Министерство сельского хозяйства Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Казанский государственный аграрный университет»

Кафедра физики и математики

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ VBA В EXCEL

Учебное пособие

Казань, 2020

УДК 51(07)
ББК 22.3Р

Ибятов Р.И., Валиев А.А., Газизов Е.Р.
«Программирование на языке VBA в EXCEL»: Учебное пособие / Р.И. Ибятов, А.А. Валиев, Е.Р. Газизов – Казань: Изд-во Казанского ГАУ, 2020. – 60с.

Рецензенты:
кандидат технических наук, доцент Дмитриев А.В.
кандидат технических наук, доцент Горская Т.Ю.

Печатается по решению Методического Совета Казанского ГАУ (протокол № от 2020 г.)

Учебное пособие предназначено для выполнения лабораторных и самостоятельных работ по дисциплине «Информатика» для направлений подготовки: 23.03.03 - «Эксплуатация транспортно-технологических машин и комплексов», 20.03.01 - «Техносферная безопасность», 23.05.01 - «Наземные транспортно-технологические средства»; по дисциплине «Информатика и цифровые технологии» для направления подготовки 35.03.06 «Агроинженерия»; по дисциплине «Технологии работы с информацией» для направления подготовки 44.03.04 - «Профессиональное обучение», студентами очной и заочной форм обучения. В учебном пособии представлены основные операторы для программирования в Visual Basic for Application (VBA) и приведены решения типовых задач.

УДК 51(07)
ББК 22.3Р

ВВЕДЕНИЕ

В 1962…1964 годах в Дартмутском колледже был разработан язык программирования Basic. Он быстро завоевал популярность в качестве языка для обучения программированию в университетах и школах. В середине 70-х годах Basic был адаптирован для использования на персональных компьютерах основателем и главой компании Microsoft Биллом Гейтсом. С тех пор для ПК последовательно было выпущено ряд версий Basic, включая GW-Basic, Turbo Basic, QuickBasic.

В начале 90-х появляется операционная система Microsoft Windows с новым графическим интерфейсом пользователя. В это же время в недрах Microsoft велось несколько параллельных проектов по созданию нового языка программирования для Windows. И в 1991 году под лозунгом «теперь и начинающие программисты могут легко создавать приложения для Windows», появилась первая версия нового инструментального средства Microsoft Visual Basic.

После разработки приложений для Windows объединенных в пакет MS Office, возникла проблема их интеграции, так как все они предназначались для работы с документами и автоматизации вычислений и расчетов разного характера. Во всех приложениях пакета MS Office повторяющиеся действия могут быть заменены последовательностью машинных команд – макроопределений или макросов. Язык Visual Basic стал использоваться как основной для разработки макросов. Появились его версии: Access Basic, Word Basic, Excel Basic, и т.д. Для последующих версий пакета Microsoft Office они были объединены и адаптированы как единый язык программирования для всех приложений.

Эта версия языка носит название Visual Basic for Application или VBA (Visual Basic для приложений). VBA включает в себя основные конструкции языка Visual Basic и является общим для всех приложений MS Office. Благодаря этому не требуется изучать каждый раз язык программирования для автоматизации некоторой задачи в другом приложении MS Office.

Одним из достоинств языка является легкость его освоения. Для того чтобы освоить основы языка VBA нет необходимости углубленных знаний по программированию. Он позволяет быстро получить ощутимые результаты – конструировать профессиональные приложения для решения практически любых задач в среде Microsoft Windows.

Электронные таблицы MS Excel разрабатывались для упрощения обработки больших объемов числовых данных, для выполнения рутинных операций и наглядного представления результатов вычислений. Макросы в MS Excel это программы, написанные на языке VBA, поэтому программирование в табличном процессоре сводятся к созданию макроса с нужным кодом.

Здесь мы рассмотрим VBA, встроенную в Excel. Это позволит нам достаточно просто использовать таблицу Excel для ввода данных и вывода результата.

