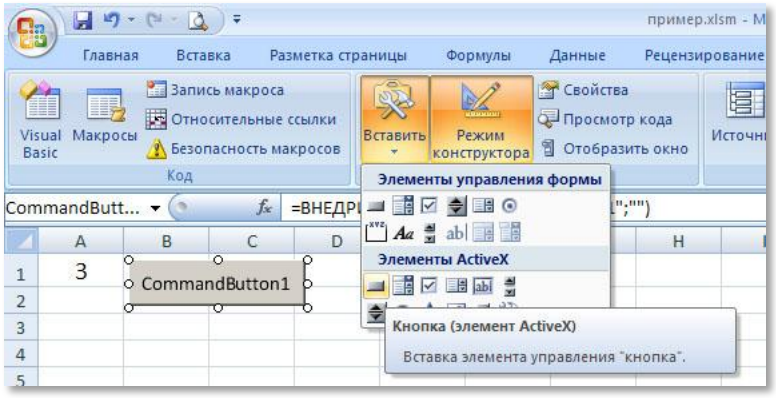
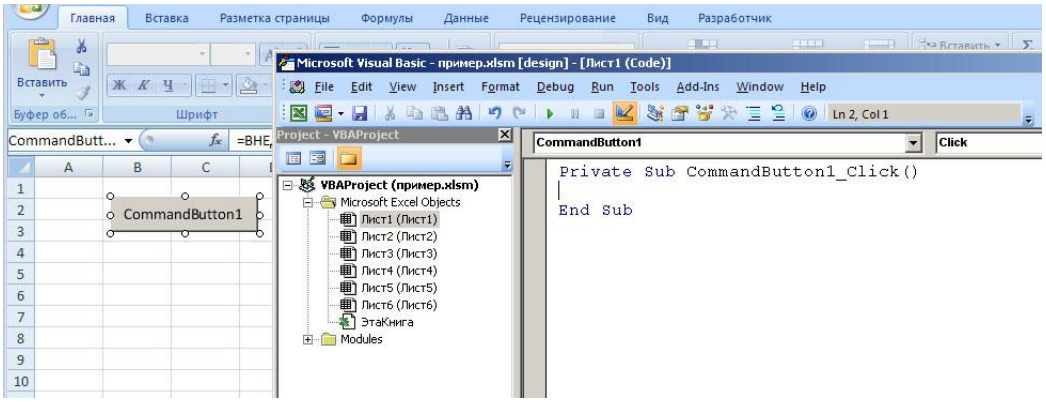
1. Подпрограммы VBA.

Подпрограмма это программный код, вынесенный из тела основной программы и имеющий собственное имя (идентификатор). Подпрограммы создают, прежде всего, в тех случаях, когда часть программного кода будет неоднократно использоваться при выполнении основной программы. Это позволяет минимизировать длину кода. Подпрограммы бывают двух видов: процедуры и функции. Подпрограммы могут иметь аргументы, через которые в подпрограмму передаются значения (входные аргументы) и возвращаются после вычислений значения (выходные аргументы) в основную программу. Подпрограмма, оформленная как функция, позволяет также возвращать в тело основной программы значение через имя функции. Идентификатор функции в этом случае можно рассматривать как переменную, в которую размещается значение по итогам вычисления функции.



Запустите Excel и перейдите на ленте на вкладку «Разработчик» (если она не видна, то её можно подключить через кнопку Office / круглая в верхнем левом углу / далее «Параметры Excel» и далее «Показывать вкладку Разработчик на ленте»). На вкладке Разработчик включите режим конструктора. С помощью клавиши вставить выберите из элементов ActiveX объект CommandButton и разместите его на листе Excel кликом мыши.

Создадим обработчик события для этой клавиши, кликнув дважды по ней мышкой.



В ответ на двойной клик редактор кода VBE сгенерирует пустую процедуру для обработки этого события. В режиме работы, когда выключена клавиша «Режим конструктора», одиночный клик мышкой по клавише `CommandButton1` будет запускать на исполнение процедуру `Sub CommandButton1_Click`.

```
Private Sub CommandButton1_Click()
  a = Cells(1, 1)
  b = Cells(2, 1)
  c = a ^ b
  Cells(3, 1) = c
End Sub
```

Напишем программный код, решающий задачу возведения числа из ячейки A1 в степень, указанную в ячейке A2. Ответ будем размещать в ячейке A3.

Разместите в ячейках A1 и A2 некоторые числа для контроля работоспособности процедуры. Отключите режим конструктора и запустите процедуру, кликнув по `CommandButton1`. Получится примерно такой результат:

Этот простой пример используем для ознакомления с организацией работы с функциями и процедурами в VBA. В качестве задачи для подпрограммы будет вычисление степени числа. Из обработчика события `CommandButton1_Click` на исполнение будет вызываться процедура или функция, реализующая возведение в степень числа. Для чего нужно будет использовать аргументы подпрограммы для передачи в неё значений и возвращение обратно результатов вычислений.

```
Sub p_st(x, y, z)
  z = x ^ y
End Sub
Private Sub CommandButton1_Click()
  a = Cells(1, 1)
  b = Cells(2, 1)
  p_st a, b, c
  Cells(3, 1) = c
End Sub
```

Начнем с разработки подпрограммы вида процедура. Измените программный код, написанный ранее следующим образом, предваряя обработчик события новой процедурой p_st:

В процедуре p_st первые два аргумента входные и третий выходной.

> Внимание! Все изменения, производимые с программным кодом, обязательно апробируйте изменением входных данных на листе Excel и запуском кода на исполнение нажатием на клавишу CommandButton.

Обратите внимание, что VBA не является строготипизированным языком программирования и использование переменных без указания типа данных, которые в них будут храниться. На самом деле под переменные, как и при формальном указании типа данных, выделяется определенное фиксированное пространство в оперативной памяти. В VBA для этих целей используется тип данных Variant, одна переменная данного типа занимает 16 байт. Такое неявное объявление типа данных для переменных универсально (в некотором смысле облегчает работу, в том числе, для начинающих программистов), но требует больше места в памяти и больше времени при циклической обработке данных.

```
Sub p_st(x As Integer, y As Integer, z As Double)
  z = x ^ y
End Sub
Private Sub CommandButton1_Click()
  Dim a As Integer
  Dim b As Integer
  Dim c As Double
  a = Cells(1, 1)
  b = Cells(2, 1)
  p_st a, b, c
  Cells(3, 1) = c
End Sub
```

При явном объявлении типов данных для переменных программный код можно было бы выразить следующим образом:

```
Function p_st(x As Integer, y As Integer) As Double
  p_st = x ^ y
End Function
Private Sub CommandButton1_Click()
  Dim a As Integer
  Dim b As Integer
  Dim c As Double
  a = Cells(1, 1)
  b = Cells(2, 1)
  Cells(3, 1) = p_st(a, b)
End Sub
```

Теперь исследуем организацию работы VBA с функциями. Преобразуем написанный ранее код до следующего вида:

Обратите внимание, что из подпрограммы p_st мы убрали третий аргумент, а результат вычисления передаем через само имя подпрограммы p_st. Так же не забудьте изменить объявление подпрограммы – `Function`.

Именно данная возможность позволяет разрабатывать так называемые рекурсивные функции, то есть такие которые вызывают сами себя. Они используются для итеративной обработки данных без применения операторов цикла.

Реализуем функцию p_st вычисления неотрицательной степени числа, но без использования оператора возведения в степень и с обыкновенным оператором параметрического цикла:

```
Function p_st(x As Integer, y As Integer) As Double
  p_st = 1
  For i = 1 To y
    p_st = p_st * x
  Next
End Function
```

До цикла инициализируем p_st значением 1, чтобы при вычислении нулевой степени любого числа получить результат 0.

Теперь перепишем код для реализации рекурсивной обработки данных:

```
Function p_st(x As Integer, y As Integer) As Double
  If y > 0 Then
    p_st = p_st(x, y - 1) * x
  Else
    p_st = 1
  End If
End Function
```

Обратите внимание, что в функции присутствует оператор ветвления, который реализует два пути перебора данных. Первый путь это шаг рекурсии (p_st = p_st(x, y – 1) * x), на котором производится изменение данных и рекурсивный вызов самой функции. Изменения такие: до этого был вызов функции p_st(x, y) для значения y, а теперь для измененного значения y-1. Если бы данные не изменялись, то рекурсия бы зациклилась. Второй путь это базис рекурсии, который предназначен для останова перебора при наступлении определенного условия.

