ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Кафедра «Математика и информатика»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению контрольной работы
по дисциплине

«Информационные технологии»

Автор

Акишин Б.А.

Ростов-на-Дону, 2015

Аннотация

Методические указания к выполнению контрольной работы
для студентов заочной сокращенной формы обучения
по дисциплине Информационные технологии

Направление подготовки 080100 –«Экономика»

Автор

Доцент, к.т. наук Акишин Б.А.

Оглавление

Результаты выполнения всех заданий представить в печатном виде, а также виде файлов MS Office 2007 (или 2003) на CD (вариант задания выбирается по последней цифре в номере зачетной книжки):

Задание 1

Представить реферат (5-10 стр.) по теме:

1. Система управления базами данных (СУБД) Oracle. Основные характеристики и возможности.
2. Реализация языка запросов SQL. Основные операторы.
3. Система управления базами данных (СУБД) MS SQL Server. Основные характеристики и возможности.
4. Принципы работы беспроводной сети Wi-Fi. Примеры использования в бизнесе.
5. Технологии экспертных систем. Примеры и перспективы использования в бизнесе.
6. Основные понятия «облачных технологий». Примеры и перспективы использования в бизнесе.
7. Система управления базами данных (СУБД) My SQL. Основные характеристики и возможности.
8. Основные отличия Access 2010 от Access 2007. Новые возможности.
9. Технология распределенных баз данных. Преимущества и возможности.
10. Прикладные приложения операционной системы Android. Средства разработки приложений.

Задание 2

Разработать макрос на VBA:

1. Массив целых чисел ***[q[i, j] , i =1,4; j=1,3]*** импортируется из некоторого заполненного диапазона ячеек Excel (функция *Cells*). Найти сумму произведений элементов 1-ой и 4-ой строки. Результат вывести в окно MsgBox.

2. Создать пользовательскую форму для ввода массива целых чисел ***[q [i, j] , i =1,2; j=1,3]*** и вывода результатов расчета суммы и количества элементов, удовлетворяющих условию: ***10 < q [i, j] <18***

3. Массив целых чисел ***[q [i, j] , i =1,4; j=1,4]*** подготовлен в некотором диапазоне ячеек Excel. Требуется обнулить в массиве элементы побочной диагонали и вывести результат на тот же лист Excel (для обмена данными используется функция *Cells*)

4. Заполнить массив ***[q [i, j] , i =1,3; j=1,3]*** целыми случайными числами, отобразить их в пользовательской форме, вычислить максимальный элемент массива и вывести результат (в той же форме).

5. Массив целых чисел ***[q [i, j] , i =1,4; j=1,4]*** импортируется из некоторого заполненного диапазона ячеек Excel (функция *Cells*). Вычислить сумму элементов главной диагонали. Результат вывести в окно MsgBox

6. Заполнить массив ***[q [i, j] , i =1,3; j=1,3]*** целыми случайными числами, отобразить их в пользовательской форме, вычислить минимальные элементы каждого столбца и вывести результат (в той же форме).

7. Массив целых чисел ***[q [i, j] , i =1,4; j=1,4]*** подготовлен в некотором диапазоне ячеек Excel. Требуется поменять местами вторую и третью строки и вывести результат на тот же лист Excel (для обмена данными используется функция *Cells*).

8. Осуществить циклический ввод элементов массива целых чисел ***[q [i, j] , i =1,4; j=1,2]*** с экрана (использовать функцию *InputBox*), найти наибольшие элементы каждого столбца и поставить их первыми. Исходный и измененный массивы отобразить на листе Excel (использовать функцию *Cells*)

9. Массив целых чисел ***[q [i, j] , i =1,5; j=1,4]*** подготовлен в некотором диапазоне ячеек Excel. Требуется вывести на тот же лист Excel массив без отрицательных элементов (для обмена данными используется функция *Cells*).

10. Осуществить циклический ввод элементов массива целых чисел ***[q [i, j] , i =1,3; j=1,2]*** с экрана (использовать функцию *InputBox*), найти наименьшие элементы каждой строки и поставить их первыми. Исходный и измененный массивы отобразить на листе Excel (использовать функцию *Cells*)

Задание 3

a) **Создать базу данных MS Access в соответствии с вашим вариантом, которая должна содержать не менее двух связанных по ключевому полю таблиц.**
b) **Создать форму для ввода, корректировки и просмотра данных основной таблицы.**
c) **Заполнить обе таблицы данными (не менее 20 записей).**
d) **Создать не менее трех разнообразных запросов (с условиями отбора, с параметром, с вычисляемым полем и т.п.)**
e) **Создать отчет по основной таблице или запросу.**
f) **В контрольной работе описать структуру базы данных, последовательность создания объектов базы данных, распечатать данные, результаты выполнения запросов и отчет.**

1. База данных "**Список студентов**", включающая поля "Номер студенческого билета", "Ф.И.О.", "Дата рождения", "Форма обучения", "Курс", "Факультет", "Домашний адрес".

2. База данных **"Данные о товарообороте",** содержащая поля "Период времени", "Название магазина", "Фактический товарооборот», "Товарные запасы", "Норматив товарных запасов".

3. База данных **«Итоги сессии»,** содержащая поля: «Номер зачетной книжки», «ФИО», «Курс», «Факультет», «Дисциплина», «Преподаватель», «Дата экзамена», «Оценка». Один из запросов должен включать отбор всех задолжников.

4. База данных **"Расчетно-платежные ведомости",** содержащая поля "Месяц", "Ф.И.О.", "Должность", "Разряд", "Начислено", "Удержано", "К выдаче".

5. База данных **"Недвижимость",** содержащая поля: "Район", "Улица", "Тип жилья", "Этаж", "Количество комнат", "Общая площадь", "Жилая площадь", "Кухня", "Стоимость", "Номер агента"

6. База данных **"Склад"**, содержащая поля "Дата поступления", "Поставщик", "Наименование товара", "Единица измерения" "Цена", "Количество", "Сумма".

7. База данных **"Платежное поручение",** содержащая поля "№", "Дата", "Получатель", "Банк получателя", "Адрес Банка", "Счет получателя", "Сумма".

8. База данных **"Остатки товаров в магазине",** содержащая поля "Дата", "Наименование товара", "Получено", "Продано", "Возвращено", "Остаток".

9. База данных **"Список сотрудников фирмы",** содержащая поля "Ф.И.О.", "Дата поступления на работу", "Должность", "Образование", "Год рождения", "Оклад".

10. База данных **"Библиотечный каталог",** содержащая поля "Раздел", "Код книги", "Название книги", "Автор", "Год издания", "Издательство", "Цена", «Количество экземпляров»

Вопросы для подготовки к зачету по информационным технологиям

1. Информация: уровни представления, свойства. Экономическая информация. Что такое "информационная технология", "информационная система"?
2. Обобщенная структура информационных систем (ИС). Базовые компоненты ИС. Организационная структура ИС.
3. Организация информации на машинных носителях. Электронный документооборот. Классификация информационных технологий, используемых при создании информационных систем в экономике.
4. Какие средства организационной, коммуникационной и вычислительной техники используются в информационных системах? Каковы состав и назначение системного и прикладного программного обеспечения, используемого в информационных системах?
5. Логическая и физическая организация данных. Реляционная модель организации данных. Этапы проектирования базы данных. Базы данных и системы управления базами данных – их основные функции и типовая организация.
6. Основные характеристики системы управления базами данных (СУБД) Microsoft Access. Структура данных файловой модели: поле, запись, файл. Открытие базы данных, создание файла базы данных. Компоненты MS Access. Различия таблиц Excel и Access.

7. MS Access: создание таблиц, основные типы и свойства полей, определение первичного и вторичного ключа. Режим Конструктора Ввод и корректировка данных непосредственно в таблице или через форму, корректировка структуры таблицы, использование данных типа «Поле объекта OLE».
8. MS Access: работа с формами. Структура и дизайн формы. Типы элементов формы. Создание формы и отдельных элементов формы. Управление последовательностью переходов.
9. Схема данных в Access. Создание межтабличных связей. Обеспечение целостности данных.
10. MS Access: поиск и замена информации, сортировка и фильтрация данных.
11. MS Access: запросы к базе данных. Их назначение и типы. Запросы SQL и QBE. Сходства и различия запросов и фильтров.
12. MS Access: создание QBE- запросов на выборку. Использование Конструктора запросов и Построителя выражений. Создание вычисляемых полей.
13. MS Access: создание QBE- запросов с параметрами. Использование шаблонов. Итоговые запросы.
14. MS Access: создание перекрестных запросов и запросов на обновление таблиц.
15. MS Access: создание и корректировка отчетов. Окно Конструктора отчетов. Создание элементов отчета.
16. MS Access: макросы. Сборка приложений. Связывание и внедрение объектов. Работа с внешними таблицами.
17. Что такое «сетевые технологии» и каковы преимущества их использования? Понятие о компьютерных сетях. Предназначение компьютерных сетей. Понятия глобальная и локальная сеть. Сетевая безопасность. Модем. Основная характеристика. Сетевая карта. Основная характеристика
18. В чем заключаются основные различия между локальными и крупномасштабными вычислительными сетями? Каналы связи и из характеристики. Сетевое оборудование и сетевые ОС.
19. Структура и топология локальных вычислительных сетей (ЛВС). Одноранговые ЛВС и ЛВС на основе сервера. Подключение к ЛВС. Программа «Сетевое окружение».