1. СОЗДАНИЕ И ЗАПИСЬ МАКРОСА

Макрос – это список инструкций, которые могут выполняться автоматически без участия пользователя. Обычно они создаются для автоматизации часто повторяющихся одинаковых задач. Другими словами, макрос – это вид VBA-программы, а VBA – встроенный язык программирования Excel.

Существует два способа создания макроса:

1. Автоматически записать последовательность необходимых действий с помощью макрорекордера. Макрорекордер – это встроенный в Excel инструмент, который может отслеживать выполнение пользователем задачи и записывать порядок работы в виде списка инструкций. При помощи макрорекордера пользователь может создавать макросы без знания языка программирования VBA.

2. Вручную ввести необходимые инструкции на языке программирования VBA. Естественно, в данном случае знание языка программирования необходимо.

Задание 1.1. Создайте при помощи макрорекордера макрос, который вычисляет сумму чисел 100 и 200.

Порядок выполнения работы

1. Открыть Microsoft Excel и сохранить файл в собственной папке используя команды **Сохранить как/ Книга Excel с поддержкой макросов.**

2. В среде Microsoft Excel в строке меню открыть вкладку **Вид/ Макросы/ Запись макроса**. В открывшемся окне Запись макроса в поле Имя макроса введите имя Prim1. Для запуска макроса с помощью комбинаций клавиш в поле

Сочетание клавиш наберите букву, например, z и нажмите кнопку Ok (рис. 1.1).

3. Далее выполняете действия, которые вам нужно записать в макрос.

- В ячейку A1 введите число 100, а в ячейку A2 – число 200.
- В ячейку B2 наберите формулу =A1+A2.

4. Щёлкните на кнопке "Остановить запись" (рис. 1.2).

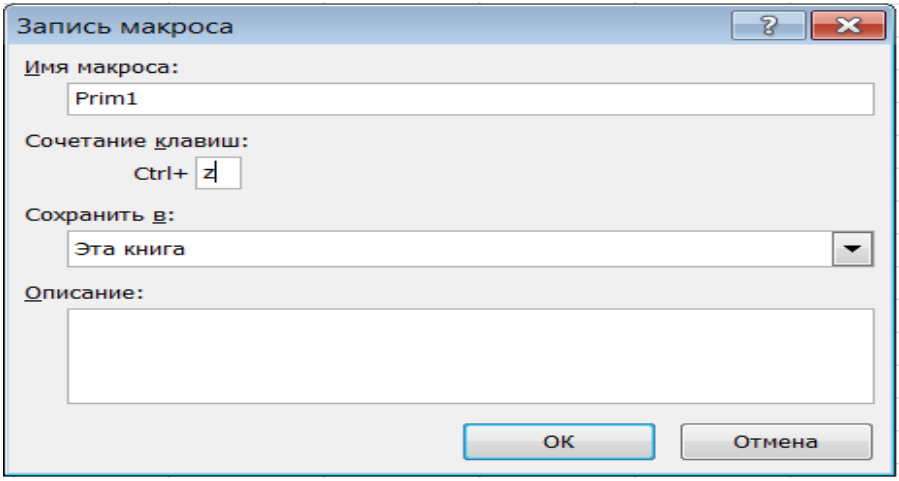


Рис. 1.1. Окно записи макроса.

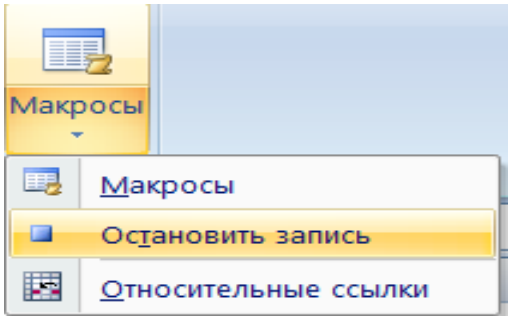


Рис. 1.2. Остановка записи макроса.

Сформированный макрос может быть использован в любом месте данной рабочей книги, на любом из его листов. Чтобы макрос повторил все записанные в него действия необходимо его запустить в работу.