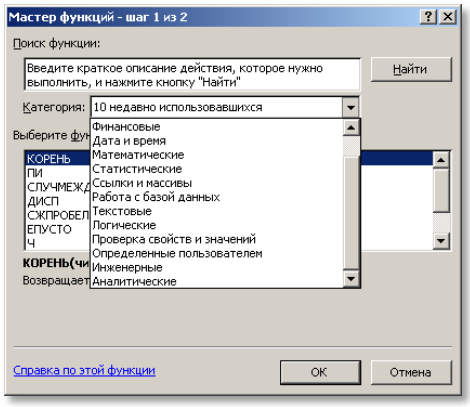
Обработка начинается со степени обозначенной пользователем, допустим нужно возвести число два в третью степень (2^3=2*2*2). Интерпретируем смысл кода следующим образом. Чтобы найти два в третьей, нужно сначала найти два во второй и только потом (когда будет найдено решения для «два во второй») домножить на два. В свою очередь рекурсивный вызов функции два во второй работает аналогично: чтобы найти два во второй нужно найти два в первой и домножить на два. Рекурсивный вызов функции два в первой: это два в нулевой домножить на два. Наконец, при запросе значения два в нулевой оператором if производится выбор второго пути реализации p_st=1 и рекурсия прерывается. На обратном ходе рекурсии один домножается на два положенные три раза и мы получаем искомый результат.

Задания для самостоятельного исполнения.

Разработайте рекурсивные функции вычисления:

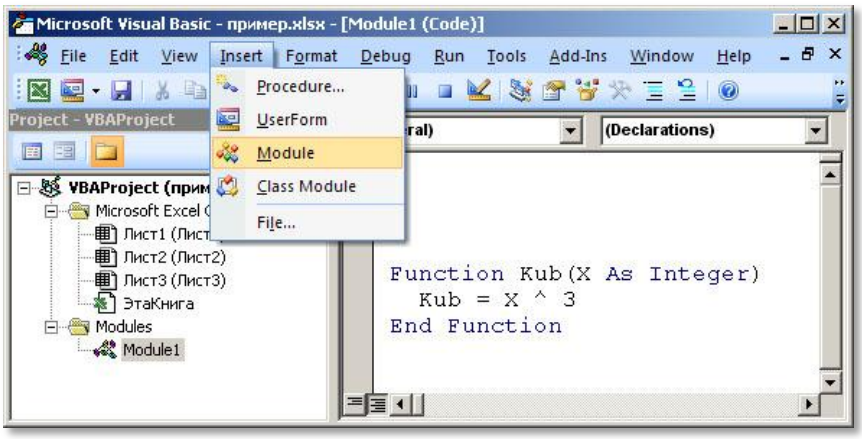
1) факториала числа N. Факториал числа N это произведение чисел от 1 до N. N!=1*2*3*…*N-1*N).
2) суммы чисел Фибоначчи. Последовательность чисел Фибоначчи начинается с двух единиц и каждое последующее число получается суммированием двух предыдущих: 1 1 2 3 5 8 13 21 34 55 89 …
3) суммы нечетных чисел от 1 до N

2. Разработка пользовательских функций для Excel.

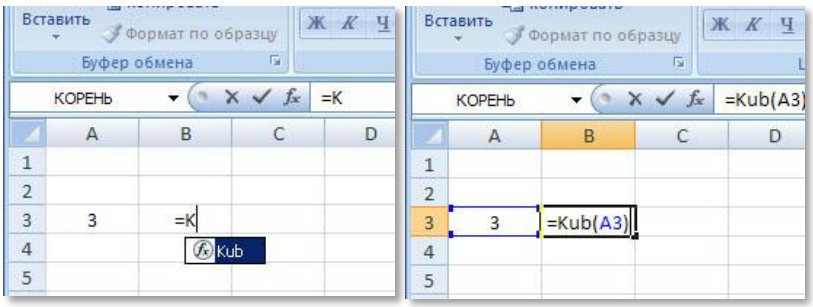


На листе Excel в ячейках вы можете размещать формулы, построенные на основе значительного количества встроенных функций (см. рис). Однако есть возможность использовать таким же образом и собственные функции. Для этого нужно сначала их разработать в редакторе VBE и затем подключить в формулу в ячейке в необходимом месте точно так же как вы подключаете стандартные функции Excel, то есть в ячейке начинаем набор формулы со знака «=» и далее пишем необходимый набор функций с адресами аргументов.

Начнем с простого примера. Пусть функция ищет значение куба числа – $y = x^3$. Назовем функцию Kub. Для входа в редактор кода нажмите сочетание клавиш Alt+F11. Если вы ранее не создавали модуль, то через меню Insert вставьте модуль и в редакторе кода опишите функцию (см. рис). Она без дополнительных манипуляций становится доступной для набора в формулах на листах Excel.



Начните набирать формулу в ячейке листа Excel обычным образом со знака «=», затем вводите нашу функцию и после первого же символа появится контекстная подсказка (см. рис), можно кликнуть по ней мышкой и, в качестве аргумента, указать адрес, откуда его брать. Нажмите Enter, закончив ввод формулы, и вы увидите, что она уже работает. Очень удобно и быстро. Естественно формулу можно копировать, редактировать и удалять как и при использовании стандартных функций. Таким образом можно создавать любые необходимые функции.

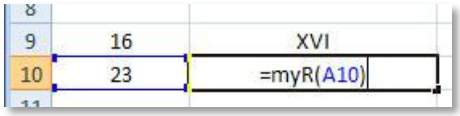


Задания для самостоятельного исполнения.

Создайте функции для перевода:

1) десятичного числа в римскую систему счисления (см. рис), используя этот код:

```
myR = Application.WorksheetFunction.Roman(D)
```

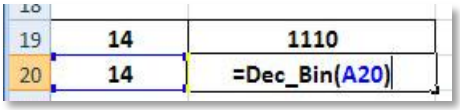


2) градусов в радианы, используя этот код:

```
R = G * Application.Pi / 180
```

3) целого десятичного числа в двоичное Dec_Bin (см. рис);

4) двоичного числа в десятичное Bin_Dec.

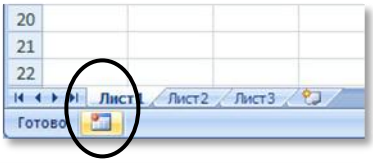


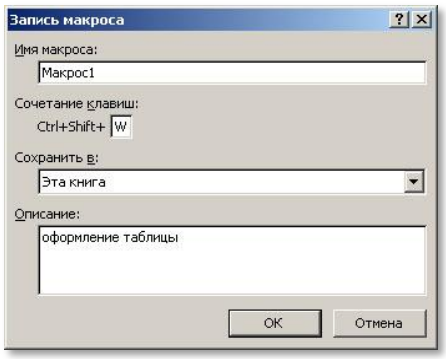
В заданиях №3 и №4 вам требуется самостоятельно написать программный код функций в модуле с использованием операторов цикла.

3. Автоматизация операций оформления таблиц.

Для программирования операций оформления (фон, шрифт, цвет, рамки и т.п.) удобно использовать запись макроса. Дело в том, что вы просто будете производить действия по настройке оформления, а редактор VBE будет за вас прописывать программный код, который реализует эти действия. В дальнейшем вы можете редактировать полученный программный код.

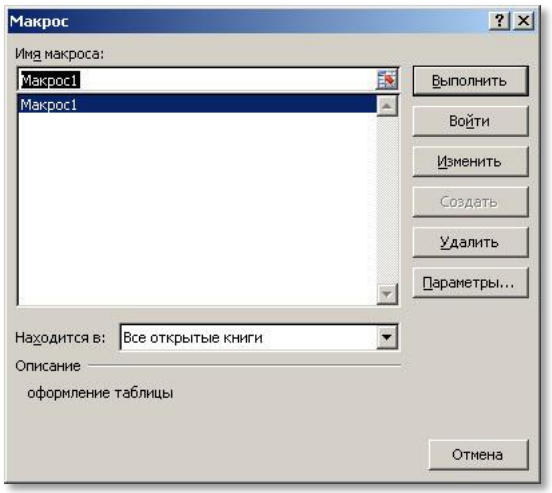
Нажмите клавишу записи в нижней левой части окна Excel (аналогичная клавиша есть на вкладке «Разработчик»). В появившемся диалоговом окне установите сочетание клавиш для запуска макроса, например, Ctrl+Shift+W, нажав Shift+W (его в дальнейшем можно будет изменить, доступны сочетания с клавишей Ctrl и с клавишами Ctrl+Shift).





