3$.

Вторая часть: $x = 0$; $x^2 + y^2 = 25$.

Точка с координатами (x, y) лежит в заштрихованной области, если она принадлежит первой или второй части.

Программа осуществляет ввод координат точки, вычисление значения логического выражения, которое определяет принадлежность точки области и выводит полученную логическую величину на экран:

```
Sub Принадлежность_точки ()
    Dim x As Single, x As Single, L As Boolean
    x = Val(InputBox("Введите абсциссу", "Ввод координат"))
    y = Val(InputBox("Введите ординату", " Ввод координат"))
       L = (x <=0) And (x*x+ y*y<=9) And (y >= -x-3) Or (x >=0)
And (x*x+ y*y<=25)
       MsgBox "Лежит ли точка в заданной области?  " & L, ,
"Результат"
End Sub
```

Задание

Создать программу для вычисления выражения (задание выполняется по вариантам, выдается преподавателем).

Ввод и вывод данных осуществлять через диалоговые окна.

Отобразить исходные данные и результат в таблице Excel, снабдив их поясняющими надписями (образец см. в табл. 3.8).

Таблица 3.8

Образец выполнения

Числитель первой дроби	Знаменатель первой дроби	Числитель второй дроби	Знаменатель второй дроби	Результат	
				Числитель	Знаменатель
1	2	5	7	7	10

Лабораторная работа № 4

УСЛОВНЫЙ ОПЕРАТОР IF И ОПЕРАТОР ВЫБОРА SELECT CASE

Цель работы: приобрести навыки решения задач на основе разветвляющихся структур.

Теоретические сведения

Условный оператор IF

Для реализации разветвляющегося вычислительного процесса в VBA используется оператор **If … Then … Else**, который представляет собой простейшую форму проверки условий. Он имеет следующий синтаксис (однострочная форма):

If УСЛОВИЕ Then ОПЕРАТОР_1 Else ОПЕРАТОР_2

ОПЕРАТОР_1 выполняется, если **УСЛОВИЕ** истинно, в противном случае выполняется **ОПЕРАТОР_2**. При этом оператор **If … Then … Else** записывается в одну строку.

УСЛОВИЕ – это выражение логического типа; может быть простым и сложным. Результат выражения всегда имеет булевский тип.

При записи простых условий могут использоваться все возможные операции отношения. Сложные условия образуются из простых путем применения логических операций и круглых скобок.

В условном операторе допустимо использование блока операторов вместо любого из операторов. В этом случае условный оператор имеет вид (многострочная форма):

```
If УСЛОВИЕ  Then
        БЛОК_ОПЕРАТОРОВ_1
Else
        БЛОК_ОПЕРАТОРОВ_2
End If
```

Например (см. рис. 4.1):

If x > 0 then
*y = 2*x*
z = 1
Else
y = x–5
z = 0
End if

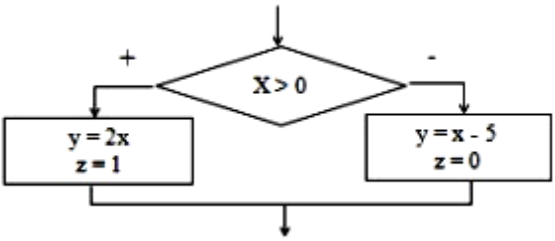

Рис. 4.1. Пример блок-схемы операторов
с одним условием

В условном операторе может проверяться несколько условий. В этом случае условный оператор имеет вид:

If УСЛОВИЕ_1 Then
БЛОК_ОПЕРАТОРОВ_1
Else If УСЛОВИЕ_2 Then
БЛОК_ОПЕРАТОРОВ_2
Else
БЛОК_ОПЕРАТОРОВ_3
End If
End If

Например (см. рис. 4.2):

If x >0 then
*y = 2*x*
Else
If x = 0 then
y = x–5
else
y = x^2
End if
End if

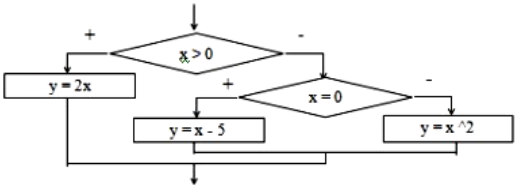

Рис. 4.2. Пример блок-схемы операторов
с несколькими условиями

Оператор выбора Select Case

Оператор **Select Case** удобно использовать, когда в зависимости от значения некоторого выражения, имеющего конечное множество допустимых значений, необходимо выполнить разные действия. Он также относится к условным операторам, но имеет другой вид:

```
Select Case  ПРОВЕРЯЕМОЕ_ВЫРАЖЕНИЕ
             Case ЗНАЧЕНИЯ_1
                    ОПЕРАТОРЫ_1
             Case ЗНАЧЕНИЯ_2
                    ОПЕРАТОРЫ_2
            . . .
             Case ЗНАЧЕНИЯ_N
                    ОПЕРАТОРЫ_N
             [Case Else
                    ИНАЧЕ_ОПЕРАТОРЫ]
End Select
```

ПРОВЕРЯЕМОЕ_ВЫРАЖЕНИЕ может иметь любой скалярный тип, кроме вещественного. **ЗНАЧЕНИЯ** состоят из произвольного количества значений или диапазонов, отделенных друг от друга запятыми. Тип **ЗНАЧЕНИЙ** должен совпадать с типом **ПРОВЕРЯЕМОГО_ВЫРАЖЕНИЯ**.