Для этого необходимо выполнить следующие действия:

1. Перейдите на новый рабочий лист вашей книги.

2. Выберите **Вид/ Макросы**. Должно появиться диалоговое окно, где в списке макросов выделите сформированный ранее макрос с именем Prim1.

3. Нажмите на кнопку Выполнить. В результате вы увидете действия, проделанные вами во время записи макроса.

Для запуска макроса можно использовать комбинацию клавиш Ctrl+z, которая была назначена при создании макроса (рис. 1.1).

4. Перейдите на новый рабочий лист и удалите все записи. Нажмите на комбинацию клавиш Ctrl+ z.

Получаем тот же результат, который дает записанный макрос, но на другом, активном листе Книги.

Код макроса, который был записан в результате проделной работы, можно посмотреть. Для этого выполните следующие действия:

1 способ:

- открыть вкладку **Вид/ Макросы/ Макросы** и нажмите на кнопку Войти или Изменить. Откроется окно редактора VBA (рисунок 1.2). Код макроса находится между строками Sub Prim1 ( ) и End Sub.

2 способ:

- установите курсор на ячейки рабочего листа и нажмите на комбинацию клавиш Alt+F11.

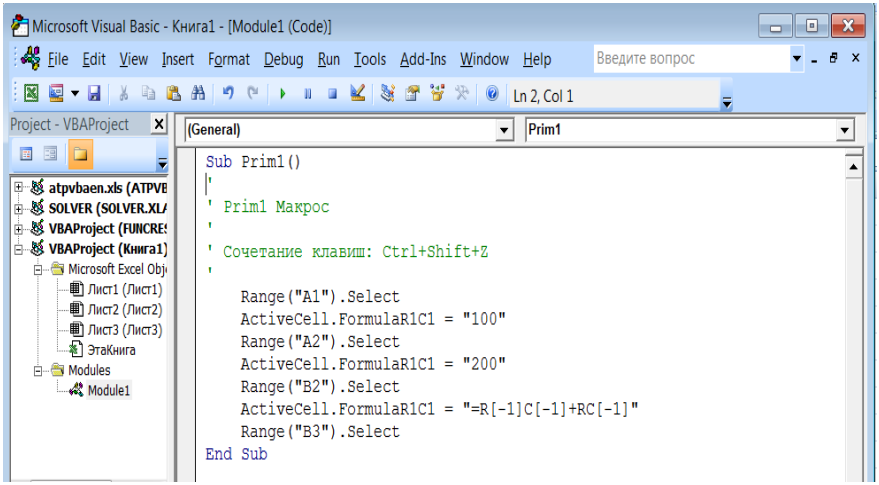


Рис. 1.3. Окно редактора VBA.

Первая строчка включает имя макроса, служебное слово **Sub** и круглые скобки (). Это стандартный заголовок макроса. Стандартной завершающей строкой макроса является строка **End Sub**. Эти две строчки появятся даже в том случае, если создать пустой макрос. Служебные слова VBA выделены синим цветом. Далее следуют строчки, выделенные зеленым цветом и содержащие описание макроса. Они имеют в первой позиции знак апострофа (‘) и являются комментариями. Они служат для разъяснения действий, записанных в макросе, не изменяют работу макроса и могут быть поставлены в любое место тела макроса. Следующие семь строк черного цвета – это инструкции (команды) языка VBA. Они описывают действия, выполняемые макросом.

Как видим, записанный макрос является завершенной и работоспособной программой. При необходимости его можно редактировать.

Задание 1.2. Используя коды Prim1, создайте новый макрос, который вычисляет площадь треугольника с основанием 15 и с высотой 4 единицы. При этом макрос Prim1 оставьте без изменений.

Порядок выполнения работы

1 .Скопируйте полностью коды макроса Prim1 и вставьте ниже команды End Sub. Название процедуры смените на Prim2.

2. В тексте макроса числа 100 и 200 меняем на 15 и 4 соответственно. Формулу =A1+A2 меняем на =A1*A2/2 (рис. 1.4).