После нажатия на экранную клавишу ОК диалогового окна клавиша записи макроса изменит свой вид (синий квадрат), что означает что запись макроса начата и вы можете производить действия по настройке оформления. Выделите указателем мыши диапазон ячеек B3:C5, затем установите цвет заливки желтый и кликните один раз мышкой на ячейке A1 чтобы снять выделение. Остановите запись макроса повторным нажатием на клавишу записи макроса.

Теперь можно просмотреть программный код, выполняющий описанные действия. Нажмите сочетание клавиш Alt+F8 или клавишу «Макросы» на вкладке «Разработчик». В открывшемся диалоговом окне посредством клавиши «Параметры» можно изменить горячие клавиши запуска макроса. С помощью клавиши «Изменить» откройте программный код:




```
Sub Макрос1()
  Range("B3:C5").Select
  With Selection.Interior
    .Pattern = xlSolid
    .PatternColorIndex = xlAutomatic
    .Color = 65535
    .TintAndShade = 0
    .PatternTintAndShade = 0
  End With
  Range("A1").Select
End Sub
```

В первой строчке устанавливается диапазон выделенных ячеек: `Range("B3:C5").Select`.

В дальнейшем удобнее будет диапазон назначать не строковой константой, а с помощью чисел или целочисленных переменных, например, так:

```
Range(Cells(3, 2), Cells(5, 3)).Select.
```

Последующий оператор `With` выделяет внутреннюю часть выделенного диапазона `Selection.Interior` и назначает различные настройки. Нас интересует только настройка цвета заливки, поэтому остальные составляющие можем исключить. Причем цвет можно назначать не только в десятичном виде, но и константами (vbYellow, vbRed и т.д.) или даже функцией RGB. Возможные варианты оформления оператора `With`:

```
With Selection.Interior
  .Color = 65535
End With
```

```
With Selection.Interior
  .Color = vbGreen
End With
```

```
With Selection.Interior
  .Color = RGB(0,255,0)
End With
```

Первый аргумент функции RGB задает значение красной составляющей цвета, второй – зеленой и третий – синей. Каждый цвет задается в диапазоне от 0 до 255.

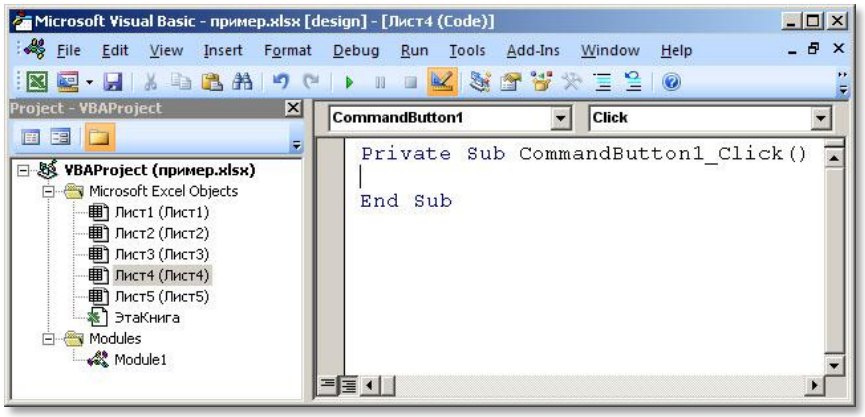
Макрос можно запускать на исполнение не только установленным ранее сочетанием клавиш, но и прямо из редактора кода VBE (курсор должен находиться в пределах процедуры от `Sub` до `End Sub`), нажав клавишу F5 или экранную клавишу Пуск (зеленый треугольник ► в меню сверху). Очистите область выделения на листе Excel от установленной ранее заливки цветом или просто переключитесь на другой лист Excel и апробируйте сокращенный макрос.

Задание для самостоятельного исполнения.

1) Самостоятельно создайте макрос очистки всего листа Excel от заливки.

Программный код можно перенести из макроса и использовать в рамках процедур обработки событий. Мы создадим две кнопки на листе: одна будет создавать заливку указанных ячеек, другая стирать заливку со всего листа. Установите на лист Excel в верхней части две клавиши CommandButton (вкладка «Разработчик», кнопка «Вставить», Элементы ActiveX). Клавишу «Режим конструктора» нажмите для перехода в режим настройки компонентов и редактирования кода, для переключения в режим работы (исполнения кода) отожмите клавишу «Режим конструктора».

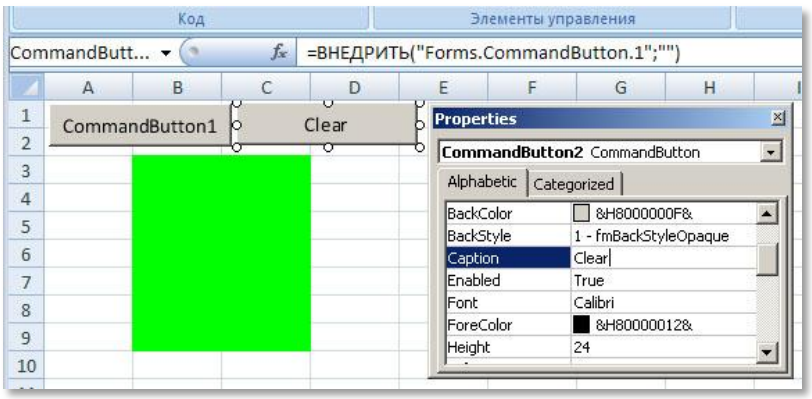
Находясь в режиме конструктора кликните два раза по клавише CommandButton1 – автоматически сгенерируется процедура обработки события клик мышкой по клавише. Этот код будет привязан к текущему листу Excel. Скопируйте туда код первого макроса (макросы описаны в модуле – он общий для всех листов). Код обработчика события для первой клавиши:




```
Private Sub CommandButton1_Click()
  Range(Cells(3, 2), Cells(5, 3)).Select
  With Selection.Interior
    .Color = vbGreen
  End With
  Range("A1").Select
End Sub
```

Аналогичным образом создайте обработчик события для второй клавиши (стирание).

Надписи на клавишах CommandButton можно поменять через инспектор Свойств (Properties), для чего нажмите правой клавишей мыши на объект CommandButton и в контестном меню выберите опцию «Свойства», в открывшемся списке (Properties) найдите поле Caption и измените его содержимое по вашему усмотрению.



Отключите режим конструктора и апробируйте работу обеих клавиш CommandButton.

Используя данный подход (через создание макросов) можно разрабатывать разнообразные процедуры автоматизирующие оформительскую работу.

Например, дополним процедуру первой клавиши диалогом с пользователем, в ходе которого уточняется количество строк в будущей таблице:

```
Private Sub CommandButton1_Click()
  K = InputBox("введите количество строк")
  Range(Cells(3, 2), Cells(2 + K, 3)).Select
  With Selection
     .Interior.Color = vbGreen
  End With
  Range("A1").Select
End Sub
```

Апробируйте работу измененной процедуры. Дальнейшие задания будут посвящены совершенствованию данной процедуры.

Задания для самостоятельного исполнения.