Сначала вычисляется **ПРОВЕРЯЕМОЕ_ВЫРАЖЕНИЕ**. Если его значение совпадает с одним из значений **ЗНАЧЕНИЯ_1**, то выполнятся **ОПЕРАТОРЫ_1** и управление передается оператору, стоящему после **End Select**. Если его значение не совпадает ни с одним из значений **ЗНАЧЕНИЯ_1**, то выполнятся **ИНАЧЕ_ОПЕРАТОРЫ** и управление передается оператору, стоящему после **End Select.**

Например, необходимо написать часть программы, определяющей значение переменной *S* в зависимости от значения переменной *n*. Для этого применяют следующий код (см. рис. 4.3):

```
Dim n as integer
Dim s as string
N=1997
Selest Case n
    Case 1960
        S=”Год рождения”
    Case 1961 to 1966
        S=”Дошкольные годы”
    Case 1967 to 1977, 1981 to1985
        S=”Годы учебы”
    Case else
        S=”Годы работы”
End Selest
  ‘Результат S=”Годы работы”
```

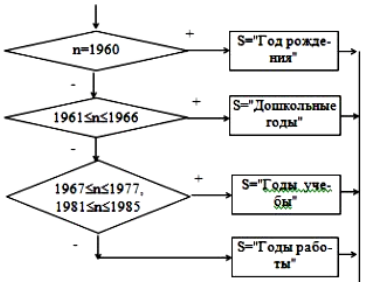

Рис. 4.3. Пример блок-схемы оператора выбора Selest Case

Контрольные вопросы

1. Какова однострочная форма условного оператора?
2. Как работает условный оператор? Что такое условие?
3. Приведите многострочную форму условного оператора.
4. Как выглядит условный оператор, в котором проверяется несколько условий?
5. Когда удобно использовать оператор выбора?
6. Каков синтаксис оператора выбора?
7. Как работает оператор выбора?

Примеры

Пример 1

Вычислить значение выражения для различных исходных данных.

1. В ячейку A1 ввести текст «Исходные данные», в ячейку A2 ввести текст «X=», в ячейку B2 ввести значение *X*.
2. В ячейку C1 ввести текст «Результат при x>0», в ячейку D1 ввести текст «Результат при x<=0».
3. В редакторе VBA ввести текст программы:

```
Sub IfThen ()
    X = Worksheets(1).Range("B2").Value
    If X > 0 Then
    f = X / 2
    Worksheets(1).Range("C2").Value = f
    Worksheets(1).Range("D2").Value = ""
    Else
    f = (X + 1) / 2: Worksheets(1).Range("D2").Value = f
    Worksheets(1).Range("C2").Value = ""
    End If
    MsgBox "Значение выражения =" & f, vbExclamation, "Результат"
End Sub
```

Пример 2

Найти наибольшее значение среди трех величин *A*, *B* и *C*.

Возникает следующая идея алгоритма решения этой задачи: сначала нужно найти большее значение среди *A* и *B* и присвоить его какой-то переменной, например, *D*. Затем найти большее среди *D* и *C*. Это значение можно присвоить той же переменной *D*.

Решение задачи сводится к двукратному применению алгоритма нахождения большего из двух. В структуре алгоритма содержится два последовательных ветвления: первое – полное, второе – неполное:

```
Sub Maximum()
    Dim a, b, c, d As Integer
    a = InputBox("Значение a", "Исходные данные", 1)
    Worksheets(1).Range("a2") = a
    b = InputBox("Значение b", "Исходные данные", 2)
    Worksheets(1).Range("b2") = b
    c = InputBox("Значение c", "Исходные данные", 3)
    Worksheets(1).Range("c2") = c
    If a > b Then d = a Else d = b
    If c > d Then d = c
    Worksheets(1).Range("d1") = "Максимум"
    Worksheets(1).Range("d2") = d
    MsgBox "Максимум из трех =" & d, vbInformation, "Задача решена"
End Sub
```

Пример 3

В старояпонском календаре был принят двенадцатилетний цикл. Годы внутри цикла носили названия животных: крысы, быка, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, петуха, собаки и свиньи. Написать программу, которая по номеру года определяет его название в старояпонском календаре, если известно, что 2008 год был годом крысы – началом очередного цикла.

Поскольку цикл является двенадцатилетним, нужно поставить в соответствие название года остатку от деления этого года на 12. При этом необходимо учесть, что остаток от деления 2008 на 12 равен 4.

Текст программы будет выглядеть следующим образом:

```
Option Explicit
Sub Goroskop()
    Dim Year As Integer
    Dim s As String
    Year = InputBox("Введите год", "Исходные данные")
    Select Case Year Mod 12
    Case 0
            s = "Год Обезьяны"
        Case 1
            s = "Год Петуха"
        Case 2
            s = "Год Собаки"
        Case 3
            s = "Год Свиньи"
        Case 4
            s = "Год Крысы"
        Case 5
            s = "Год Быка"
        Case 6
            s = "Год Тигра"
        Case 7
            s = "Год Зайца"
        Case 8
            s = "Год Дракона"
        Case 9
```

```
            s = "Год Змеи"
        Case 10
            s = "Год Лошади"
        Case 11
            s = "Год Овцы
End Select
MsgBox s, vbQuestion, "Результат"
End Sub
```

Пример 4

Найти наименьшее из двух действительных чисел, используя оператор выбора. Здесь селектором является логическая величина:

```
Option Explicit
Sub Minimum()
    Dim X As Single, Y As Single, Min As Single
    X = InputBox ("Введите первое число", "Ввод чисел для сравнения")
    Y = InputBox ("Введите второе число", "Ввод чисел для сравнения")
    Select Case X < Y
        Case True
            Min = X
        Case False
            Min = Y
End Select
MsgBox "Наименьшее из двух чисел: " & Min, , "Результат сравнения"
End Sub
```

Пример 5

По введенному с клавиатуры числу x определить, какое из условий выполняется: $x = 1$ или $x = -1$; $x \in [3; 5]$; $x > 5$. Выдать словесный ответ. В случае невыполнения условий, так же сообщить об этом.