20. Как осуществляется передача данных по каналу связи, соединяющему два соседних узла сети? Какие существуют стандарты, правила и соглашения в области построения вычислительных сетей? Вопросы совместимости в сетях. Модель ISO/OSI. Безопасность в сетях.
21. Информационно- технологическая архитектура ИС. Сетевые технологии и системы распределенной обработки информации. В чем состоят основные особенности архитектуры «файл-сервер», «клиент-сервер», Intranet?
22. Как организована всемирная компьютерная сеть Интернет? Адресация в Интернет. Обеспечение совместимости работы в сети. Сетевые протоколы TCP/IP. Как осуществляется передача данных в Интернет?
23. Регистрация в Интернет. Основные службы Интернет. Электронная почта и ее адресация, протоколы, программная поддержка
24. Что такое гипертекстовые технологии, какое применение они нашли в Интернет? Служба WWW. Указатели ресурсов. Поиск информации в Интернет, основные поисковые системы.
25. Язык программирования высокого уровня Visual Basic. Основные операторы и функции.
26. Основные понятия объектно-ориентированного программирования: объекты, свойства, методы, семейства, инкапсуляция, наследование и полиформизм. Синтаксис.
27. Объекты VBA в MS Excel и их основные свойства. Справочная система VBA, работа с Object Browser.
28. События и процедуры обработки событий. Элементы управления как объекты VBA и их основные свойства.
29. Пользовательские формы как объекты VBA. Их основные свойства и порядок создания.

Студент должен иметь практические навыки по информатике:

1. **MS Access:** создание таблиц и межтабличных связей, запросы на выборку различных типов, создание простейших форм и отчетов.

2. **VBA в MS Excel:** работа с макрорекордером и Object Browser, анализ и корректировка созданных кодов, создание макросов по обработке диапазона ячеек, создание пользовательских функций, создание элементов управления на листах Excel, создание пользовательских форм

Литература:

1. Соболь Б.В. Информатика
2. Максимов Н.В. и др. Современные информационные технологии: учебник. М.: Форум, 2008
3. Гарнаев А.Ю. Excel, VBA, Internet в экономике и финансах
4. Акишин Б.А. Информатика. Программирование на VBA. Учебно-методическое пособие для студентов
5. Кошелев В.Е. Access 2007. Эффективное использование.

ПРОГРАММИРОВАНИЕ НА VBA

1. Введение

В настоящее время Microsoft Office является наиболее используемым и полезным программным продуктом. Набор его приложений, включающий текстовый редактор Word, электронные таблицы Excel, систему управления базами данных Access, редактор Web-страниц Front Pages, пакет подготовки презентаций PowerPoint, электронный секретарь Outlook и другие, предназначен для решения очень широкого круга задач - от создания простых документов и отчетов до полной автоматизации документооборота организации с использованием систем управления базами данных и создания сайтов в сети Интернет.

При этом каждый пользователь имеет возможность настроить любое приложение MS Office для оптимального выполнения своих специфических задач, автоматизировать рутинную повторяющуюся работу, а также создать собственные нестандартные процедуры и функции и разработать довольно сложные программы, работающие в интегрированной среде MS Office.

Эти возможности реализуются, как правило, путем создания программ на встроенном объектно - ориентированном языке Visual Basic for Applications (VBA). Важнейшим достоинством VBA является возможность объединять любые приложения Office для решения, практически, любых задач по обработке информации. VBA позволяет работать с MS Office, как с некоторым конструктором: в распоряжении разработчика VBA-приложения большое количество объектов и коллекций.

Основной единицей программного кода на языке VBA является *макрос*, который представляет собой надлежащим образом оформленную последовательность команд, способных выполнить определенные действия или произвести определенные расчеты. Макросы могут писаться вручную в Редакторе VBA, как обычные процедуры на языке программирования Visual Basic (*процедуры пользователя*), или в автоматическом режиме с помощью макрорекордера (*процедуры макросов*). Различают также *процедуры обработки событий* (подробнее в разделе 9).

Каждый макрос VBA начинается с ключевого слова Sub (от слова Subroutine – процедура), за которым следует имя макроса и пустые круглые скобки. Первую строку кода, содержащую эти данные, называют *строкой объявления* (declaration) макроса. Заканчивается макрос строкой End Sub.

Макросы VBA сохраняют- ся в специальной части основ-

ного файла данных (документа Word, книги Excel и др.), называемой *модули* (Modules). Каждый модуль может содержать исходный код (source code) нескольких макросов, а документ Office может содержать несколько модулей, которые объединены общим названием *проект* (Project). В проекте VBA автоматически создает модули для каждого рабочего листа, для всей рабочей книги, а также для каждой *пользовательской формы* (User Form). По назначению модули бывают двух типов: *модули объектов и стандартные*. В окне Project отображается реестр модулей и форм.

В дальнейшем, не ограничивая общности, будем излагать основы VBA- программирования в среде MS Excel.

Отметим, что операционная система Windows воспринимает макросы как элементы управления ActiveX и при загрузке приложений MS Office предупреждает, что макросы могут быть вирусоопасными. Чтобы упростить работу с создаваемыми макросами, в учебных целях рекомендуется установить средний уровень безопасности макросов через меню Excel: **Сервис | Параметры | Безопасность | Безопасность макросов | Средняя.**

2. Основные операторы и функции VBA

Каждый оператор VBA начинается с новой строки. Если возникла необходимость разместить на одной строке несколько операторов (например, в целях лучшей обозримости кода макроса), то они отделяются друг от друга в этой строке двоеточием.

Перенос длинной строки можно осуществить, добавив в конце строки символы (пробел)+(знак подчеркивания _).

2.1. Оператор присваивания

<Переменная> = <Выражение>

Заданное или вычисляемое в правой части оператора выражение присваивается переменной левой части, «стирая» ее предыдущее значение. Каждое хранимое значение имеет в компьютере физический адрес памяти, которая в данный момент его содержит, и имя переменной, которая им обладает (ссылается на этот адрес).

2.2. Описание типов переменных

Dim <Имя переменной> As <Тип переменной>

Часто используемые типы данных: Integer – целый, Single – вещественный, String - символьный, Boolean – логический.
В VBA имеется универсальный тип данных Variant, который подразумевается по умолчанию.

2.3. Условный оператор

If <Условие> Then <Действия1> Else <Действия2> End If

Если Условие истинно, то выполняются Действия1, иначе (если Условие ложно) выполняются Действия2.

Замечание. Условные операторы, как и операторы циклов, могут быть вложенными. Вместо вложенных условных операторов можно использовать оператор множественного ветвления Select Case …

2.4. Цикл с параметром

```
For <Переменная-счетчик> =<Нач. знач.> To  <Кон. знач.> Step <Приращ.>
        <Тело цикла>
Next
```

Эти циклы используются, когда число повторений известно или может быть вычислено заранее. По умолчанию Приращение=1.

2.5. Цикл с предусловием

```
While  <Условие>                          DoWhile <Условие>
      <Тело цикла>          или
<Тело цикла>
Wend                                      Loop
```

Тело данного цикла выполняется, пока Условие истинно.

2.6. Цикл с постусловием

```
Do
<Тело цикла>
Loop Until  <Условие>
```

Данный цикл работает до выполнения Условия, пока оно не верно, т.е. здесь истинность Условия означает выход из цик-

ла. Оператор используется в задачах, когда необходимо, чтобы тело цикла хотя бы один раз отработало.

2.7. Досрочный выход из цикла

Exit For – в циклах, начинающихся с For,
Exit Do – в циклах, начинающихся с Do.

2.8. Ввод с экрана

a = InputBox("приглашение к вводу")

2.9. Вывод в специальное окно экрана

MsgBox(x)

При склеивании строк используется знак & (амперсанд).

2.10. Обмен данными с ячейками Excel

Cells(i, j)

где i – номер строки Excel, j – номер столбца

В данном разделе приведены только основные операторы VBA и их синтаксис. Более подробную информацию можно получить в [1, 2] или по справке F1 в Редакторе VBA.

3. Возможности редактора VBA

Редактор запускается из меню MS Excel **Сервис | Макрос | Редактор Visual Basic** или при нажатии клавиш **Alt+F11** (к сожалению, в большинстве русифицированных версий MS Office Редактор Visual Basic не русифицирован). Текст макроса вводится и отображается в окне **Code**. Чтобы вывести на экран окно **Code** (если его нет), нужно выбрать **View | Code** или нажать клавишу **F7**.