Дополните процедуру `CommandButton1_Click` (используя запись макросов) так, чтобы она:

2) не только заливала цветом ячейки листа, но и рисовала рамки вокруг ячеек (постарайтесь максимально сократить излишки программного кода);

3) уточняла у пользователя не только количество строк, но и количество столбцов и строила таблицу соответствующих размеров;

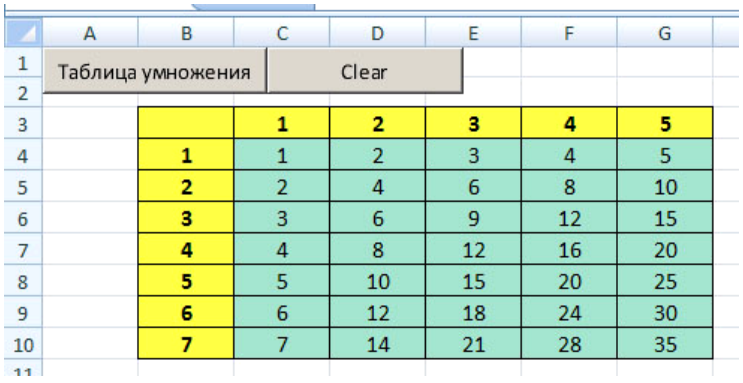
4) заголовки таблицы (первый столбец и первую строку) заливала цветом отличным от цвета остальных ячеек (цвет – на ваше усмотрение);

5) шрифт для заголовков таблицы устанавливала полужирный, а для остальных ячеек оставляла нормальным;

6) заполняла заголовки (первый столбец и первую строку) порядковыми номерами. Тут вам потребуется прописать два отдельных цикла: один расставляет номера по верхней строке, а другой – по левому столбцу. Напомню, что для записи в ячейку листа следует использовать оператор присвоения, например, `cells(5,2)=2`. Естественно, в качестве координат ячеек листа можно использовать не только числовые константы, но и целочисленные переменные:

```
For r = 1 To 5
  Cells(r, 1) = r        Этот цикл заполняет первый столбец.
Next r
```

7) заполняла ячейки таблицей умножения (тут вам потребуется прописать два цикла заполнения содержимого ячеек), например, так:



	1	2	3	4	5
1	1	2	3	4	5
2	2	4	6	8	10
3	3	6	9	12	15
4	4	8	12	16	20
5	5	10	15	20	25
6	6	12	18	24	30
7	7	14	21	28	35

4. Обработка текстовых файлов.

Текстовый файл представляет собой последовательность строк. Как правило, текстовый файл имеет расширение txt. Нередко встречается задача извлечь из текстового файла некоторые данные, обработать их, возможно даже сохранить изменения в текстовом файле. В свою очередь листы книги Excel удобны для размещения данных и их визуального представления в табличной форме.

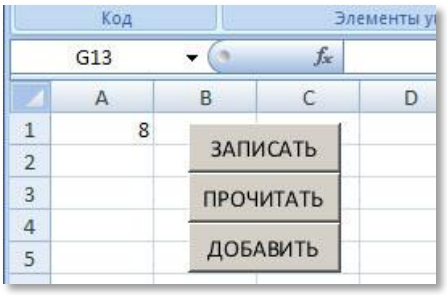
Разработаем три процедуры:

1) первая будет создавать файл test.txt с одной строкой, содержащей случайное целое число в диапазоне от 1 до 100;

2) вторая будет считывать содержимое файла test.txt построчно и выводить в последовательные ячейки памяти на лист Excel;

3) третья будет добавлять к файлу test.txt одну строку, содержащую случайное целое число в диапазоне от 1 до 100.

Разместите на новом листе три клавиши ComandButton. Измените надписи на них в свойстве Caption (ЗАПИСАТЬ, ПРОЧИТАТЬ, ДОБАВИТЬ), в соответствии с их предназначением. Для каждой клавиши сгенерируйте соответствующую процедуру (описание процедур далее по тексту).



Рассмотрим процедуру для первой клавиши. В первой строке процедуры формируем имя файла, включая и путь к нему – файл будет располагаться в текущей директории, то есть в той, в которой хранится книга Excel.

```
Private Sub CommandButton1_Click()
  NameFile = ThisWorkbook.Path + "\" + "test.txt"
  Open NameFile For Output As #1
  Print #1, LTrim(Int(10 * Rnd))
  Close #1
End Sub
```

Далее открываем файл для записи (`For Output`) и даём ему номер, по которому и будем обращаться при выводе данных. В следующей строке выводим в файл одно случайное число.

Для генерации случайного числа используем функцию Rnd, которая возвращает случайное вещественное число в диапазоне [0, 1). Функцией Int преобразуем число в целое. Умножением на 10 расширим диапазон – [0, 9). Функцией LTrim удалим один пробел слева, резервируемый под знак числа (в данном случае числа положительные).

Завершаем процедуру закрытием файла.

Запустите процедуру и оцените содержимое созданного файла (откройте Блокнотом в Проводнике).

Следующая процедура открывает файл для чтения (`For Input`), читает одну строку в переменную Stroka (`Line Input #1, Stroka`) и выводит результат в ячейку A1 (`Cells(1, 1) = Stroka`).

```
Private Sub CommandButton2_Click()
  NameFile = ThisWorkbook.Path + "\" + "test.txt"
  Open NameFile For Input As #1
  Line Input #1, Stroka
  Cells(1, 1) = Stroka
  Close #1
End Sub
```

Запустите вторую процедуру и посмотрите результат на листе Excel.

Третья процедура открывает файл для дозаписи:

```
Private Sub CommandButton3_Click()
  NameFile = ThisWorkbook.Path + "\" + "test.txt"
  Open NameFile For Append As #1
  Print #1, LTrim(Int(10 * Rnd))
  Close #1
End Sub
```

Содержимое третьей процедуры практические совпадает с первой процедурой за исключением изменения способа открытия файла (`For Append`), позволяющего добавлять строки к уже существующим.

Запустите третью процедуру – в файл добавится строка с очередным случайным числом. Оцените содержимое измененного файла (откройте Блокнотом в Проводнике) – увидите уже две строки со случайными числами. Каждое нажатие на клавишу CommanButton3 (ДОБАВИТЬ) будет увеличивать количество строк в файле. Однако, если вы попытаетесь нажатием на клавишу CommanButton2 (ПРОЧИТАТЬ) считать и вывести на лист Excel результаты, то потерпите неудачу – будет считываться каждый раз только содержимое первой строки. В данном случае необходимо доработать процедуру.

Обеспечим считывание всех строк из файла циклом с предусловием (`While`). Чтение будем производить пока курсор не достигнет конца файла (не закончатся строки в файле):

```
Private Sub CommandButton2_Click()
  NameFile = ThisWorkbook.Path + "\" + "test.txt"
  Open NameFile For Input As #1
  r = 1
  While Not EOF(1)
    Line Input #1, Stroka
    Cells(r, 1) = Stroka
    r = r + 1
  Wend
  Close #1
End Sub
```

В процедуре используется переменная r (от Row – строка) – счетчик строк. До входа в цикл мы инициализируем переменную значением 1 и первую считанную из файла строку (`Stroka`) выводим в первую ячейку первой строки листа Excel (`Cells(r, 1) = Stroka`). Затем переменная r увеличивается, обеспечивая на следующем шаге цикла запись второго считанного из файла значения уже в следующую ячейку листа. Так будет продолжаться пока истинно условие «не конец файла» (`While Not EOF(1)`), тем самым обеспечивается последовательный вывод всех строк из файла на лист Excel.

Задания для самостоятельного исполнения.

Текстовые файлы и строки.

1. Создайте процедуру, которая будет формировать текстовый файл text.txt, содержащий 10 строк. В каждой строке случайное число в диапазоне от 0 до 99.

2. Создайте процедуру, которая будет считывать содержимое текстового файла text.txt и размещать значения в столбик на листе Excel

3. В текстовом файле хранятся данные. В каждой строке через символ «пробел» записаны: Фамилия Имя Возраст Пол. Пример файла:

```
Иванов Иван 28 м
Сидоров Сидор 24 м
Маринина Марина 22 ж
```

Количество строк заранее неизвестно. Файл создайте самостоятельно с произвольным содержимым и с количеством строк от 10 до 20.

3.1. Обеспечить считывание файла и автоматическое создание таблицы на листе Excel с нумерацией и границами. Примерный вид итоговой таблицы:

№	Фамилия	Имя	Возраст	Пол
1	Иванов	Иван	28	м
2	Сидоров	Сидор	24	м
3	Маринина	Марина	22	ж

3.2. Проверить – является ли столбец имен множеством (элементы множества не повторяются). Результат вывести в произвольную ячейку на листе.