В этой программе использованы различные формы записи условий после слова Case:

```
Option Explicit
Sub Diapason()
    Dim X As Single, S As String
    X = InputBox("Введите число", "Ввод")
    Select Case X
        Case 1, -1
            S = "выполнено первое условие"
        Case 3 To 5
            S = "выполнено второе условие"
        Case Is > 5
            S = "выполнено третье условие"
        Case Else
            S = "ни одно из условий не выполняется"
    End Select
    MsgBox "Проверка показала:" & Chr(13) & S, , "Результат"
End Sub
```

*Во **всех заданиях** исходные данные и результаты сохранять на листах рабочей книги Excel, а также для ввода и вывода информации использовать диалоговые окна.*

Задания

Задание 1

Создать программы для решения задачи (по вариантам, номер варианта выдает преподаватель).

Задание 2

Создать программу для решения задачи с использованием оператора Select Case (по вариантам, номер варианта выдает преподаватель.

Лабораторная работа № 5

СОЗДАНИЕ СОБСТВЕННЫХ ДИАЛОГОВЫХ ОКОН

Цели работы: изучить свойства, методы, события формы и элементов управления; приобрести практические навыки работы с формами, размещения элементов управления в форме, их редактирования.

Теоретические сведения

Форма, ее свойства, методы и события

Форма – это главный объект, образующий визуальную основу приложения. **Окно формы** используется для создания диалоговых окон разрабатываемых приложений в VBA. В приложении может быть как одна, так и несколько форм.

Как и любой другой объект VBA, форма имеет набор **свойств**, основные из которых приведены в табл. 5.1. Для получения справки по любому свойству достаточно выделить его в окне свойств и нажать **F1**.

Таблица 5.1

Основные свойства формы

Свойство	Описание
BackColor	Цвет фона для формы
BorderStyle	Определяет тип границы, окружающей форму
Caption	Текст, который выводится в заголовке формы
Font	Определяет тип и вид шрифта в форме
Height	Определяет высоту формы в твипах
(Name)	Имя объекта, для программы VBA
Width	Определяет ширину формы в твипах

Свойства можно изменять в режиме конструирования в окне свойств, либо программно в режиме выполнения. Например, в ходе выполнения программы можно изменить заголовок формы командой:

frmForm1.Caption = "Привет"

Программы в ОС Windows управляются **событиями**. Каждый раз, когда нажимается кнопка, перемещается мышь, изменяются размеры, формы и другие свойства объектов, ОС Windows генерирует сообщение. Сообщение доставляется соответствующему объекту, например, форме, а та генерирует соответствующее событие. Следовательно, можно составить фрагмент программы, в котором объект будет реагировать на событие определенным образом, т. е. любому стандартному событию соответствует определенная процедура. Чтобы просмотреть события связанные с формой, необходимо в режиме конструирования дважды щелкнуть на ней – появится окно программы, в котором нужно щелкнуть на списке **Процедура**. В табл. 5.2 приведены наиболее часто используемые события.

Таблица 5.2

Основные события форм

Событие	Описание
Initialize	Происходит во время конфигурации и до загрузки формы в память
Activate	Происходит после загрузки формы в память
Deactivate	Происходит, если форма перестает быть активной
Click	Происходит при нажатии левой кнопки мыши на форме

Следующий пример изменяет заголовок формы при активизации, и уменьшает размер формы после щелчка левой кнопкой мыши на форме.

```
Private Sub UserForm_Activate()
        frmForm1.Caption = "Щелчок на форме уменьшает ее
        размеры"
End Sub
Private Sub UserForm_Click()
        frmForm1.Width = frmForm1.Width / 2
        frmForm1.Height = frmForm1.Height / 2
        frmForm1.Caption = "ПОПРОБУЙ ЕЩЕ РАЗ."
End Sub
```

Также форма обладает набором методов и инструкций.

Метод определяет действие, которое может быть выполнено с объектом.

Инструкция инициирует действие. Она может выполнить метод или функцию.

В табл. 5.3 и 5.4 приведены наиболее часто используемые методы и инструкции для работы с формами.

Таблица 5.3

Основные методы форм

Метод	Описание
Hide	Скрывает объект UserForm, но не выгружает его
Show	Выводит на экран объект UserForm

Таблица 5.4

Основные инструкции форм

Инструкция	Описание
Load	Загружает объект UserForm, но не отображает его на экране
Unload	Удаляет объект UserForm из памяти

В следующем примере предполагается, что в проекте созданы две формы **frmForms**. При запуске проекта происходит событие **Initialize** для формы **frmForm1**, форма **frmForm2** загружается и выводится на экран. Когда при помощи мыши выбирается **frmForm2**, это форма делается невидимой, появляется форма **frmForm1**. Если же выбирается **frmForm1**, то **frmForm2** появляется вновь.

```
'Событие Initialize формы frmForm1.
Private Sub UserForm_Initialize()
        Load frmForm2
        frmForm2.Show
```

```
End Sub
' Событие Click для формы frmForm2
Private Sub UserForm_Click()
        frmForm2.Hide
End Sub
' Событие Click для формы frmForm1
Private Sub UserForm_Click()
        frmForm2.Show
End Sub
```

Форма в проект добавляется с помощью команды **Insert–UserForm** или нажатием кнопки **Вставить объект–UserForm.**

В результате на экран выводится незаполненная форма с **Панелью элементов**. Также **Панель элементов** отображается на экране либо выбором команды **View–Toolbox**, либо нажатием кнопки **Панель элементов**  а панели инструментов **Standard.** Используя **Панель элементов**, из незаполненной формы можно сконструировать любое требуемое для приложения диалоговое окно. Для создания элементов управления служат все кнопки панели инструментов, за исключением кнопки **Выбор объекта**  **.** Щелкнув по кнопке **Выбор объекта**, можно выбрать уже созданный в форме элемент управления для последующего его редактирования (изменения размеров или редактирования).