Новый стандартный модуль для записи макросов можно добавить через меню **Insert | Module.**

Редактор VBA содержит ряд возможностей, помогающих в написании процедур:

a. после ввода первой строки объявления автоматически добавляется последняя строка End Sub;
b. при вводе имени встроенной процедуры или функции появляется подсказка Auto Quick Info – всплывающее окно с информацией об аргументах этой процедуры или функции, причем аргумент, значение которого вы должны ввести, вы-

деляется полужирным шрифтом;

c. при нажатии клавиши F1 на выделенном ключевом слове или имени оператора вызывается соответствующая справочная информация, включая примеры использования;

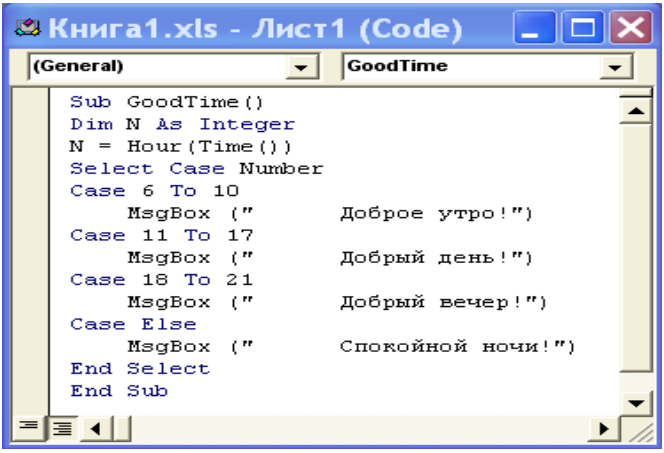
d. редактор автоматически выделяет синим цветом все ключевые слова операторов VBA, зеленым – комментарии, а красным – синтаксические ошибки.

Возврат из Редактора VBA в Excel осуществляется либо через меню **File | Close and Return to Microsoft Excel**, либо при помощи комбинации клавиш **Alt+Q**.

4. Создание простейшего макроса

Создадим макрос, при выполнении которого, в зависимости от времени суток, на экран выводится приветствие «Доброе утро!» - с 6-00 до 10-00, «Добрый день!» - с 11-00 до 17-00, «Добрый вечер!» - с 18-00 до 21-00, «Спокойной ночи!» - в остальное время.

Для этого запустим MS Excel, выберем **Сервис | Макрос | Редактор Visual Basic** и в окне **Code** введем текст (исходный код) макроса, который назван нами как **GoodTime** (впрочем, имена процедур, функций и переменных могут быть и русские, как, например, в разделе 7):



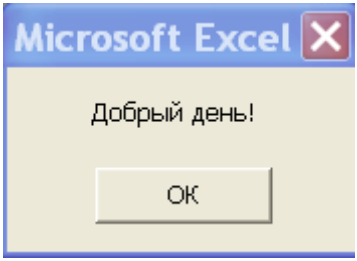
Сделаем некоторые пояснения к коду:

- функция **Time()** возвращает системное время компьютера, а функция **Hour()** – целое число, содержащее часы как

часть времени. Это значение и присваивается переменной **N**, определенной оператором **Dim** как переменной целого типа (Integer);

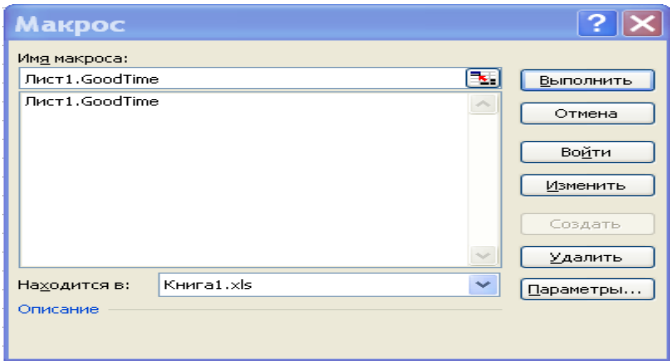
- выбор соответствующего сообщения осуществляется с помощью оператора условного перехода **Select Case**, который используются с четырьмя операторами **Case**, проверяющими условие на текущее значение переменной **N** (напомним, что подробный синтаксис оператора и примеры его использования можно получить по справке, нажав клавишу F1);
- вывод строки сообщения в диалоговое окно на экране осуществляется процедурой **MsgBox(...)**;
- код макроса записан в модуле **Лист1**.

Контрольный запуск макроса можно выполнить, не выходя из редактора – выберем **Run | Run Sub/UserForm**. Результатом работы макроса будет выдача на экран диалогового окна (запуск был осуществлен в 17:07)



Непосредственно из Excel макрос можно запускать из **Сервис | Макрос | Макросы.** В появившемся диалоговом окне выбираем макрос (его имя будет **Лист1.GoodTime**, где Лист1 – имя модуля) и нажимаем кнопку Выполнить.

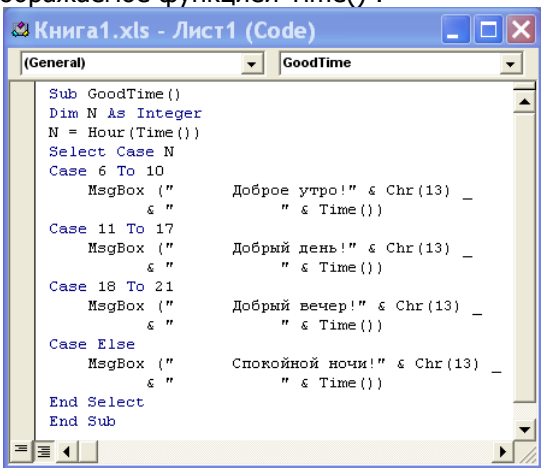
Замечание: Диалоговое окно Макрос перечисляет все макросы, сохраненные в любых рабочих книгах, открытых в данный момент.



5. Редактирование макроса

Требуется изменить код ранее созданного макроса **GoodTime**, таким образом, что наряду с приветствием в окне сообщения выводилось также системное время компьютера. Для этого загрузим Excel- файл Книга1.xls, выберем **Сервис | Макрос | Макросы** , в диалоговом окне выберем макрос **Лист1.GoodTime** и нажмем кнопку Изменить.

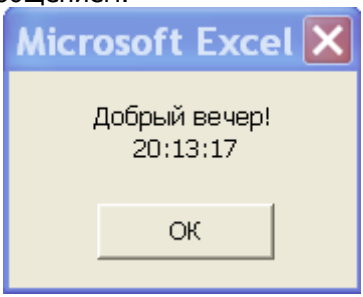
Расширим выводимые строки в вызовах процедуры вывода MsgBox, используя оператор **&** конкатенации («склеивания») строк, символ возврата каретки Chr(13) и системное время компьютера, отображаемое функцией Time() :



Замечание: как отмечалось в разделе 2, перенос текста оператора на другую строку осуществляется вводом символов (пробел)+(знак подчеркивания _).

Результатом работы из- мененного макроса будет выда-

ча на экран окна с сообщением:



5.1. Упражнение

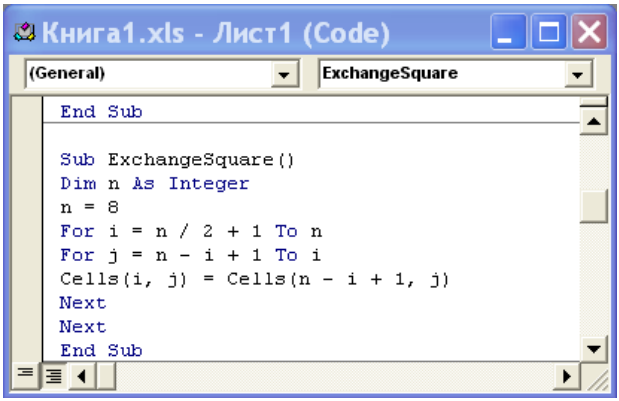
Создайте макрос, при выполнении которого компьютер выдает соответствующее сообщение в ответ на один из запросов пользователя с экрана: «Время ?», «Дата ?» или «День недели ?». *Указание*: используйте функции Time, Date, Weekday, WeekdayName.

6. Пример макроса по обработке диапазона ячеек

На листе Excel в диапазоне A1:H8 ячейки верхнего треугольника заполнен некоторым символом так, как показано на рисунке. Требуется зеркально отобразить этот треугольник на нижние строки диапазона.

	A	B	C	D	E	F	G	H
1	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼
2		▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	
3			▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼	▼▼▼▼▼▼▼		
4				▼▼▼▼▼▼▼	▼▼▼▼▼▼▼			
5								
6								
7								
8								

Текст подготовленного макроса, названного нами ExchangeSquare(), представлен ниже. Он записан в том же модуле **Лист1**, что и наш первый макрос GoodTime(). Разделительную черту между макросами редактор ставит сам – ее видно вверху окна.



В VBA ячейки активного листа Excel имеют имена Cells(i, j), где i – номер строки, j – номер столбца. С помощью вложенных циклов For и оператора присваивания осуществляется переписывания 4-ой строки в 5-ую, 3-ей в 6-ую и т.д. После выполнения макроса диапазон ячеек будет иметь вид:

	A	B	C	D	E	F	G	H
1	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥
2		♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	
3			♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥		
4				♥♥♥♥♥♥♥	♥♥♥♥♥♥♥			
5				♥♥♥♥♥♥♥	♥♥♥♥♥♥♥			
6			♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥		
7		♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	
8	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥	♥♥♥♥♥♥♥

Заметим, что при отображении ячеек при помощи макроса Exchange Square() фон ячеек нижних строк не изменился. Вернемся к этому вопросу позже (раздел 10).