3.3. Выбрать из столбца имен те элементы, которые составляют множество имен и разместить их отдельно от таблицы (в столбце на этом же листе).

3.4. Упорядочить множество имен – построить рядом справа список элементов данного множества, упорядоченный по возрастанию или убыванию (на ваш выбор).

3.5. Разработать процедуру, обеспечивающую диалог с пользователем: предоставляется возможность ввести имя, в ответ в диалоговом окне выводится результат – имя принадлежит множеству имен или нет.

4. В текстовом файле №1 хранится множество A фамилий студентов имеющих средний балл не ниже оценки «4». В текстовом файле №2 хранится множество B фамилий студентов активно участвующих в жизни факультета. Каждая фамилия в отдельной строке.

Пример файла №1:

```
Иванов
Сидоров
Маринина
```

Пример файла №2:

```
Иванов
Маринина
```

Количество строк заранее неизвестно. Файлы создайте самостоятельно с произвольным содержимым и с количеством строк от 5 до 10.

Разработать процедуру, обеспечивающую решение следующих задач:

4.1. построить столбец содержащий множество $C=A \cap B$;

4.2. построить столбец содержащий множество $C=A \setminus B$;

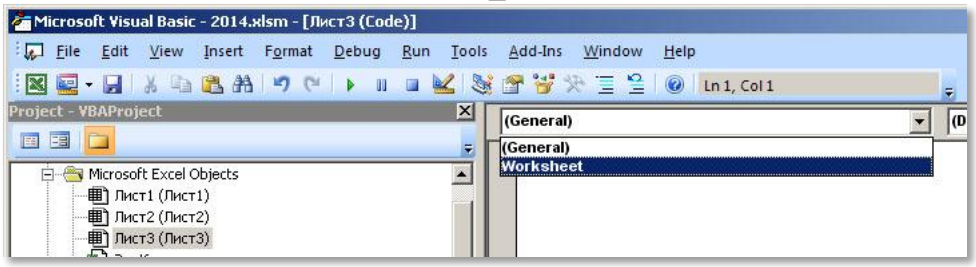
4.3. построить столбец содержащий множество $C=A \cup B$.

5. Событийное программирование.

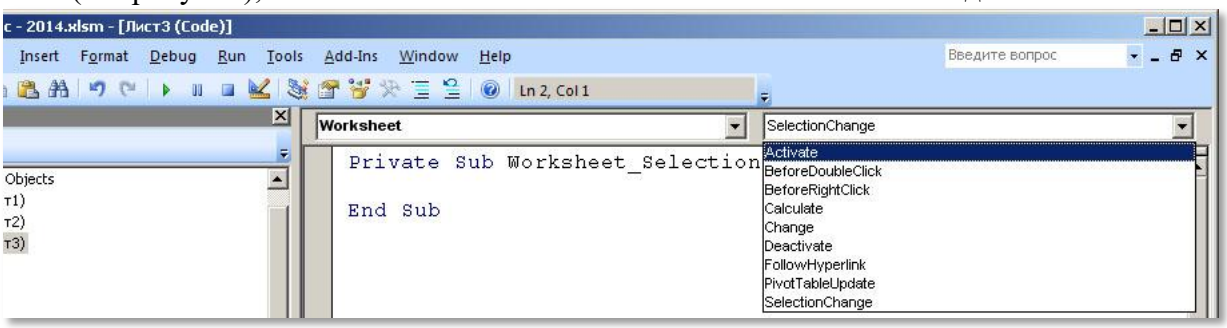
Как вы уже заметили, в VBA нет единой программы – программный код разбит на отдельные участки и сформирован в виде подпрограмм: процедур и функций. Некоторые подпрограммы вызываются к исполнению из других подпрограмм, но чаще всего программист привязывает исполнение процедуры к какому-либо событию. И мы уже такое делали ранее. Обратите внимание, что большинство процедур, разработанных ранее, запускаются при возникновении события клик мышкой по экранной клавише и в названии процедуры это явно указано: `Sub CommandButton1_Click()`. Такой способ формирования кода называется событийным программированием. События, к которым могут быть подключены процедуры могут быть разными: клик мышкой, нажатие клавиши на клавиатуре, изменение выделения на листе, движение мыши, события таймера и т.п. Рассмотрим лишь некоторые для примера.

Пример №1.

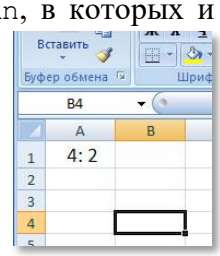
Создадим процедуру, которая будет отслеживать координаты ячейки на листе, по которой мы будем кликать мышкой или передвигаться на неё клавишами-стрелками. Откройте редактор VBE, выберите из доступных объектов `Worksheet` (рабочий лист), в ответ редактор сгенерирует пустую процедуру `Sub Worksheet_SelectionChange(ByVal Target As Range)`.



Справа от выбора объектов есть выпадающий список доступных для этого объекта событий (см. рисунок), но сейчас остановимся на событии «Изменение выделения».



Обратите внимание, что в данной процедуре есть переменная `Target`, её тип данных `Range`, то есть диапазон. Переменная `Target` содержит поля `Row` и `Column`, в которых и хранятся текущие координаты. Воспользуемся этим обстоятельством для реализации нашей задачи. Координаты текущей ячейки будем записывать в ячейку A1. Напишите и апробируйте процедуру:




```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  Cells(1, 1) = Str(Target.Row) + ":" + Str(Target.Column)
End Sub
```

Пример №2.

Перейдите на новый лист Excel. Создадим процедуру, которая будет отслеживать, как часто пользователь просматривает определенный лист книги Excel. Используем для этого соответствующее событие – активация листа. Напомню, что справа от выбора объектов есть выпадающий список доступных для этого объекта событий (см. рисунок ранее). Теперь в списке выберите событие `Activate` (Активация листа). В ответ редактор кода сгенерирует пустую процедуру, заполните её следующим образом:

```
Private Sub Worksheet_Activate()
  Cells(1, 1) = Cells(1, 1) + 1
End Sub
```

Теперь можете попробовать переключаться на другие листы и возвращаться на этот. Каждый раз при возвращении счетчик будет увеличивать количество учтенных просмотров листа на единицу. Естественно, можно ячейку счетчик сделать незаметной (белый шрифт или расположить далеко за пределами просматриваемого окна).

Используя описанный подход можно создавать меню программы прямо из ячеек листа.

Для просмотра и пролистывания изображений может потребоваться компонент Image. Разместить изображение на данный компонент вы сможете посредством поля Picture. Если потребуется во время работы формы динамически менять изображение (загружать новое), то пригодится следующий метод:

```
UserForm1.Image1.Picture = LoadPicture(ThisWorkbook.Path + "\" + "ris.jpg")
```

Данный метод загружает указанный рисунок из той же папки, что и сама книга Excel.

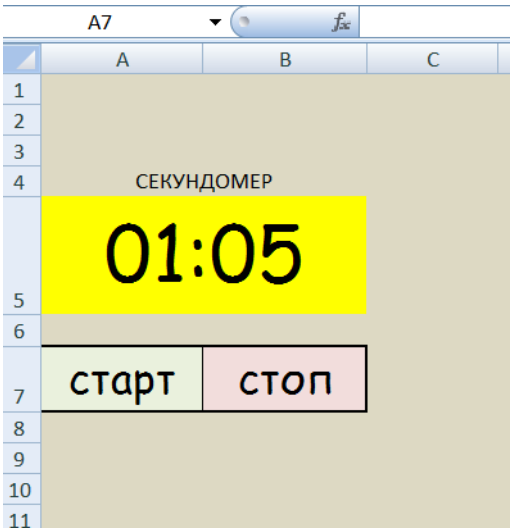
Свойство AutoSize регулирует размеры компонента – установите его в значение True и компонент Image будет растягиваться до размеров рисунка. Если же, наоборот, требуется рисунок, загружаемый в компонент Image, сжимать/растягивать до размеров компонента, то воспользуйтесь свойством PictureSizeMode.

Пример №3.

События таймера.