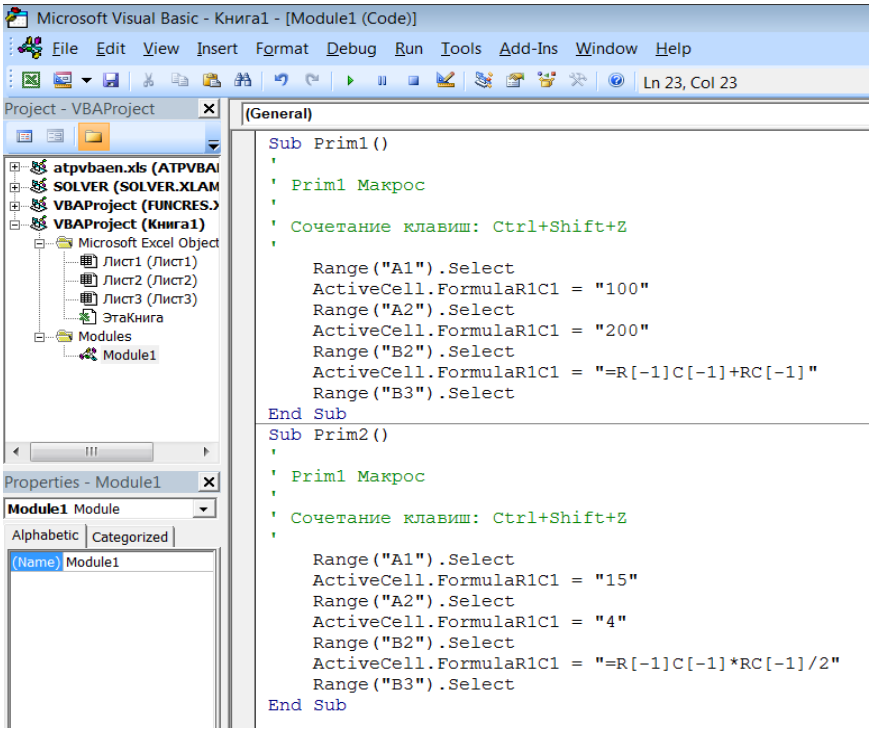


Рис. 1.4. Окно редактора VBA.

Новый макрос Prim2 создан и готов для использования. Когда открыто окно редактора VBA, для выполнения макроса могут быть использованы следующие команды:

- нажать на зеленый треугольник на панели инструментов;
- нажать на кнопку F5;
- использовать команды Run/ Run Sub.

При этом курсор должен находиться в области того макроса, которого намериваем запускать.

Для просмотра полученного результата необходимо вернуться на рабочий лист электронной таблицы с помощью нажатия на комбинацию клавиш Alt+F11 или пиктограмму EXCEL на панели инструментов.

3. Одним из указанных способов выполняйте макрос Prim2 и посмотрите на результат.

4. Перейдите на новый рабочий лист и запустите макрос Prim2 через вкладки **Вид/ Макросы/ Макросы.** Посмотрите на результат.

5**.** Сохраните созданий вами файл. Если файл сохраняете впервые или сохраняете в другой папке с помощью команды **Сохранить как,** не забудьте выбрать тип файла **Книга Excel с поддержкой макросов.**

Задание 1.3. Постройте график функции $y = x^2 - 20$ на интервале от - 5 до 10.

Порядок выполнения работы

1. Откройте вкладку **Вид/ Макросы/ Запись макроса**. Назначьте макросу имя Prim3. Для запуска макроса выберите комбинацию клавиш Ctrl+x,. В поле Описание наберите текст

«Построение графика функции y=x*x-20 на интервале от -5 до 10» и нажмите кнопку Ok.

2. Далее выполняйте действия, которые нужно записать в макрос.