Для каждого объекта проекта необходимо определить его имя. В соответствии с общепринятыми соглашениями об именах объектов первые три символа имени должны отражать вид элемента, а остальные символы – назначение (см. табл. 5.5 и 5.6).

Таблица 5.5

Список основных элементов управления и соответствующих кнопок панели элементов

Элемент управления	Имя	Кнопка, его создающая	Элемент управления	Имя	Кнопка, его создающая
Поле	TextBox		Переключатель	OptionButton	

Окончание табл. 5.5

Элемент управления	Имя	Кнопка, его создающая	Элемент управления	Имя	Кнопка, его создающая
Надпись	Label		Флажок	CheckBox	
Кнопка	CommandButton		Выключатель	ToggleButton	
Список	ListBox		Рамка	Frame	
Поле со списком	ComboBox		Рисунок	Image	
Полоса прокрутки	ScrolBar		Набор страниц	MultiPage	
Счетчик	SpinButton		Набор вкладок	TabStrip	

Таблица 5.6

Рекомендуемые сочетания первых трех символов имен

Объект	Первые три символа имени	Пример имени
Форма	frm	frmMyForm
Надпись	lbl	lblInfo
Текстовое поле	txt	txtInput
Командная кнопка	cmd	cmdExit
Флажок	chk	chkSound
Переключатель	opt	optLevel
Список	lsb	lsbTypes
Рамка	fra	fraChoices
Полоса прокрутки	vcb	vcbSpeed
Рисунок	pic	picChema

Каждый управляющий элемент (объект) характеризуется набором свойств (которые можно изменять в режимах конструирования или выполнения), событий и методов.

Размещение нового управляющего элемента в форме осуществляется следующей последовательностью действий:

1. Щелкнуть значок того элемента, который нужно разместить в форме.

2. Поместить указатель мыши на то место, где будет располагаться управляющий элемент.

3. Нажать левую кнопку мыши и, не отпуская ее, растянуть появившийся прямоугольник до требуемых размеров.

4. Отпустить кнопку мыши.

После размещения элементов управления на форме необходимо связать объект на форме с кодом:

1. Дважды щелкнуть по элементу управления в форме. Появляется окно модуля для выбранного объекта. Выбрать событие, для которого требуется создать процедуру обработки, в списке, расположенном в верхнем правом углу окна модуля. Ввести текст процедуры.

2. Вызвать контекстное меню необходимого объекта правой клавишей мыши и нажать на поле **Программа**.

Размеры формы и расположенных на ней элементов управления можно изменять. Технология изменения размеров стандартная для ОС Windows: следует выделить изменяемый элемент, разместить указатель мыши на одном из размерных маркеров и протянуть его нажатой левой кнопкой мыши так, чтобы объект принял требуемые размеры. Окно редактирования форм поддерживает операции буфера обмена. Таким образом, можно копировать, вырезать и вставлять элементы управления, расположенные на поверхности формы.

Элементы управления, их свойства, события, методы

Командная кнопка является самым распространенным элементом управления и может использоваться для выполнения вычислений и других действий, вызова процедур и функций пользователя, открытия форм и т. д. Командным кнопкам можно присваивать различные свойства (табл. 5.7).

В свойстве **Caption** можно ставить символ **&** перед буквой, которая будет использоваться в сочетании с клавишей **Alt** для ускоренного доступа к кнопке. Также можно перейти к кнопке путем нажатия клавиши **Tab**, а затем **Enter**.

Таблица 5.7

Свойства командных кнопок

Свойство	Описание
BackColor	Цвет фона кнопки.
Caption	Текст, который выводится на кнопке.
Enabled	Значение False делает кнопку недоступной.
Font	Определяет тип и вид шрифта на кнопке.
ForeColor	Определяет цвет шрифта на кнопке.
(Name)	Имя объекта, для программы VBA.
Picture	Добавляет рисунок на кнопку.
PicturePosition	Определяет расположение текста и рисунка на кнопке.
Visible	Значение False делает кнопку невидимой.

Основным событием кнопки является **Click**. Для написания программного кода, который будет выполняться при нажатии командной кнопки, достаточно два раза щелкнуть на ней левой кнопкой мыши в режиме конструирования проекта.

Наиболее полезным методом командной кнопки является **SetFocus**, позволяющий вернуться к кнопке (передать ей фокус). Например, следующая команда позволяет вернуться к кнопке по умолчанию после ввода данных в текстовое поле:

cmdMyButtum.SetFocus

Текстовое поле применяется для ввода или вывода информации. Основные свойства текстового поля представлены в табл. 5.8.

Таблица 5.8

Свойства текстового поля

Свойство	Описание
Enabled	Значение False делает поле недоступным
Font	Определяет тип и вид шрифта в текстовом поле

Окончание табл. 5.8

Свойство	Описание
ForeColor	Определяет цвет шрифта в текстовом поле
(Name)	Имя объекта, для программы VBA
MaxLength	Определяет количество вводимых символов в текстовое поле
PasswordChar	Определяет символ, отображаемый при вводе в текстовое поле
Text	Определяет содержимое текстового поля

Например, для очистки содержимого текстового поля в ходе выполнения программы необходимо ввести в требуемом месте программного кода команду:

txtResult.Text=" "

Основным событием текстового поля является **Change**, происходящее при вводе или удалении символов. Например, команду **cmdMyButtum.SetFocus** можно поместить в процедуру события **Change** текстового поля.

Надпись применяется как самостоятельно для вывода справочной информации, так и в виде подсказок для текстового поля, списка или другого элемента. Главное ее отличие от текстового поля в том, что пользователь не может изменить текст надписи (хотя его можно изменить как свойство во время выполнения программы). Основные свойства надписи представлены в табл. 5.9.