Обсудим один из вариантов отслеживания событий таймера. Если организовать обычный цикл, в теле которого отслеживать изменения системного таймера, то само приложение Excel перестанет быть «кликабельным» (зависнет). Для преодоления описанной проблемы предусмотрена специальная процедура передачи фокуса от программы пользователя к книге Excel – DoEvent, которая позволит обрабатывать события, происходящие с книгой, во время работы программы пользователя.

Ниже приведён пример «Секундомер». Кнопки Старт и Стоп реализованы через ячейки листа и вывод посекундный осуществляется также непосредственно на лист.




```
Dim b As Boolean

Function toStr(num)
  If num < 10 Then
    toStr = "0" + LTrim(num)
  Else
    toStr = LTrim(num)
  End If
End Function

Function toMinSek(sek)
  Min = sek \ 60
  sek = sek Mod 60
  toMinSek = toStr(Min) + ":" + toStr(sek)
End Function

Sub delay(z)
  Dim old As Long
  old = Timer
  Do
    DoEvents ' чтобы не зависало приложение Excel
  Loop While Timer - old <= z
End Sub

Sub start()
  Dim startTime As Long
```

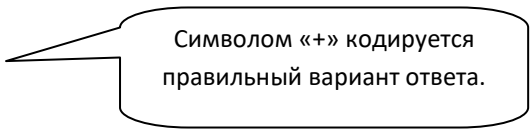
```
  startTime = Timer
  While b
    sek = toMinSek(Fix(Timer - startTime))
    Cells(5, 1) = sek
    delay (1)
    DoEvents ' чтобы не зависало приложение Excel
  Wend
End Sub

Private Sub Worksheet_SelectionChange(ByVal Target As Range)
  r = ActiveCell.Row
  c = ActiveCell.Column
  If r = 7 Then
    If c = 1 Then b = True: start
    If c = 2 Then b = False
  End If
End Sub
```

Задания для самостоятельного исполнения.
Событийное программирование в VBA.

В текстовом файле хранится одно тестовое задание по ПДД. Структура файла:

```
вопрос 1
ris_01.jpg
– вариант ответа 1
+ вариант ответа 2
– вариант ответа 3
– вариант ответа 4
```

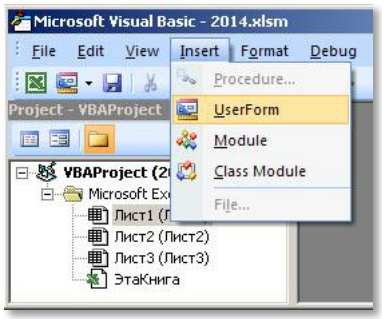


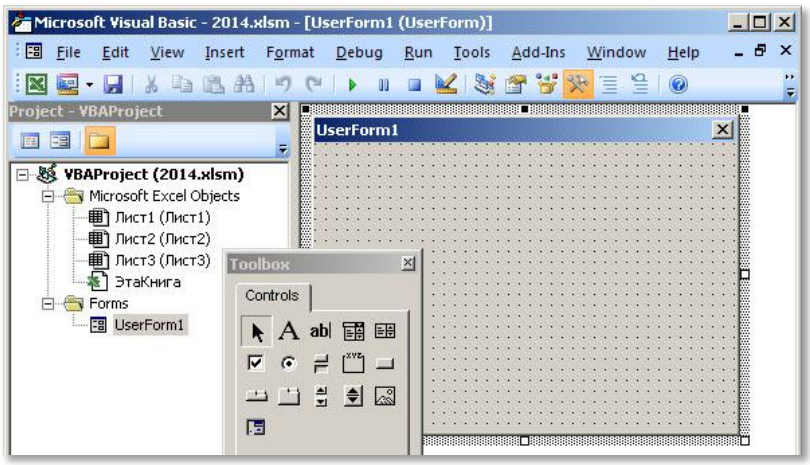
На листе Excel находятся пункты меню: «Начать тестирование» и «Закончить тестирование». При клике по первому пункту меню включается таймер, на лист в ячейку загружается вопрос, в объект `Image` – рисунок из текущей папки, затем в последовательные (сверху-вниз) ячейки листа – варианты ответа. Символ «+» подменяется на символ «–» при отображении в ячейке листа. Пользователь может кликать разные варианты ответа, при этом они «подсвечиваются» фоном. Если пользователь кликнул «Закончить тестирование», то в специально отведенное место на листе (ячейки листа) выводятся затраченное время и правильность выбранного варианта ответа.

6. Форма и элементы управления.

Бывают случаи, когда программный код зависит от многих параметров, выбираемых пользователем. Для организации диалога с пользователем нужны различные элементы управления (кнопки, переключатели, флажки, поля со списком и т.п.), но размещать их прямо на листе книги не совсем удобно. В этом случае имеет смысл вынести элементы управления на отдельную форму.

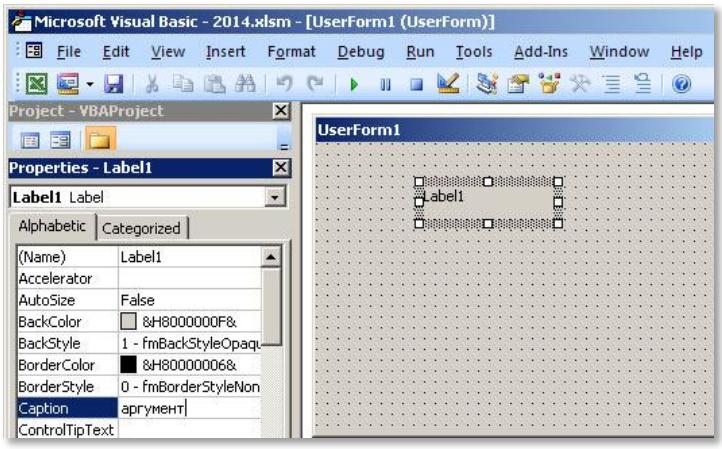
Рассмотрим работу с формой. Переключитесь в редактор кода VBE и в меню Insert выберите опцию UserForm. В окне проекта добавится раздел Forms со списком доступных форм, вновь созданная форма будет сразу же выделена и доступна для редактирования, станет видна панель с элементами управления Toolbox.



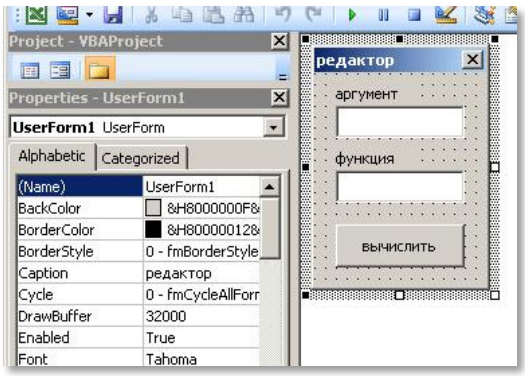


Переключаться между формой и листами можно двойным кликом мыши в окне проекта. Само окно проекта можно включить или в него перейти сочетанием клавиш Ctrl+R или в меню View опция Project Explorer.

Исследуем возможности некоторых компонентов. Самый простой компонент – метка Label. Метка хранит и отображает текст, который не доступен для редактирования пользователем. Чтобы разместить компонент на форме кликните по нему один раз в панели инструментов и затем кликните в том месте на форме, где желаете его разместить. Компоненты отличаются свойствами. Список доступный свойств можно увидеть в специальном окне, кликнув правой клавишей мыши по компоненту и в контекстном меню, выбрав опцию Properties.



Окно свойств компонента разбито на два столбца: левый содержит название свойства (поле), правый – его значение. Многие свойства интуитивно понятны. Поле Caption содержит отображаемый на метке текст. Его, как и значения других полей, можно изменить: кликните по значению мышкой и исправьте текст, затем нажмите Enter. Поле Caption есть у многих компонентов, в том числе и на самой форме. Разместите на форме также пару компонентов TextBox (рамка с текстом, доступным для редактирования пользователем) и добейтесь примерно такого внешнего вида:



Наша цель такова: форма будет получать значение аргумента (число) из ячейки A1 с листа Excel и по нажатию на экранную клавишу на самой форме (“вычислить”) будет размещать значение квадрата этого числа во втором TextBox`е и на листе Excel в ячейке A1.