- В ячейку A1 введите число -5. В ячейку A2 наберите формулу =A1+1 и копируйте ее в ячейки A3:A16.
- В ячейку B1 введите формулу =A1*A1-20 и копируйте ее в ячейки B2:B16.
- Постройте диаграмму с помощью вкладки **Вставка/ Точечная/ Выбрать данные**. В открывшемся окне Выбор источника данных в поле Диапазон данных для диаграммы укажите диапазон A1:B16 (рис. 1.5). Для этого достаточно на рабочем листе мышкой выделить нужный диапазон. Нажмите кнопку Ok. График функции построен (рис. 1.6).

3. Щёлкните на кнопке "Остановить запись".

4. Удалите все записи. Нажмите на комбинацию клавиш Ctrl+x.

5. Перейдите на новый рабочий лист и нажмите на комбинацию клавиш Ctrl+x.

Получаем тот же результат, который дает записанный макрос. Он доступен от любого рабочего листа Книги.

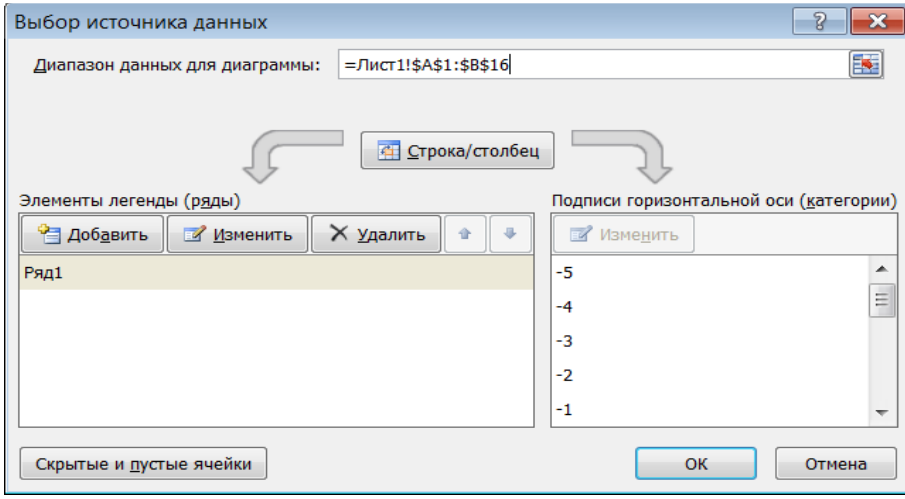
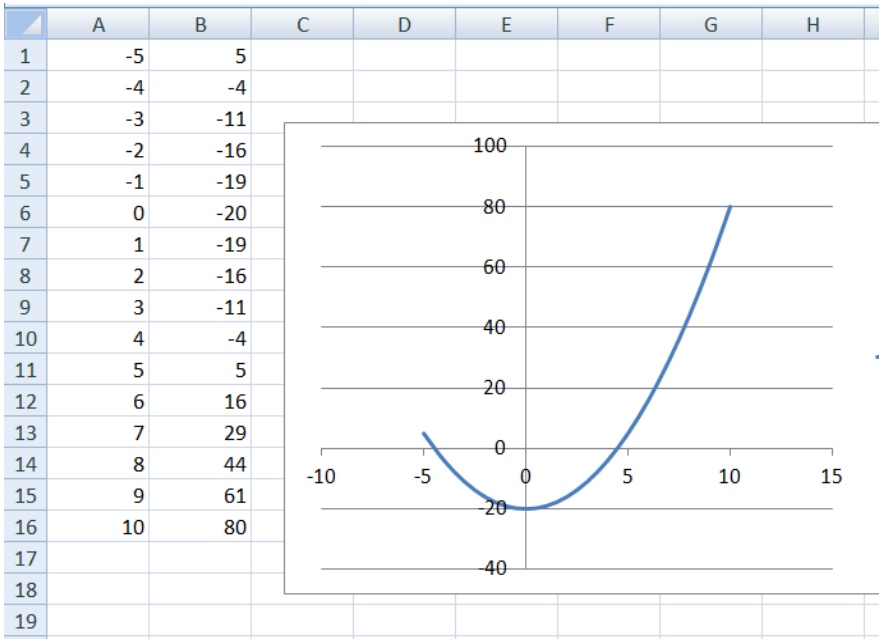


Рис. 1.5. Окно выбора источника данных.