Таблица 5.9

Свойства надписи

Свойство	Описание
Caption	Определяет текст, содержащийся в надписи
Font	Определяет тип и вид шрифта надписи

Окончание табл. 5.9

Свойство	Описание
ForeColor	Определяет цвет шрифта надписи
(Name)	Имя объекта, для программы VBA
Picture	Добавляет рисунок в надпись
PicturePosition	Определяет расположение текста и рисунка надписи

Список позволяет работать с перечнем из нескольких вариантов. Пользователь может просмотреть содержимое списка и выбрать один из вариантов для последующей обработки. Прямое редактирование содержимого списка невозможно. Если в списке помещаются не все строки, то автоматически добавляется вертикальная полоса прокрутки. Основные свойства списка представлены в табл. 5.10.

Таблица 5.10

Свойства списка

Свойство	Описание
(Name)	Имя объекта, для программы VBA
ListIndex	Возвращает номер текущей выделенной строки списка –1
Text	Содержимое текущей выделенной строки списка

Для списка чаще всего используются события **Click** и **DblClick** (двойной щелчок левой кнопкой мыши на одной из строк списка). Во втором случае пользователь одновременно выделяет строку и начинает ее обработку.

Работа со списком начинается с его заполнения методом **AddItem**, который может вызываться несколько раз подряд. Часто метод **AddItem** помещается в процедуру **UserForm_Initialize()**, чтобы список заполнялся при загрузке формы. Метод **RemoveItem** удаляет строки из списка. Метод **Clear** очищает сразу весь список. Следующий пример показывает, как работают списки, при этом предполагается, что в проекте создана форма с двумя списками

(List1 и List2). Двойной щелчок на любой строке одного списка перемещает ее в другой список. Строка включается в другой список до того, как она будет удалена из текущего.

```
Private Sub UserForm_Initialize()
    List1.AddItem "Стол"
    List1.AddItem "Стул"
    List1.AddItem "Диван"
    List1.AddItem "Кресло"
    List1.AddItem "Кровать"
End Sub
Private Sub List1_DblClick()
    List2.AddItem List1.Text
    List1.RemoveItem
    List1.ListIndex
End Sub
Private Sub List2_dblClick()
    List1.AddItem List2.Text
    List2.RemoveItem
    List2.ListIndex
End Sub
```

Переключатели позволяют выбрать один вариант из группы. Обычно они группируются в рамках, однако их можно располагать прямо на форме, если используется только одна группа переключателей. Основные свойства переключателей представлены в табл. 5.11.

Таблица 5.11

Свойства переключателя

Свойство	Описание
Caption	Задает текст, определяющий назначение переключателя
(Name)	Имя объекта, для программы VBA.
Value	Значение True указывает, что переключатель выбран

Наиболее важным является свойство **Value** со значением **True** (переключатель находится в установленном состоянии), которое в режиме конструирования задается только у одного переключателя в группе. В режиме выполнения это свойство чаще всего проверяется в процедуре события **Click** кнопки, нажатой после установки нужного переключателя, что позволяет проверить перед вызовом следующей процедуры некоторое условие. Однако определенные действия можно выполнять сразу же после выбора переключателя в процедуре его события **Click**.

Флажок частично схож с переключателем, но в отличие от него может использоваться как отдельный самостоятельный элемент. Даже объединенные в группу флажки работают независимо друг от друга. Основные свойства флажков такие же, как и у переключателя. Однако свойство **Value** может принимать три значения (флажок может находиться в установленном, снятом или неопределенном состоянии).

Наиболее часто используемым событием флажков является **Click**, в процедуре которого можно проверять состояние флажка по свойству **Value**. Следующий пример иллюстрирует работу флажков, при этом предполагается, что в проекте создана форма с двумя флажками (**ChkBold** и **ChkInalic**) и текстовым полем **TxtExam** (рис. 5.1).

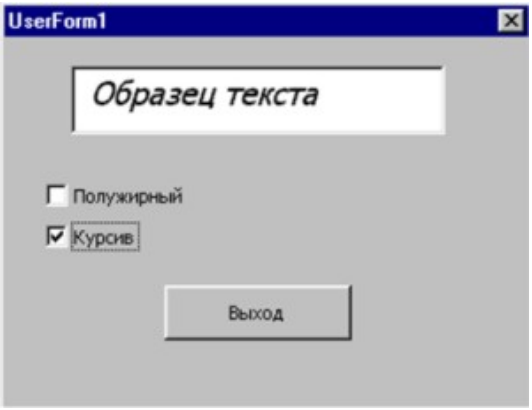

Рис. 5.1. Использование флажков

После ввода символов в текстовое поле, с помощью флажков можно придать набранному тексту полужирное или курсивное начертание. Именно свойства **FontBold** и **FontItalic** текстового поля устанавливают начертания текста.

```
Private Sub Chkbold_Click()
  If ChkBold.Value = True Then
    TxtExam.FontBold = True
  Else
    TxtExam.FontBold = False
  End If
End Sub
Private Sub ChkItalic_Click()
  If ChkBold.Value = True Then
    TxtExam.FontItalic = True
  Else
   TxtExam.FontItalic = False
  End If
End Sub
```

Рамка используется для группировки переключателей или флажков и помещается на форму раньше элементов, находящихся внутри нее. Переключатели находящиеся внутри рамки, работают как самостоятельная группа и не влияют на состояние переключателей в других рамках. Основным свойством рамки является **Caption**, которое задает текст, определяющий назначение элементов в рамке.

Рисунок используется для простейшего вывода изображения на форме. Он может отображать растровые файлы (.BMP), значки (.ICO), метафайлы (.WMF), а также файлы в формате JPEG (.JPG) и GIF (.GIF). Основные свойства рисунка представлены в табл. 5.12.