Нужно добиться того, чтобы форма отобразилась (запустилась программа). Можно использовать различные способы запуска формы, но сейчас мы воспользуемся несложным. Разместите на листе Excel кнопку ComandButton1 через вкладку Разработчик, Вставить, Элементы управления ActiveX. Сгенерируйте процедуру CommandButton1_Click(), кликнув дважды мышкой по клавише, и заполните следующим кодом:

```
Private Sub CommandButton1_Click()
  UserForm1.TextBox1.Text = Cells(1, 1)
  UserForm1.Show
End Sub
```

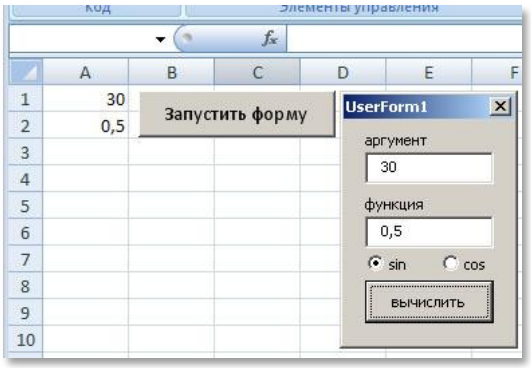
Апробируйте работу процедуры и запуск формы. Форма должна запускаться, но пока ещё не функциональна – нужно добавить обработчик события к клавише на самой форме. Закройте форму, в режиме редактора кликните по кнопке формы дважды – сгенерируется пустая процедура с тем же именем, как и процедура для кнопки листа (так как мы оставили имена кнопок данные им по умолчанию). Заполните процедуру для кнопки формы:

```
Private Sub CommandButton1_Click()
  x = TextBox1.Text
  y = x ^ 2
  TextBox2.Text = y
  Cells(1, 1) = y
End Sub
```

Апробируйте работу формы. Итак, данные с листа можно передавать на форму, там обрабатывать и возвращать обратно.

Рассмотрим ещё несколько наиболее часто используемых компонента.

Добавим нашей форме функционала: теперь на форме можно будет выбирать вычисляемую функцию (sin или cos). Попутно исследуем возможности переключателя – OptionButton. Добавьте на форму два компонента OptionButton, измените их Caption на sin и cos. По умолчанию они оба выключены, во время работы формы они функционируют совместно – включение одного приводит к выключению другого. Для корректной работы нашей формы имеет смысл заранее включить один из OptionButton, для чего переведите свойство Value из состояния False в True.



Измените программный код кнопки «вычислить» следующим образом:

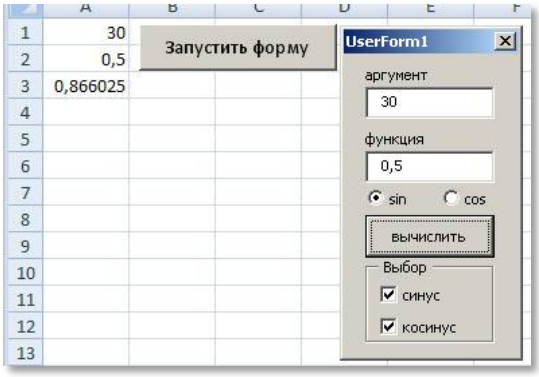
```
Private Sub CommandButton1_Click()
  g = TextBox1.Text
  r = g * Application.Pi / 180
  If OptionButton1.Value = True _
    Then y = Sin(r) _
    Else y = Cos(r)
  TextBox2.Text = y
  Cells(2, 1) = y
End Sub
```

В первой строке мы берем значение из `TextBox1` – это значение угла в градусах. Во второй строке переводим его в радианы, так как тригонометрические функции работают именно с радианами. Далее условным оператором выбираем вид вычисляемой функции. Результат выводим в `TextBox2` и на лист Excel.

Обратите внимание, что в момент запуска формы значение в `TextBox1` берется с листа Excel, но, когда форма уже запущена, вы можете сами набирать/исправлять значение в градусах в `TextBox1` и нажимать клавишу «вычислить» столько раз сколько вам нужно.

Компонент OptionButton, как переключатель, позволяет организовать альтернативные пути работы программы, то есть вычисления пойдут либо по одному, либо по другому пути. Но бывает необходимость организовать работу так, чтобы был выбор либо по одному пути, либо по другому пути, либо оба пути выполняются, либо оба не выполняются. Для данной цели используют компонент CheckBox.

Пусть наша форма будет обладать возможность выводить на лист Excel либо только синус, либо только косинус, либо оба значения сразу, либо ничего не выводить на лист. Разместите на форме компонент Frame (рамка). Внутри неё расположите два компонента CheckBox, поля Caption которых исправьте на «синус» и «косинус». В отличие от OptionButton компоненты CheckBox могут быть активированы независимо друг от друга. Можете запустить фурму и проверить как устанавливаются и убираются «галочки» в CheckBox.



Функционал, который нужно добавить в программный код, такой: при нажатии на кнопку «вычислить», в поле TextBox2 заносится значение функции, выбранной компонентами OptionButton, а на лист Excel размещаются значения функций, указанных компонентами CheckBox (может и ни одно, может одно из двух, а может и оба).

Для реализации указанной задачи измените программный код следующим образом:

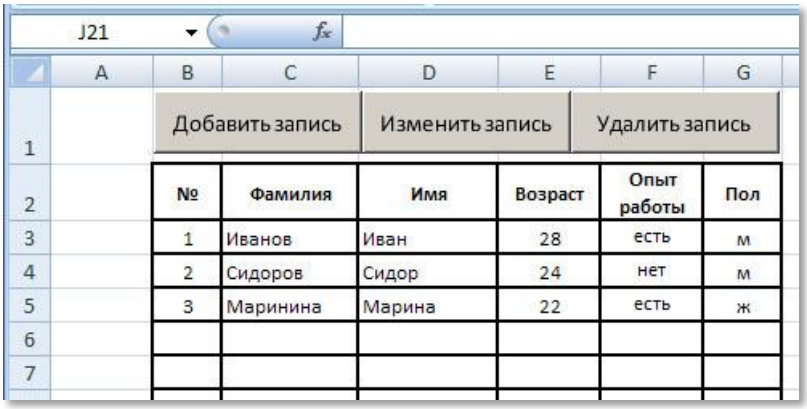
```
Private Sub CommandButton1_Click()
  g = TextBox1.Text
  r = g * Application.Pi / 180
  If OptionButton1.Value = True _
    Then y = Sin(r) _
    Else y = Cos(r)
  TextBox2.Text = y
  Cells(2, 1) = "": Cells(3, 1) = ""
  If CheckBox1.Value = True Then Cells(2, 1) = Sin(r)
  If CheckBox2.Value = True Then Cells(3, 1) = Cos(r)
End Sub
```

Начинается программный код, как и ранее, далее ячейки A2 и A3 листа Excel стираются, так как возможно ранее там были значения. Затем, в зависимости от выбора пользователя, туда могут быть выведены значения функций.

Задание для самостоятельного исполнения.

Форма и элементы управления.

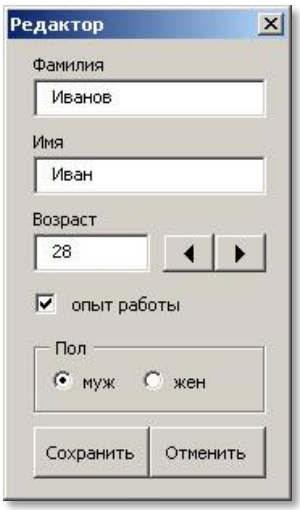
На листе книги Excel находится таблица:



№	Фамилия	Имя	Возраст	Опыт работы	Пол
1	Иванов	Иван	28	есть	м
2	Сидоров	Сидор	24	нет	м
3	Маринина	Марина	22	есть	ж

К таблице прилагается меню, состоящее из трёх пунктов: Добавить запись, Изменить запись, Удалить запись. Пункты меню – это кнопки CommandButton. Следует «Закрепить области» (на ленте вкладка «Вид» / пункт «Закрепить области») так чтобы при прокручивании таблицы вниз строки с кнопками меню и заголовком таблицы оставались на месте, а прокручивались только остальные строки с данными (с первой и далее…).

Разработать форму Редактора записей таблицы.