6.1. Упражнение

На листе Excel подготовьте матрицу вещественных чисел размером 7x7. Создайте макрос, который бы менял местами два любых столбца матрицы.

7. Создание пользовательских функций Excel

Функция – это подпрограмма, которая выполняет действия в пределах своего блока и возвращает программе единственное значение. Функции имеют следующий синтаксис:

Function ИмяФункции (Список параметров) As ТипДан-

ных

« тело функции»
ИмяФункции = Возвращаемое значение
End Function

При вызове функции указывается ее имя и передаваемые ей параметры (список параметров может и отсутствовать). Возможен досрочный выход из функции по инструкции Exit Function. Готовая функция, подготовленная в стандартном модуле (например, Module1), автоматически попадает в категорию **Определенные пользователем** окна Мастера функций Excel.

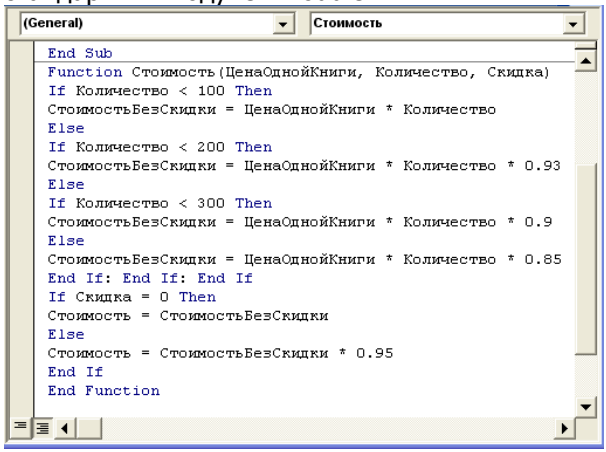
Пример.

Создадим функцию пользователя для расчета стоимости партии книг при следующих условиях продажи:

- если продается от 100 до 200 экземпляров книги, то скидка от ее отпускной цены составляет 7%, если продается 201-300 экземпляров, то скидка составляет 10%, а если свыше 300 экземпляров, то 15%,
- для постоянных клиентов предусмотрена дополнительная скидка в размере 5%.

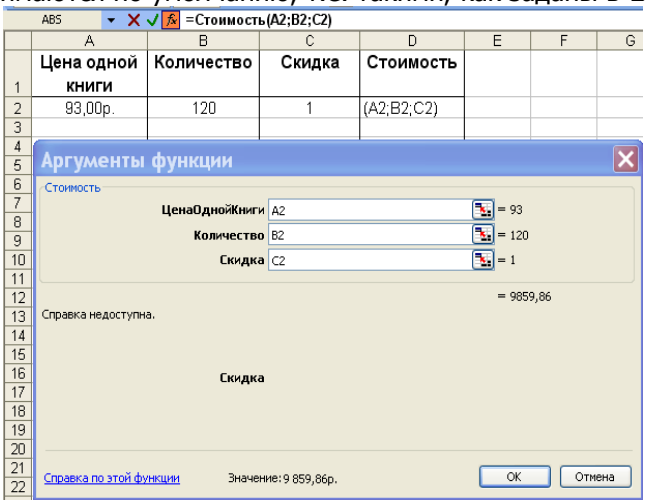
Ниже приводится программный код VBA этой функции (названной нами **Стоимость**), записанный в Module1.

Замечание: для того, чтобы любая пользовательская функция стала встроенной, необходимо, чтобы ее код был записан в одном из стандартных модулей ModuleN.



В коде использовались русские имена всех переменных, что в VBA допустимо. Поскольку типы переменных не определены, то

они понимаются по умолчанию, т.е. такими, как заданы в Excel.



Заметим еще, что если параметр Скидка = 0, то дополнительной скидки не предполагается, для постоянных клиентов параметр должен быть отличным от нуля, например, Скидка = 1.

Теперь зайдем в Excel, подготовим исходные данные о продаже книг и обратимся к нашей функции **Стоимость** через Мастер функций (напомним, что функция находится в категориях «**Определенные пользователем**» и «**Полный алфавитный перечень**»).

Получаем результат:

D2 =Стоимость(A2;B2;C2)

	A	B	C	D
1	**Цена одной книги**	**Количество**	**Скидка**	**Стоимость**
2	93,00р.	120	1	9 859,86р.
3				

7.1. Упражнение

Разработайте функцию, позволяющую рассчитывать комиссионные. Процент комиссионных зависит от объема проданного товара и начисляется по следующему правилу:

Объем продаж за неделю, руб.	Комиссионные, %
От 0 до 9 999	8
От 10 000 до 19 999	10
От 20 000 до 39 999	12
Более 40 000	14

8. Автоматическая запись макроса

Если, например, при работе с MS Excel у вас возникла необходимость несколько раз выполнить одну и ту же последовательность действий, то вы можете автоматически, с помощью макрорекордера (Macro Recorder), записать эту последовательность как макрос. Макрос записывается на языке VBA. Правила выбора модуля, в котором макрорекордер сохраняет макрос, являются достаточно сложными [1]. Записанный макрос можно вызывать для выполнения из основного меню, а также путем нажатия назначенной комбинации клавиш. Текст макроса можно редактировать.

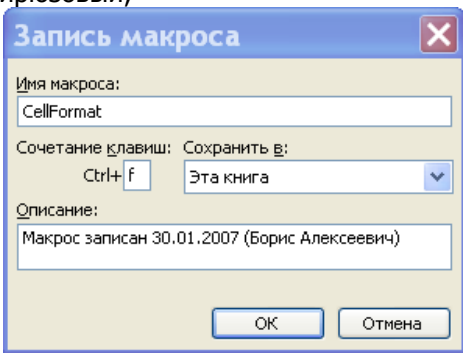
Пример.

К часто повторяющимся действиям можно отнести выбор нужного формата ячеек. Создадим макрос, который устанавливал бы выравнивание содержимого по горизонтали «по центру» и определенный цвет фона любого выделенного диапазона ячеек.

Для этого:

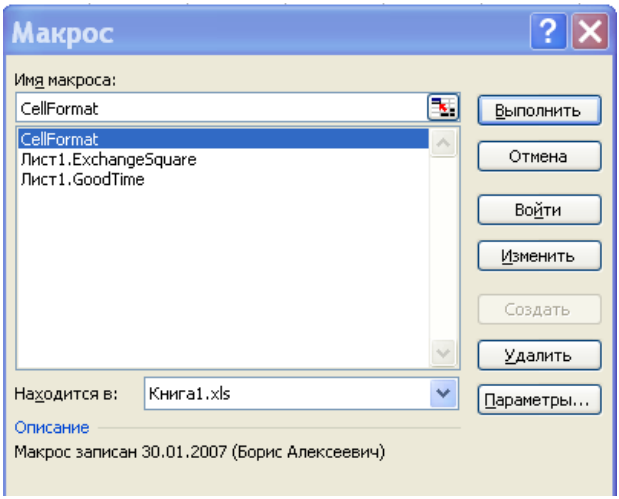
- выделим на листе Excel произвольный диапазон ячеек;
- выполним команду **Сервис | Макрос | Начать запись**. Появится окно диалога «**Запись макроса**», в котором введем имя макроса **CellFormat**, а также установим сочетание клавиш, по которому этот макрос можно запускать (выбрано Ctrl+f);

- выберем команду **Формат ячейки**, на закладке **Выравнивание** установим Выравнивание по горизонтали - «по центру», а на закладке **Вид** установим цвет заливки ячеек – светло-бирюзовый;

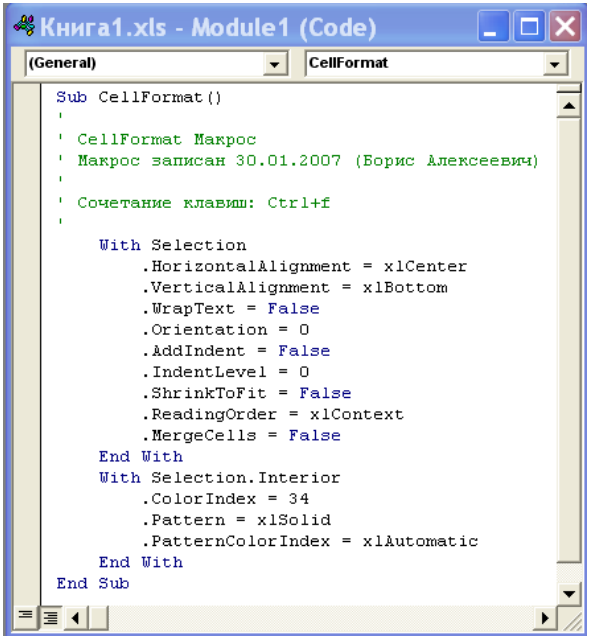


- завершим запись макроса, нажав на кнопку «**Остановить запись**», или выполнив команду меню **Сервис | Макрос | Остановить запись**.