Таблица 5.12

Свойства рисунка

Свойство	Описание
Autosize	Значение True подгоняет размер элемента под размер содержимого

Окончание табл. 5.12

Свойство	Описание
(Name)	Имя объекта, для программы VBA
Picture	Задает файл для рисунка

События и методы рамок и рисунков практически не используются.

Контрольные вопросы

1. Дайте определение формы.
2. Перечислите основные свойства формы.
3. Как можно изменять свойства?
4. Перечислите основные события формы.
5. Перечислите основные методы формы.
6. Перечислите основные инструкции формы.
7. Как добавить форму в проект?
8. Как выбрать уже созданный в форме элемент?
9. Перечислите основные элементы управления?
10. Что отражают первые три символа имени элемента?
11. Перечислите общие свойства элементов управления.
12. Как связать элемент управления с кодом?
13. Дать пояснения следующим элементам программы:

1) Dim a As Integer;
2) CInt;
3) $c = a + b$;
4) MsgBox "результат смотри в TextBox3";
5) TextBox3.Text = c;
6) TextBox1.Text = "", TextBox2.Text = "", TextBox3.Text = "";
7) TextBox1.SetFocus;
8) CheckBox1.Value = False.

Примеры

Пример 1

1. Расположить на форме следующие элементы: **Label1**; **TextBox1**; **CommandButton1**.

2. Активизировать **Label1**, в окне свойств найти свойство **Caption** и изменить **Label1** на название факультета.

3. Те же действия произвести с **CommandButton1**, изменяя **Caption** на «**Ок**».

4. Активизировать **TextBox1** и изменить свойство **Text**, набрав номер своей группы.

5. С помощью элемента **Выбор объекта** выделить все элементы на форме. В окне свойств хорошо видно какие свойства одинаковы для всех элементов. Изменяя свойства **Font** (шрифт: полужирный, курсив, размер – 15 пт.), **BackColor** (светлая тень для кнопки), **Visible** (**False**), проследить все изменения элементов управления на форме.

6. Осуществить запуск программы**.**

7. Вернуться в режим конструктора VBA для этого нажать крестик на форме.

8. Выделить опять все объекты и поменять только свойство **Visible** (**True**) и снова произвести запуск программы.

9. Самостоятельно изменить другие свойства элементов данной формы и пронаблюдать их изменения.

Пример 2

Создать программу для решения задачи $c = a + b$.

Порядок выполнения работы:

1. Запустить редактор VBA.

2. Выполнить команду **Вставка–User form**

3. Поместить на форму элементы, требуемые для решения задачи, с панели элементов, и расположить их нужным образом. Пример представлен на рис. 5.2.

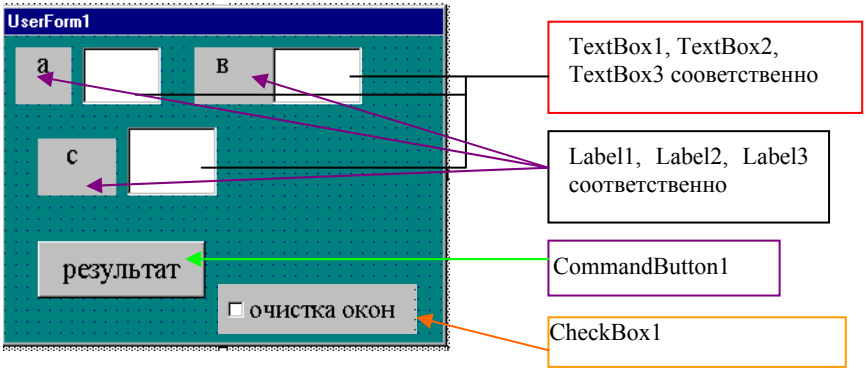

Рис. 5.2. Пример формы

4. Изменить свойства объектов на форме с помощью окна свойств, используя данные табл. 5.13.

Таблица 5.13

Свойства объектов формы

Свойство	Значение
Label1.Caption	A
Label2.Caption	B
Label3.Caption	C
CommandButton1.Caption	Результат
CheckBox1.Caption	Очистка окон
Свойство BackColor (для всех объектов)	По своему вкусу выбрать цвет из палитры цветов
Свойство Font (для Label1, Label2,Label3	В диалоговом окне «Шрифт», которое появится после щелчка по кнопке «…», расположенной напротив свойства Font в окне свойств, выбрать размер 16 пт.

5. Написать программный код. Для этого рекомендуется выполнить двойной щелчок по кнопке **Результат** и перейти в окно программы, в котором набрать текст процедуры обработки события **Click**() для кнопки и для флажка (**CheckBox1**):

```
Private Sub CheckBox1_Click()
  TextBox1.Text = ""
  TextBox2.Text = ""
  TextBox3.Text = ""
  TextBox3.Visible = False
  TextBox1.SetFocus
  CheckBox1.Value = False
End Sub
```

```
Private Sub CommandButton1_Click()
    Dim a As Integer
    Dim b As Integer
    Dim c As Integer
    a = CInt(TextBox1.Text)
    b = CInt(TextBox2.Text)
    c = a + b
    MsgBox "результат смотри в
                TextBox3"
    TextBox3.Visible = True
    TextBox3.Text = c
End Sub
```

Задание

Создать программу для решения задачи (по вариантам). Номер варианта выдает преподаватель:

- ввод и вывод данных осуществить аналогично примеру 2 данной лабораторной работы;
- количество элементов **TextBox** и **Label** определить по количеству переменных в выражениях;
- для вариантов, указанных преподавателем, поместить на форму 2 кнопки **CommandButton** для вычисления значения выражений 1 и 2 (см. задания, выданные преподавателем).