На форме доступны для редактирования поля согласно структуры таблицы: Фамилия, Имя, Возраст, Опыт работы, Пол. Номер по порядку формируется автоматически. Для остальных полей есть соответствующие элементы управления: Фамилия и Имя – `TextBox`, Возраст – `TextBox` и `SpinButton` (прокручивание элемента приводит к изменению возраста в TextBox), Опыт работы – `CheckBox` (наличие галочки означает наличие опыта работы), Пол – `OptionButton`.

На форме Редактора должны быть две функциональные клавиши – «Сохранить» и «Отменить». При нажатии на клавишу «Сохранить» все данные переносятся в таблицу на листе книги Excel и форма Редактора закрывается. При нажатии на клавишу «Отменить» просто закрывается форма Редактора.

При нажатии на пункт меню «Изменить запись» соответствующие поля с листа книги Excel переносятся в соответствующие элементы управления Формы редактора. После их редактирования клавишей «Сохранить» изменения переносятся обратно на лист Excel в соответствующие ячейки.

Если был выбран пункт меню «Добавить запись», форма редактора отображается с пустыми полями. Если впоследствии пользователь нажал клавишу «Сохранить», то данные вносятся сразу после последней существующей записи в таблице на листе Excel.

Если был выбран пункт меню «Удалить запись», то текущая строка из таблицы удаляется, а содержимое нижележащих строк сдвигается на одну позицию выше.

Вам пригодится метод закрытия формы `Unload`. Пример для клавиши «Отменить»:

```
Private Sub CommandButton2_Click()
  Unload UserForm1
End Sub
```

Также вам понадобится способ получения текущей позиции курсора (для реализации функционала «Изменить запись» и «Удалить запись») на листе Excel, например:

`R = ActiveCell.Row` – в переменную R запишется номер текущей строки,

`C = ActiveCell.Column` – в переменную R запишется номер текущего столбца.

Приложение 1.

Структурные операторы VBA.

Операторы ветвления.

Примеры оформления:

```
If <условие> Then <действие1> Else <действие2>

If <условие> Then <действие1> : <действие2> Else <действие3> : <действие4>.
```

```
If условие Then
      <действие1> : <действие2>
Else
      <действие3> : <действие4>
End If
```

```
If условие Then
      <действие1>
      <действие2>
Else
      <действие3>
      <действие3>
      <действие4>
End If
```

```
Select Case <проверяемое выражение>
  Case <значение1>
            <действие1>
  Case <значение2>
            <действие2>
End Select
```

```
Select Case <проверяемое выражение>
  Case <значение1>
            <действие1>
  Case <значение2>
            <действие2>
  Case Else
            <действие3>
End Select
```

```
Select Case <проверяемое выражение>
  Case 2012
                  <действие1>
  Case 2021, 2029
                  <действие2>
  Case 2000 To 2010
                  <действие3>
  Case Is>2029
                  <действие4>
End Select
```

Операторы цикла.

Примеры оформления программы поиска суммы чисел от 1 до 10:

```
For i = 1 To 10
      sum = sum + i
Next
MsgBox "Sum=" & Sum
```

```
For i = 10 To 1 Step -1
      sum = sum + i
Next
MsgBox "Sum=" & Sum
```

```
i = 0
Do
      i = i + 1
      sum = sum + i
Loop Until i >= 10
```

```
i = 0
While i < 10
      i = i + 1
      sum = sum + i
Wend
```

Приложение 2.

Функции VBA для обработки строк.

Dim S As String – объявление строковой переменной.
Dim S(12) As String – объявление строковой переменной фиксированной длинны.

Функция	Действие
Asc(S)	Возвращает число кода символа, соответствующее первой букве строки S.
Chr(N)	Возвращает строку из одного символа, соответствующего коду символа N, который должен быть числом между 0 и 255 (обратная предыдущей).
Format(E,S)	Возвращает строку, содержащую значение, представленное выражением E, в формате в соответствии с инструкциями, содержащимся в S.
Hex(N)	Возвращает строку, содержащую шестнадцатеричное представление N.
Oct(N)	Возвращает строку, содержащую восьмеричное представление N.
Str(N)	Возвращает строку, эквивалентную численному выражению N.
Val(S)	Возвращает численное значение, соответствующее числу, представленному строкой S, которая должна содержать только цифры и одну десятичную точку, иначе возвращается 0.
Len(S)	Определяет длину строки S.
Left(S, K)	Выделяет из строки S указанное K символов слева.
Right(S, K)	Выделяет из строки S указанное K символов справа.
Mid(S, N [,K])	Выделяет из аргумента S подстроку с указанным числом символов K, начиная с позиции N.
Mid(S, N)	Выделяет подстроку от позицииN до конца строки S.
LTrim(S)	Удаляет пробелы в начале строки S.
RTrim(S)	Удаляет пробелы в конце строки S.
Trim(S)	Удаляет пробелы в начале и в конце строки S.
InStr([N,] S, F [,P])	Производит поиск подстроки в строке. Возвращает позицию первого вхождения строки F в строку S; N – позиция, с которой начинается поиск. Если этот аргумент пропущен, поиск начинается с начала строки. P – параметр сравнения (0 – бинарное, 1 – строковое).
InStrRev([N,] S, F [,P])	Аналогично, но начинает поиск с конца строки и возвращает позицию последнего вхождения подстроки.
Replace (S, F, G)	Заменяет в строке S подстроку F подстрокой G (все вхождения).
Space(K)	Создает строку, состоящую из K пробелов.
String(K,C)	Создает строку, состоящую из K символов C.

Split(S [, R])	Преобразует строку S в массив подстрок. По умолчанию в качестве разделителя используется пробел. Удобно использовать для разбиения предложения на слова:
	`Dim st() As String`
	`st = Split("Это тестовое предложение", " ")`
	`Cells(17, 4) = st(2)`

Оператор сравнения строк Like позволяет обнаруживать неточное совпадение.
Пример кода: `X = «Входной сигнал»: B = X Like «Вход*»`
Результат операции сравнения (X Like «Вход*») – ИСТИНА будет присвоен переменной B.

* – любое количество любых символов
? – любой один символ
#– одна цифра (0–9)
[<список>] – символ, совпадающий с одним из символов списка;
[!<список>] – символ, не совпадающий ни с одним из символов списка.

Массивы VBA.

Массив – проиндексированная последовательность данных одного типа. К элементам массива обращаются по их индексам. Например, X = M(1) – в переменную X присваивается первый элемент массива M.

Статические массивы.

Dim Mas(3) As String – объявление статического массива (3 элемента в массиве, их индексы – 0, 1, 2).

Директива: Option Base 1 – предварительное описание первого индекса (индексы массивов будут начинаться с единицы, директиву ставить до процедуры / см. пример ниже).

Dim Mas(1 To 3) As String – явное указание границ.

Dim Mas(1 To 3) – тип данных указывать необязательно.
Dim D(5, 2) As Double – двумерный массив с индексами от 0.
Dim D(1 To 5, 1 To 2) As Double – двумерный массив с указанием нижней границы.

Функции определения границ массива:
LBound – нижняя граница;
UBound – верхняя граница;
Lbound(A) – пример для одномерного массива;
Lbound(A,2) – пример для двумерного массива.

Функция генерации массива.
Array – эта функция создает массив динамически (в ходе выполнения программы) без предварительного описания.
Пример: Mas = Array(5, 2, 14).

Динамические массивы.
Dim A() As Integer – объявление динамического массива (верхняя граница изменяемая).
Изменение границ массива:
ReDim Preserve A(1 To 3) – с сохранением элементов;
ReDim A(1 To 3) – без сохранения элементов.

Пример программы заполнения элементов массива случайными числами и вывода их в ячейки на листе Excel:

```
Private Sub CommandButton1_Click()
  Dim Mas(1 To 10) As Integer
  For i = 1 To 10
    Mas(i) = Int(10 * Rnd)
    Cells(i, 1) = Mas(i)
  Next
```

Применение директивы `Option Base`:

```
Option Base 1
Private Sub CommandButton1_Click()
  Dim Mas(10) As Integer
  For i = 1 To 10
    Mas(i) = Int(10 * Rnd)
    Cells(i, 1) = Mas(i)
  Next
End Sub
```