Теперь, чтобы отформатировать в данной книге Excel любой диапазон ячеек соответствующим образом (выровнять их содержимое «по центру» и выделить светло-бирюзовым цветом), необходимо выделить этот диапазон и запустить макрос **CellFormat** из меню **Сервис | Макрос | Макросы | Выполнить** или нажав комбинацию клавиш Ctrl+f **.**



Проанализируем код макроса CellFormat, записанного при помощи макрорекордера (он записал его в модуль Module1):



Макрорекордер использовал особую структуру:

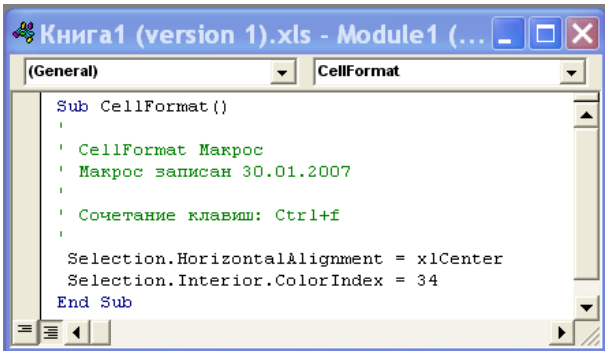
```
With Object
` операторы, использующие свойства и методы Object
End With
```

позволяющую ссылаться на ряд свойств или методов, принадлежащих одному и тому же объекту, без задания полной объектной ссылки каждый раз (подробнее об этом – в разделе 10).

В макросе CellFormat устанавливаются свойства выбранного объекта Selection (в нашем случае – любой диапазон ячеек Excel). Поскольку мы устанавливали только свойства:

HorizontalAlignment = xlCenter - выравнивание (Alignment) по горизонтали и «по центру»
Interior.ColorIndex = 34 - задание фону ячейки (Interior внутренность) цвета с кодом 34 (светло-бирюзовый),

а остальные свойства ячейки не меняли, т.е. их значения используются по умолчанию, то исходный код макроса можно отредактировать вручную, удалив стандартные установки:



Замечание. Как отмечалось в разделе 2, строки комментариев начинаются знаком апострофа (одинарная кавычка) и выделены зеленым цветом.

Таким образом, при изучении приемов программирования на VBA, весьма полезным и эффективным является анализ кодов различных макросов, записанных автоматически при помощи макрорекордера.

8.1. Упражнение

С помощью макрорекордера записать макрос, который помещал бы в ячейки A6:A8 название и адрес некоторой фирмы. Проанализировать код записанного макроса. Скорректировать макрос - перейти на относительные ссылки.

Замечание. Рядом с кнопкой «Остановить запись» на панели инструментов **Остановить запись** находится кнопка «Относительные ссылки». Ее надо нажать сразу после начала записи.

9. Основные понятия объектно-ориентированного программирования

Главная идея объектно-ориентированного программирования заключается в том, что программное приложение (как и реальный мир вокруг нас) должно состоять из особых объектов, каждый из которых имеет собственные специфические качества и поведение.

Объект (object) – это программный элемент, который имеет свое отображение на экране, содержит некоторые переменные, определяющие его свойства, и некоторые методы, управляющие им.

Свойство (property) представляет собой атрибут объекта, определяющий его характеристики. Чтобы изменить характеристики объекта, надо просто изменить значения его свойств.

Синтаксис: Объект.Свойство = ЗначениеСвойства

Например, инструкция: Application.Caption = "Пример" устанавливает новый заголовок окна (свойство Caption) приложения (объекта Application). Окно будет называться: «Пример»

Значения свойств можно использовать в выражениях, например:

If Объект.Свойство = Некоторое3начение Then …

Методы (methods) – это программные процедуры (с параметрами или без), реализующие некоторые алгоритмы, определяющие взаимодействие объекта с внешней средой. Методы могут быть встроенными или созданными непосредственно разработчиками приложения.

Вызов метода: Объект.Метод

К методам объекта можно обратиться, только используя объект. Например, Application.Quit – при помощи метода Quit (закрыть) закрывается приложе- ние (объект Application).

Программные объекты, как и объекты реального мира, группируются в ***семейства* (*Collections)***. Другие названия: классы, коллекции. Например, все рабочие листы книги Excel образуют семейство WorkSheets (множественное число от имени объекта WorkSheet - одного рабочего листа).

Все объекты одного семейства имеют одни и те же (или подобные) свойства и методы.

Семейства можно использовать одним из двух способов: либо какое-либо действие совершается над всеми объектами семейства (например, WorkSheets.Delete – удалить все листы), либо со ссылкой на семейство выбирается конкретный объект для работы с ним (например, инструкция WorkSheets("Лист2").Select выбирает рабочий лист Лист2 из активной рабочей книги Excel).

В объектно – ориентированном программировании важным является понятие события.

Событие (event) представляет собой действие, распознаваемое объектом (например, щелчок мыши или нажатие клавиши), для которого можно запрограммировать отклик. События возникают в результате действий пользователя программ, или же они могут быть вызваны системой.

Специальный вид процедур, генерирующих отклик на события, называется ***процедурами обработки событий*.** Если такой отклик не создан (не записана соответствующая процедура), то система никак не будет реагировать на это событие.

Процедуры обработки событий имеют следующий синтаксис:

```
Private Sub ИмяОбъекта_Событие()
« код обработки события»
End Sub
```

VBA является идеальным инструментом для изучении основ объектно-ориентированного программирования, так как имеет реальные встроенные объекты MS Office: рабочие книги и листы, ячейки, документы, формы, выделенные фрагменты текста - всего более 100, их свойства и методы. Поэтому для иллюстрации объектов, свойств и методов не нужно обращаться к объектам, не имеющим никакого отношения к программированию.

Операторы программ VBA, использующие объекты, обычно выполняют одно или несколько из следующих действий:

- определяют текущее состояние или статус объекта путем выборки значения, сохраняемого в определенном свойстве;

- изменяют состояние или статус объекта установкой значения, сохраненного в определенном свойстве;
- используют один из методов объекта, обеспечивая выполнение объектом одной из его встроенных задач.

Отметим еще несколько понятий объектно- ориентированного программирования:

Инкапсуляция — это скрытие информации. При объектно-ориентированном программировании возможен доступ к объекту только через его методы и свойства. Внутренняя структура объекта скрыта от пользователя, т.е. объекты — это самостоятельные сущности, отделенные от внешнего мира. Инкапсуляция позволяет изменять реализацию объектов любого класса без опасений, что это вызовет нежелательные побочные эффекты в программной системе. Это мощное средство обеспечивает многократное использование одного и того же программного кода, позволяя собирать программу из готовых модулей, как здание из отдельных кирпичиков, но различной архитектуры и функционального назначения.

Наследование — это возможность выделить свойства, методы события одного объекта и приписать их другому объекту, иногда с их модификацией. С точки зрения программиста, новый класс должен содержать только коды и данные для новых или изменяющихся методов.

Полиморфизм — это способность объектов выбирать операцию на основе данных, принимаемых в сообщении. Каждый объект может реагировать по-своему на одно и тоже сообщение. Например, команда Print будет по-разному воспринята черно-белым или цветным принтером.

Объектно-ориентированное приложение организует данные и выполняемые операторы программного кода в связанные объекты, что облегчает разработку, организацию и работу со сложными структурами данных и действиями, выполняемыми над этими данными. На примере имеющихся объектов, свойств и методов, с которыми работает VBA, можно создавать собственные объекты, добавлять новые свойства и методы.

10. Объектно-ориентированное программирование в среде MS Excel

В MS Excel наиболее важными объектами (с точки зрения программиста VBA) являются:

Application	- само приложение Excel (host- приложение)
WorkBook	- открытая рабочая книга
Sheet	- лист
WorkSheet	- рабочий лист
ActiveSheet	- активный лист
Window	- любое окно
Range	- диапазон ячеек (может включать только одну ячейку)
Cell	- ячейка
UserForm	- пользовательская форма
Chart	- диаграмма в рабочей книге
Font	- этот объект содержит атрибуты шрифта и стиля для текста в рабочем листе
Shape	- графический объект
CommandButton	- элемент управления «кнопка»
Selection	- любой выбранный объект

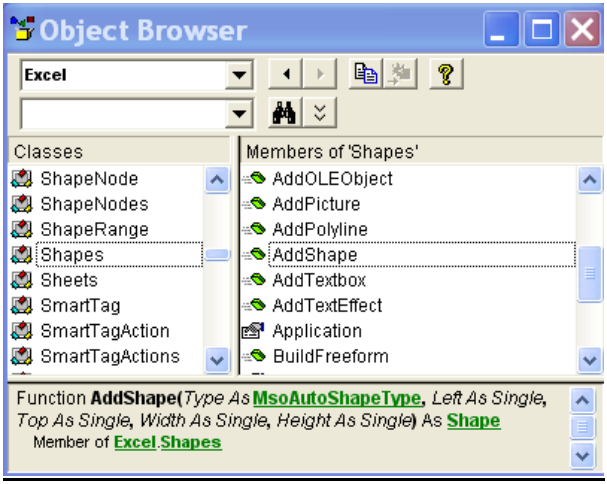
Возможность просмотра всех доступных VBA объектов (с их различными методами и свойствами) обеспечивает универсальный инструмент **Object Browser** (просмотр объектов). Отображение окна Object Browser на экране выполняется по команде **View | Object Browser**, нажатием соответствующей кнопки на панели инструментов или клавиши F2. Object Browser можно использовать также для нахождения и получения справок по макросам, процедурам, функциям и константам.