Лабораторная работа № 6

ОПЕРАТОРЫ ЦИКЛА

Цель работы: приобрести навыки решения задач на основе циклических структур.

Теоретические сведения

Цикл с предусловием For … Next

Для реализации циклического вычислительного процесса, т. е. многократного выполнения одной или нескольких операций, служит оператор цикла **For…Next**, который имеет следующий синтаксис:

For СЧЕТЧИК=НАЧ_ЗНАЧЕНИЕ To КОН_ЗНАЧЕНИЕ Step ШАГ
БЛОК_ОПЕРАТОРОВ
[Exit For]
БЛОК_ОПЕРАТОРОВ
Next СЧЕТЧИК

Например (рис. 6.1.):

```
S = 0
P = 1
For i = 1 to 5
    S = S + 2
    P = P * S
Next i
```

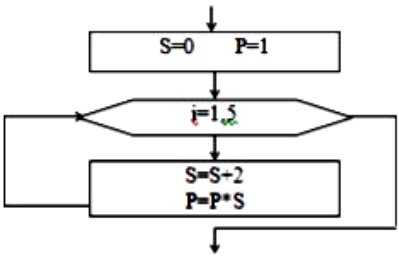

Рис. 6.1. Цикл с предусловием For … Next

Цикл **For … Next** перебирает значения переменной **СЧЕТЧИК**, которая является параметром цикла, от начального до конечного значения с указанным шагом изменения. При этом обеспечивается

выполнение блока операторов тела цикла при каждом новом значении счетчика. Если **Step ШАГ** в конструкции отсутствует, то по умолчанию считается, что шаг равен 1. По оператору **Exit For** можно выйти из оператора цикла до того, как **СЧЕТЧИК** достигнет последнего значения.

Примечание. Не рекомендуется принудительно изменять значения параметра цикла, его начального и конечного значения в теле цикла **For … Next**.

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

For Each Элемент In Группа
    БЛОК_ОПЕРАТОРОВ
    [Exit For]
    БЛОК_ОПЕРАТОРОВ
Next Элемент

В VBA для организации циклов с неизвестным заранее числом повторений используются и другие операторы цикла:

- циклы с предусловием – **Do While … Loop**, **Do Until … Loop**;
- циклы с постусловием – **Do … Loop While**, **Do … Loop Until**.

Циклы с предусловием – Do While … Loop, Do Until … Loop

Цикл предусловием **Do While … Loop** имеет следующий синтаксис:

Do While УСЛОВИЕ
    БЛОК_ОПЕРАТОРОВ
    [Exit Do]
    БЛОК_ОПЕРАТОРОВ
Loop

Например (см. рис. 6.2):

```
k = 0
f = 1
Do While f < = 15
    k = k + 1
    f = f + 2
Loop
```

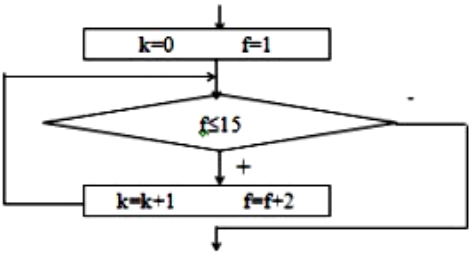

Рис. 6.2. Цикл с предусловием
Do While … Loop

Цикл с предусловием **Do Until … Loop**

Do Until УСЛОВИЕ
    БЛОК_ОПЕРАТОРОВ
    [Exit Do]
    БЛОК_ОПЕРАТОРОВ
Loop

Цикл с постусловием Do … Loop While, Do … Loop Until

Цикл с постусловием **Do … Loop While** имеет следующий синтаксис:

Do
    БЛОК_ОПЕРАТОРОВ
    [Exit Do]
    БЛОК_ОПЕРАТОРОВ
Loop While УСЛОВИЕ

Цикл с постусловием **Do … Loop Until**

Do
    БЛОК_ОПЕРАТОРОВ
    [Exit For]
    БЛОК_ОПЕРАТОРОВ
Loop Until УСЛОВИЕ

Например (см. рис. 6.3):

D = 1
S = 0
Do
    S = S + D
    D = D + 1
Loop Until d > = 10

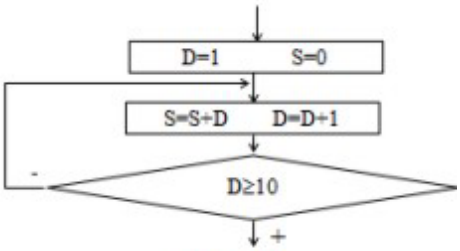

Рис. 6.3. Цикл с предусловием
Do … Loop Until

Оператор **Do While … Loop** обеспечивает многократное повторение блока операторов до тех пор, пока **УСЛОВИЕ** соблюдается, а оператор **Do Until … Loop** пока **УСЛОВИЕ** не соблюдается.

Операторы **Do … Loop While, Do … Loop Until** отличаются от перечисленных выше операторов тем, что сначала блок операторов выполняется по крайней мере один раз, а потом проверяется **УСЛОВИЕ**. Для избежания зацикливания в теле цикла должен быть *хотя бы один оператор*, который изменяет значения переменных, стоящих в **УСЛОВИИ**.

Оператор **Exit Do** обеспечивает досрочный выход из оператора цикла.

Контрольные вопросы

1. Что такое цикл?
2. Приведите синтаксис оператора цикла For … Next.
3. Как работает оператор цикла For … Next?
4. Напишите синтаксис оператора цикла For … Each … Next. В каком случае используется этот оператор цикла?
5. Перечислите циклы с предусловием.
6. Перечислите циклы с постусловием.
7. Каков синтаксис оператора цикла Do While … Loop.
8. Каков синтаксис оператора цикла Do Until … Loop.
9. Каков синтаксис оператора цикла Do … Loop While.
10. Каков синтаксис оператора цикла Do … Loop Until.