	A	B
1	-5	5
2	-4	-4
3	-3	-11
4	-2	-16
5	-1	-19
6	0	-20
7	1	-19
8	2	-16
9	3	-11
10	4	-4
11	5	5
12	6	16
13	7	29
14	8	44
15	9	61
16	10	80
17		
18		
19		

Рис. 1.6. График функции

Для просмотра кода инструкций нужно открыть вкладку **Вид/ Макросы/ Макросы** нужно щелкнуть на кнопке Войти (рис. 1.7). Как видим, вся информация, введенная в окне Запись макроса, содержится в кодах инструкций макроса в виде комментариев. В комментариях приведены имя и назначение макроса, а также указано сочетание клавиш для его запуска.

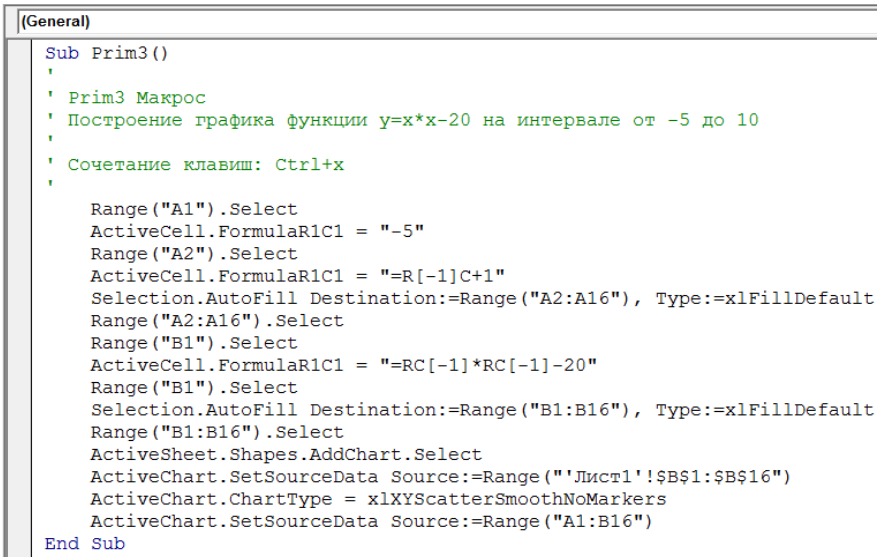


Рис. 1.7. Код инструкций макроса.

Задание 1.4. Требуется создать макрос для решения системы линейных уравнений.

Решение. Систему линейных уравнений можно записать в виде

$$AX = B,$$

где A - квадратная матрица, содержащая значения коэффициентов левой части, B - вектором правых частей, X –

вектор неизвестных. Тогда систему уравнений можно решить методом обратной матрицы

$$X = A^{-1}B\,.$$

Все необходимые инструменты для реализации данного метода в электронной таблице EXCEL имеются. Математическая функция **МУМНОЖ(массив1;массив2)** возвращает матричное произведение двух массивов. Обратную матрицу можно находить с помощью функции **МОБР(массив**). Поэтому решения системы линейных уравнений можно получить с помощью формулы

=МУМНОЖ(МОБР(массивА); массивВ).

Отметим, что данную формулу необходимо ввести как формулу массива. Выделите диапазон для решения системы, наберите формулу МУМНОЖ(МОБР(массивА); массивВ), затем нажмите клавиши Ctrl+Shift+Ввод.

Пусть для системы из трех линейных уравнений числовые значения коэффициентов элементов матрицы А и вектора В заблаговременно внесены в ячейки электронной таблицы (рис. 1.8).

	A	B	C	D	E
1		A			B
2	3	4	9		41
3	1	8	6		20
4	5	2	7		58
5					

Рис. 1.8. Коэффициенты системы уравнений.