Чтобы просмотреть список доступных объектов host- приложения, а также список свойств и методов для каждого заданного объекта, нужно выполнить следующие шаги:

1. выбрать библиотеку в раскрывающемся списке **Project | Library** (например, библиотека Excel) в верхней части окна **Object Browser**;
2. в списке **Classes** (классы или семейства) выбрать интересующий объект (например, класс Shapes). Чтобы получить бо-

лее подробную информацию о выбранном объекте, нужно щелкнуть мышью на кнопке со знаком вопроса (?) в верхней правой части окна **Object Browser** или нажать клавишу F1;

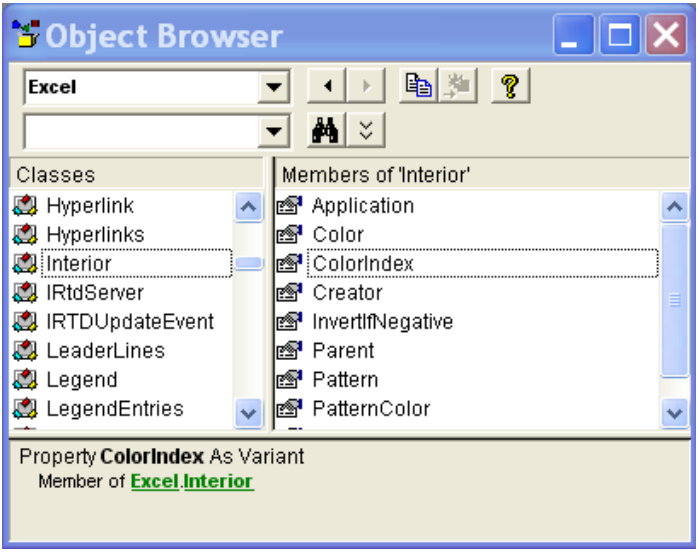
3. методы и свойства выбранного объекта отображаются в списке **Members of <class>** : свойства обозначаются значком с изображением руки, указывающей на индексную карту, а методы обозначаются значком, похожим на зеленый прямоугольник; имя метода или свойства с гиперссылками отображается в нижней части окна, причем для методов перечисляются все аргументы (если они есть). Ниже представлено окно **Object Browser** для выбранного метода AddShape, имеющего 5 аргументов. Более подробную информацию о свойствах, методах и аргументах можно также получить при нажатии кнопки со знаком вопроса, по клавише F1 или при переходе на гиперссылки.



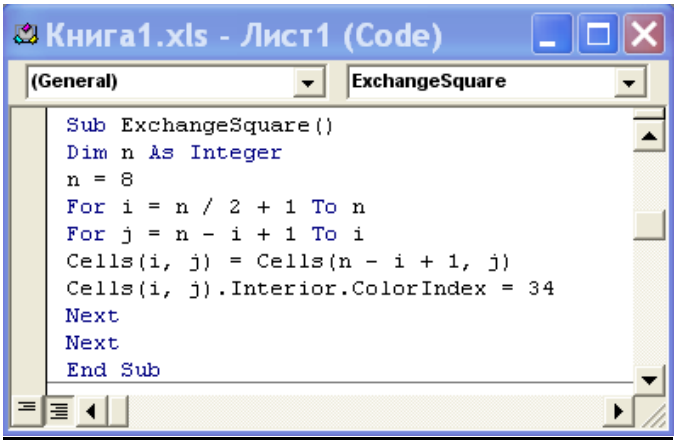
Пример.

Вернемся к макросу ExchangeSquare() раздела 6. Скорректируем его таким образом, чтобы фон ячеек отображенного треугольника стал светло-бирюзовым. В примере раздела 8 мы уже видели, что макрорекордер устанавливал светло-бирюзовый фон ячеек заданием свойства Inte- rior.ColorIndex = 34. Убедимся в

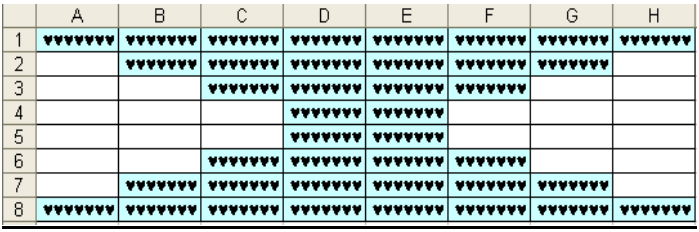
этом же, используя Object Browser:



Теперь, с учетом вышесказанного, внесем изменения в код макроса:



Результатом его выполнения будет следующая картина:



10.1. Упражнение

Изменить код макроса примера 6.1 таким образом, чтобы перемещаемые столбцы матрицы получили желтый фон и красный цвет шрифта.

11. Создание элементов управления интерфейса на листах MS Excel

Помимо использования стандартных возможностей, многие удачно реализованные на Excel проекты имеют удобный, максимально приспособленный для решения конкретной задачи интерфейс. Это можно осуществить, используя средства VBA.

Пример.

На Листе3 **поместить кнопку**, при нажатии которой вызывается калькулятор из стандартных программ Windows.

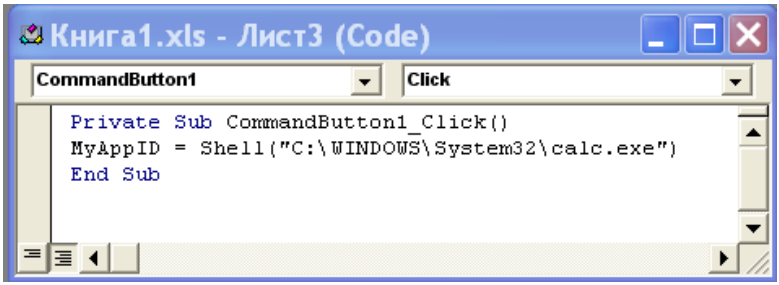
Перейдем на Лист3, активизируем панель инструментов **Элементы управления** и нажав на ней кнопку **Режим конструктора** переведем MS Excel в режим конструктора. Перетащим мышью элемент Кнопка на лист в нужное местоположение. На поверхности кнопки будет автоматически отображена надпись CommandButton1.

Нажмем кнопку **Свойства** на панели инструментов **Элементы управления** и изменим значения некоторых свойств Кнопки:

- Caption – введем надпись «Калькулятор»,
- Font - увеличим размер шрифта и установим его жирность,
- BackColor - установим красный цвет шрифта,

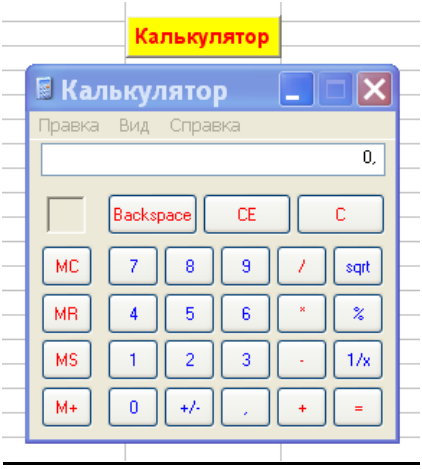
- ForeColor – установим желтый цвет фона кнопки.

Нажмем кнопку **Исходный текст** на панели инструментов, в результате откроется редактор VBA, причем в окне кода автоматически будут создана первая и последняя инструкции обработки события Click кнопки, генерируемого при ее нажатии. Добавим строку вызова функции Shell(…), которая будет запускать на выполнение программу калькулятора calc.exe



Замечание. Если функция Shell успешно запускает исполняемый файл, она возвращает ID запущенной программы – уникальный номер, который идентифицирует выполняемую программу. В противном случае, если функция Shell не может запустить программу, она возвращает значение MyAppID=0.

На Листе3 получаем результат нажатия кнопки:



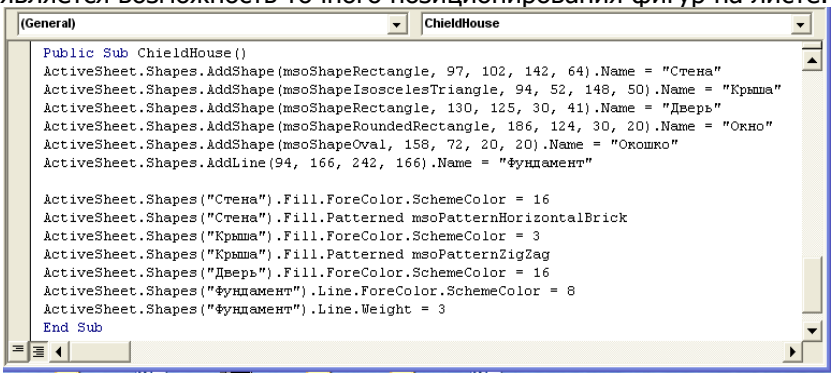
11.1. Упражнение

На Листе1 поместить кнопку, при нажатии которой активным становился бы Лист2 и выполнялся бы макрос примера 8.1.