11. Как работают операторы цикла Do While … Loop и Do Until … Loop.
12. Как работают операторы цикла Do … Loop While и Do … Loop Until.
13. Как избежать зацикливания?
14. Что обеспечивает оператор Exit Do?

Примеры

Пример 1

Найти сумму десяти случайных чисел. Написать программу, воспользовавшись циклами различных видов.

```
Sub Сумма()
   Dim sum1 As Integer, sum2 As Integer, i As Integer
   Randomize
   'решение задачи с помощью цикла с предусловием
   i = 10
   Do While i > 0
      'цикла выполняется, пока логическое условие истинно
      sum1 = sum1 + Int((10 * Rnd) + 1)
      i = i – 1
   Loop
   MsgBox "Сумма чисел=" & sum1
   'решение задачи с помощью цикла с постусловием
   i = 10
   Do
      'цикла выполненяется, пока логическое условие ложно
      sum2 = sum2 + Int((10 * Rnd) + 1)
      i = i – 1
   Loop Until i = 0
   MsgBox "Сумма чисел=" & sum2
End Sub
```

Особенностью интерпретатора VBA является то, что значения переменных числовых типов перед выполнением процедуры полагаются равными 0. Поэтому в программе отсутствуют команды присваивания вида: sum1 = 0 и sum2 = 0.

Пример 2

Найти максимальное из *n* введенных с клавиатуры чисел. Приведены два варианта решения задачи с использованием циклов разных видов.

```
Option Explicit
Sub Max_n_while()
   Dim n As Byte, k As Single, i As Byte, Max As Single
   n = Val(InputBox("Введите количество чисел"))
   i = 1
   Do While i <= n
      k = Val(InputBox("Введите число", "Ввод чисел"))
      If i = 1 Then Max = k
      If k > Max Then Max = k
      i = i + 1
   Loop
   MsgBox "Наибольшее из чисел " & Max
End Sub
```

```
Option Explicit
Sub Max_n_until()
   Dim n As Byte, k As Single, i As Byte, Max As Single
   n = Val(InputBox("Введите количество чисел"))
   i = 1
   Do Until i > n
      k = Val(InputBox("Введите число", "Ввод чисел"))
      If i = 1 Then Max = k
      If k > Max Then Max = k
      i = i + 1
   Loop
   MsgBox "Наибольшее из чисел " & Max
End Sub
```

Если из текста программ удалить строку: **If i = 1 Then Max = k**, то программа будет работать корректно только в случае, когда хотя бы одно вводимое число неотрицательно. Это объясняется тем, что начальное значение переменной **Max** считается равным 0.

Пример 3

Найти сумму *n* первых членов ряда 1, 1/2, 1/3, ..., 1/*n*

```
Option Explicit
Sub Summ_n()
   Dim n As Byte, i As Byte, sum As Single
   n = Val(InputBox("Введите количество членов ряда"))
   For i =1 To n
      sum = sum + 1 / i
   Next
   MsgBox "Сумма " & sum
End Sub
```

Пример 4

Найти сумму всех четных чисел в первой десятке:

```
Option Explicit
Sub Summa ()
  Dim j As Integer, sum As Integer
  For j = 2 To 10 Step 2
     sum = sum + j
  Next
  MsgBox "Сумма равна " & sum
End Sub
```

Пример 5

Написать программу, осуществляющий вывод на экран введенного числа после его проверки. Ввод должен завершиться, когда вводимое значение окажется положительным числом.

```
Sub Пример()
   Dim d As Integer
   Do
      d = Val(InputBox("Введите положительное число", "Ввод числа"))
```

```
    Loop Until d > 0 And (IsNumeric(d))
    MsgBox "Введеное число  " & d
End Sub
```

Задания

Задание 1

Создать программу для решения задачи (по вариантам). Номер варианта выдает преподаватель.

Задание 2

Составить программы вычисления значений функции $F(x)$ на отрезке $[a; b]$ с шагом h (по вариантам, номер варианта выдает преподаватель). Результат представить в виде таблицы, первый столбец которой – значения аргумента, второй – соответствующие значения функции

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Михеев, Р. Н. VBA и программирование в MS Office для пользователей / Р. Н. Михеев – СПб. : БХВ, 2006. – 372 с.

2. Петров, В. Н. Информационные системы / В. Н. Петров, Ю. С. Избачков, – СПб. : Питер, 2008. – 656 с.

3. Эйткен, П. Разработка приложений на VBA в среде Office XP / П. Эйткен – М. : Издательский дом «Вильяме», 2003. – 496 с.

4. Клименко, Б. Microsoft Word : комфортная работа с помощью макросов / Б. Клименко, М. Розенберг. – СПб. : БХВ, 2006.

5. Уокенбах Дж. Excel 2010 : профессиональное программирование на VBA = Excel 2010 Power Programming with VBA / Дж. Уокенбах. – М. : Диалектика, 2011. – 944 с.

Содержание

Учебное издание

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ VBA

Лабораторный практикум
по дисциплине «Информационные системы и сети»
для студентов специальности
1-02 06 02 «Технология. Дополнительная специальность»

Составители:
АСТАПЧИК Наталья Ивановна
ЗУЁНОК Анна Юльяновна

Редактор *Т. А. Зезюльчик*
Компьютерная верстка *А. Г. Занкевич*

Подписано в печать 17.10.2013. Формат 60×84 $^{1}/_{16}$. Бумага офсетная. Ризография.
Усл. печ. л. 4,77. Уч.-изд. л. 3,73. Тираж 100. Заказ 1387.

Издатель и полиграфическое исполнение: Белорусский национальный технический университет. ЛИ № 02330/0494349 от 16.03.2009. Пр. Независимости, 65. 220013, г. Минск.