Замечание. Для запуска макроса можно использовать метод Run(…).

12. Создание графического объекта

Поясним подробно технологию объектно-ориентированного программирования на простом наглядном примере создания на активном листе Excel (объект **ActiveSheet**) графического объекта - раскрашенного игрушечного домика. Объектную модель графики в VBA составляют специальные семейства, свойства и методы. Преимуществом программного черчения по сравнению с ручным является возможность точного позиционирования фигур на листе.



Сделаем пояснения к созданному коду:

- Тип процедуры **Public** означает, что она может быть использована для вызова в других процедурах.
- Метод **AddShape** создает новый графический объект семейства Shapes, в качестве параметров метода указываются: тип фигуры (Type), ее местоположение (координаты) на листе Excel и размеры; метод **AddShape** позволяет строить фигуры 139 типов – в их числе:

msoShapeRectangle	- прямоугольник
msoShapeRoundedRectangle	- прямоугольник с закругленными углами
msoShapeIsoscelesTriangle	- равнобедренный треугольник
msoShapeOval	- овал

Полный список фигур можно просмотреть через **Object Browser** или найти в справочнике VBA: нажать кнопку вызова справки и ввести в поле вопроса имя раздела – AddShape Method. Следующие два числовых параметра представляют собой коорди-

наты верхнего левого угла прямоугольника, мысленно описанного около выводимой фигуры данного конкретного типа: Left – горизонтальный отступ (X-координата) и Top - вертикальный отступ (Y-координата) от верхнего левого угла листа Excel, выраженные в точках (пикселях). В дальнейшем будем их называть *координатами начала фигуры.* И, наконец, два последних числовых параметра метода AddShape означают ширину Width и высоту Height фигуры, также выраженные в пикселях.

• Для удобства дальнейшего выбора каждому объекту присвоено собственное имя – свойство **Name**. Таким образом, к примеру, третья строка кода процедуры ChieldHouse означает, что графический объект «Крыша» представляет собой равнобедренный треугольник, начало которого (вершина верхнего левого угла описывающего прямоугольника) отстоит на 94 пикселя вправо и 52 пикселя вниз от верхнего левого угла листа Excel, длина основания треугольника равна 148, а высота – 50 пикселей.

• Последовательность вывода фигур имеет значение, так как фигура, выведенная позже, перекрывает предыдущую при наложении, т.е., если сначала вывести «Окошко», а потом «Крышу», то «Окошка» видно не будет. Прозрачность контуров предыдущих фигур связана со свойством Visible – по умолчанию прозрачность не установлена и Visible=msoTrue. Чтобы установить прозрачность, нужно установить Visible=msoFalse ;

• Графический объект «Фундамент» создан методом **AddLine(x1, y1, x2, y2)**, который используется для черчения отрезков прямой, где x1, y1 – координаты одного конца, а x2, y2 – координаты другого конца отрезка. При построении линий неважно, какой из концов отрезка считать первым, а какой – вторым.

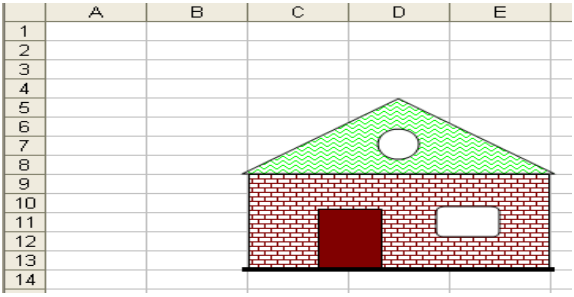
• После того, как графические фигуры созданы, можно задавать их свойства. Для установки характеристик фигуры используются последовательности свойств и методов, примененных друг за другом (отделяются точкой), начиная с исходного объекта-фигуры. Это необходимо потому, что в объектной модели VBA некоторые свойства применимы только к специальным объектам, созданным системой. Например, для окраски фигуры каким- либо цветом можно использовать следующую последовательность:

```
Объект.Fill.ForeColor.SchemeColor = НомерЦвета
```

При этом свойство **Fill** (заполнение цветом, заливка) порождает (выдает в качестве своего значения) системный объект **Fill-**

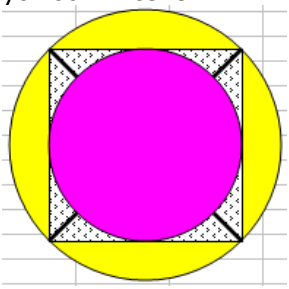
Format, имеющий свойства для задания цвета заливки (свойство **ForeColor**), типа штриховки узором (свойство **Patterned**), прозрачности заливки (свойство **Visible**) и другие. В свою очередь, к возвращаемому значению свойства **ForeColor** применяется свойство **SchemeColor** выбора конкретного цвета по номеру. Некоторые номера цветов: 1- белый, 2- красный, 3- зеленый, 4 –синий, 5- желтый, 6- лиловый, 7- бирюзовый, 8 – черный, 16- коричневый и т.д. Подробности о цветах заливки можно посмотреть по справке Color Constans, а по типам узоров – Patterned Method.

• Аналогично, свойство **Line** порождает объект **LineFormat**, свойства которого позволяют установить цвет (свойство **ForeColor**), толщину (свойство **Weight**), тип граничных (контурных) линий исходной фигуры (свойство **DashStyle**) и другие. Как отмечалось выше, все объекты и их свойства просматриваются также через **Object Browser**.



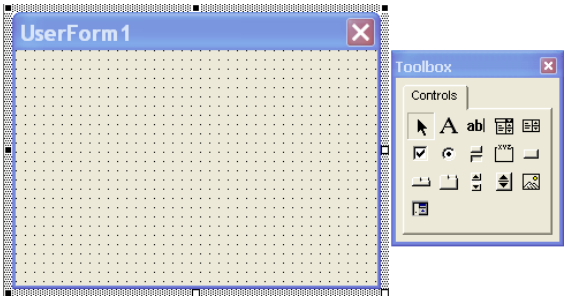
12.1. Упражнение

На листе Excel программно нарисовать и раскрасить фигуру. Внешний круг имеет радиус 100 пикселей.

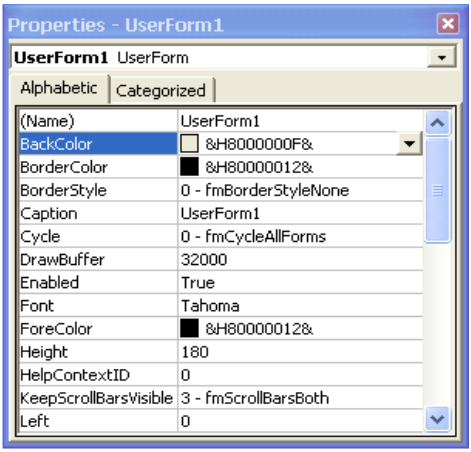


13. Создание пользовательской формы

Для создания собственных диалоговых окон разрабатываемых приложений в VBA используются формы. Редактор форм является одним из основных инструментом визуального программирования. Новые формы добавляются в проект выбором команды **Insert | UserForm** из меню Редактора VBA. В результате выводится незаполненная форма (объект UserForm1) с панелью элементов Toolbox, которые можно внедрять в форму, как объекты. На форме присутствует фоновая сетка точек для выравнивания внедряемых элементов управления.



Просмотреть или установить свойства пользовательской формы можно в диалоговом окне Properties, которое вызывается из меню **View | Properties Window** Редактора VBA, по клавише F4 или через контекстное меню – **Properties.**



Свойства отображаются на двух закладках: Alphabetic – по алфавиту и Categorized - сгруппированные. Большинство значений свойств «по умолчанию» как самой формы, так и отдельных элементов, вполне удовлетворительны, а в изменениях нуждаются только некоторые свойства. Ниже приводятся основные свойства объекта UserForm, которые обычно устанавливаются отдельно для каждой конкретной пользовательской формы:

Свойство	Описание
Name	Имя пользовательской формы.
Caption	Текст, отображаемый в строке заголовка формы.
BackColor	Цвет фона формы – выбирается на закладке Palette.
Font	Тип, стиль и размер используемого шрифта
Left, Top	Местоположение верхнего левого угла формы.
Height, Width	Высота и ширина формы. Эти свойства включают в себя толщину границы формы и строку заголовка.

Замечание. Последние свойства можно установить, изменяя размеры формы на экране путем протягивания за маркеры.

Некоторые стандартные методы и события, используемые при работе с формами:

Метод	Описание
Show	Отображает форму на экране.
Hide	Закрывает форму.
Move	Изменяет местоположение и размер формы
PrintForm	Печатает изображение формы.

Событие	Описание
Initialize	Происходит во время конфигурирования формы, но до ее загрузки.
Load	Происходит после инициализации формы, но до ее отображения на экран.
Unload	Событие, противоположное Load. Обычно используется для того, чтобы уточнить, действительно ли пользователь желает закрыть форму.
Click, DblClick	Происходят при щелчке и двойном щелчке на форме.

Элементы управления разработчик приложения может поместить на форму через кнопки, расположенные на панели элементов Toolbox путем их перетаскивания на форму.

Основные элементы управления:

Элемент управления	Имя
Поле	TextBox
Надпись	Label
Кнопка	CommandButton
Список	ListBox
Поле со списком	ComboBox
Полоса прокрутки	ScrollBar
Счетчик	SpinButton
Переключатель	OptionButton
Флажок	CheckBox
Выключатель	ToggleButton
Рамка	Frame
Рисунок	Image

Каждый элемент управления в форме является объектом, поэтому имеет набор свойств, которые отображаются в окне свойств элемента и могут быть изменены.

Общие свойства элементов управления:

Свойство	Описание
Name	Имя элемента управления
Caption	Возвращает или устанавливает надпись, отображаемую при элементе управления.
Left, Top	Местоположение верхнего левого угла элемента
Height, Width	Высота и ширина элемента
BackColor	Цвет фона
Font	Тип, стиль и размер используемого шрифта (если элемент выводит на экран текст)
AutoSize	Логическое свойство, которое устанавливает режим автоматического изменения размеров элемента управления так, чтобы на нем полностью помещался текст, являющийся значением свойства Caption.
Visible	Логическое свойство, которое определяет, надо ли отображать элемент управления во время выполнения программы.
Enabled	Логическое свойство, которое определяет, достижим ли для пользователя элемент управления во время работы приложения.

Рассмотрим более подробно некоторые элементы управления и перечислим их основные свойства и методы:

- **Надпись** (**Label**) – для создания поясняющих надписей на форме; основные свойства: **Caption, Font, TextAlign** (способ выравнивания);
- **Поле** (**TextBox**) – применяется для ввода текста, который в последующем используется в программе, или вывода в него результатов расчетов программы (после соответствующих преобразований форматов); основные свойства: **Text** (возвращает текст, содержащийся в поле), **MultiLine** (устанавли-

вает многострочный режим ввода текста в поле), **Font, TextAlign** и др.**;** одним из основных событий, связанных с полем ввода является событие **Change** (инициируется всякий раз, когда изменяется значение элемента управления);

- **Кнопка** (**CommandButton**) – в основном используется для инициирования выполнения некоторых действий, вызываемых ее нажатием, например, запуск программы или остановка ее выполнения, печать результатов и т.д.; основное свойство – **Caption** , возвращающее надпись на кнопке, а основное событие – **Click,** инициируемое всякий раз, когда по кнопке щелкают мышью;
- **Переключатель** (**OptionButton**) – позволяет выбрать одну из нескольких взаимоисключающих альтернатив; переключатели обычно отображаются группами по выбираемым альтернативам; группировка производится при помощи свойства **GroupName**, т.е. переключатели, которые имеют одно и то же значение этого свойства, образуют одну альтернативную группу вариантов. Наиболее часто используемым свойством переключателя является **Value**, которое возвращает значение **True**, если переключатель выбран, и **False** в противном случае, а также свойство **Caption**, возвращающее текст надписи, отображаемой рядом с переключателем. Основными событиями переключателя являются события **Click** и **Change**;
- **Флажок (CheckBox)** – предоставляет пользователю возможность выбора; имеет два состояния: установлен и сброшен. Имеет те же основные свойства **Value** и **Caption**, что и переключатель; основным событием является событие **Change**.

Как отмечалось выше, для каждой новой формы VBA создает отдельный модуль, в котором записываются коды процедур обработки событий формы и элементов управления, размещенных в ней, причем первая и последняя строки процедур формируются автоматически.

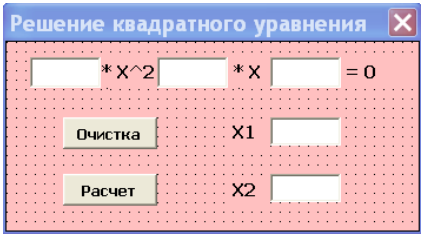
Таким образом, что создать интерфейс для разрабатываемой программы в виде пользовательской формы, необходимо:

- вставить в проект объект UserForm и настроить таблицу его свойств;
- разместить на форме необходимые элементы управления (надписи, поля, кнопки, флажки, рисунки и т.п.) и настроить таблицы их свойств;
- написать коды процедур обработки событий.

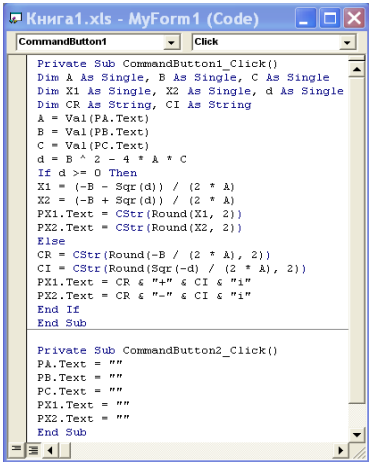
Пример.

Создадим пользовательскую форму для решения квадратного уравнения $A \cdot x^2 + B \cdot x + C = 0$, в том числе, и с комплексными корнями.

На форме (она названа как Name = MyForm1) разместим элементы управления **Поле** для ввода коэффициентов A, B и C (их имена PA, PB и PC соответственно) и вывода корней уравнения $X1$ и $X2$ (имена полей PX1 и PX2), две кнопки для очистки полей от данных предыдущего решения (кнопка с именем CommadButton2) и запуска программы (кнопка с именем CommadButton1), а также поясняющие надписи:



В автоматически сформированные заготовки для процедур событий кнопок допишем коды обработки событий (по нижеприведенному коду самостоятельно восстановите блок-схему алгоритма решения !):



Пояснения к коду: функция Val преобразует символьную информацию в числовую; функция CStr – наоборот, число в строку

символов; функция Round(…,n) округляет вещественное число до n знаков; & - оператор склеивания строковых выражений.

13.1. Упражнение

Создать пользовательскую форму для расчета сдачи за покупку товара с полями: Цена, Количество, Сумма, Оплачено, Сдача.

14. Использование встроенных функций Excel

При создании макросов VBA можно использовать некоторые встроенные функции Excel, которых нет среди встроенных функций VBA, например, различные функции, выполняющие математические, логические, финансовые и статистические операции над данными в рабочих листах.

Чтобы использовать функцию, принадлежащую Excel, необходимо обращаться к ней посредством программного объекта Application через свойство WorksheetFunction:

Application.WorksheetFunction.ИмяФункции(Список аргументов)

Объект Application VBA представляет host- приложение и все его ресурсы. Если Вы не уверены, доступна ли определенная функция Excel для VBA, используйте Object Browser, чтобы проверить содержит ли список Members эту функцию, при выбранном Application в списке Classes и при выбранном host- приложении в списке Project/Library. Если нужной функции нет в списке, то она недоступна для VBA.

Другая сложность использования встроенных функций Excel заключается в том, что VBA воспринимает только латинские

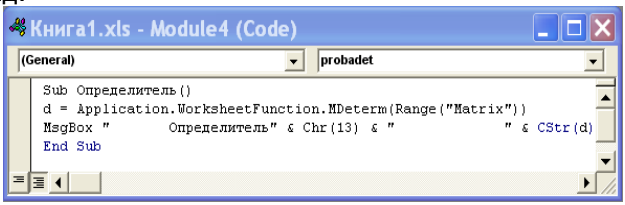
имена встроенных функций, в то время как Мастер функций русифицированной версии Excel использует русские аналоги имен функций (хотя справку выдает и по английским именам, если их знать). Например, для вычисления определителя матрицы в Excel можно использовать функцию как с русским именем МОПРЕД(Массив), так и с латинским именем MDETERM(…). Чтобы узнать латинское имя требуемой функции, нужно либо обратиться к хорошему справочнику, например, [2], либо запустить макрорекордер, вызвать функцию под русским именем, а затем посмотреть созданный код макроса.

Пример.

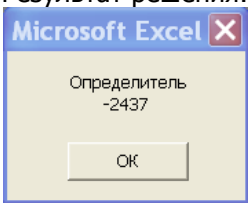
Найти определитель матрицы размерностью 5x5, которая представлена листе Excel. Результат вывести в стандартном окне MsgBox.

2	-1	1	1	3
1	5	-2	2	-1
-2	3	4	1	4
-1	-4	-3	2	1
3	1	5	-1	-4

Диапазону ячеек, составляющих матрицу, присвоим имя Matrix (через меню Вставка→ Имя→ Присвоить). Тогда код макроса имеет вид:



Результат решения:



15. Литература

1. Кузьменко В.Г. Программирование на VBA 2002. – М.: ООО «Бином- Пресс», 2003.
2. Гарнаев А.Ю. Excel, VBA, Internet в экономике и финансах. – Спб.: БХВ-Петербург, 2005.
3. Биллинг В.А. VBA и Office 2000. Офисное программирование. – М.: Издательско-торговый дом «Русская Редакция», 1999.
4. Мельников П.П. и др. Практикум по экономической инфоматике: Учебное пособие: ч.III, М: Финансы и статистика; Перспектива, 2002.