Порядок выполнения работы

1. Откройте вкладку **Вид/ Макросы/ Запись макроса**. Назначьте макросу имя Prim4 и комбинацию клавиш Ctrl+c для запуска. В поле Описание наберите текст «Решение системы уравнений».

2. Выполняйте действия, которые нужны для решения системы уравнений:

- выделите ячейки G2:G4 и наберите формулу =МУМНОЖ(МОБР(A2:C4);E2:E4);
- нажмите клавиши Ctrl+Shift+Ввод.

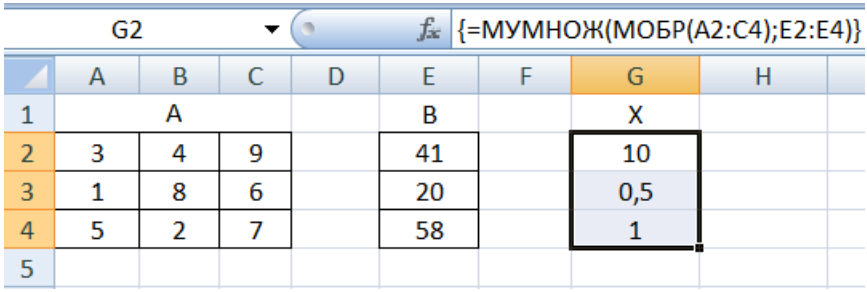
В ячейках G2:G4 появится решение системы уравнений (рис. 1.9).

3. Щёлкните на кнопке "Остановить запись".

4. Очистите ячейки G2:G4 и нажмите на комбинацию клавиш Ctrl+c. Решение системы уравнений появится заново.

5. В ячейках A2:C4 или (и) E2:E4 поменяйте значения, затем нажмите на клавишу Ввод. Как видите, в данном случае новая система уравнений решалась без участия макроса.

Через вкладки **Вид/ Макросы/ Макросы** и кнопки Войти посмотрите на листинг макроса (рис. 1.10). Как видите, макрос состоит из двух инструкций – выбираются ячейки G2:G4 для вывода решения и запускаются внутренние матричные функции МУМНОЖ(МОБР(A2:C4);E2:E4) электроной таблицы. В последнем случае эти функции были запущены клавишей Ввод минуя макроса Prim4.



	A	B	C	D	E	F	G	H
1		A			B		X	
2	3	4	9		41		10	
3	1	8	6		20		0,5	
4	5	2	7		58		1	
5								

Рис. 1.9. Решение системы уравнений.

```
Sub Prim4()
'
' Prim4 Макрос
' Решение системы уравнений
'
' Сочетание клавиш: Ctrl+c
'
    Range("G2:G4").Select
    Selection.FormulaArray = "=MMULT(MINVERSE(RC[-6]:R[2]C[-4]),RC[-2]:R[2]C[-2])"
End Sub
```

Рис. 1.10. Листинг макроса Prim4.

Контрольные вопросы:

1. Дать определение макросу.
2. Описать процесс создания макроса.
3. Описать процесс запуска макроса в работу.

2. ВВОД И ВЫВОД ИНФОРМАЦИИ. ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ЛИНЕЙНОЙ СТРУКТУРЫ

Для написания новых программ или внесения изменений в код инструкции необходимо знание языка программирования VBA. Нужно знать элементы языка, инструкции, их синтаксис и правила составления кода программы. Далее рассмотрим отдельные аспекты, которые будут необходимы для создания несложных программ.

Структура программы

Исходный текст программы состоит из последовательности строк, каждая из которых может начаться с любой позиции. Программа может иметь один из двух видов:

1. В виде процедуры

```
Sub  Name()
    …
  <операторы >
    …
End Sub
```

2. В виде функции

```
Function  Имя()
    …
  <операторы>
    …
End Function
```

Отличия этих вариантов оформления программ проявляются при использовании подпрограмм и будут обсуждены в лабораторной работе №6.

Name представляет собой имя программы, состоящее из букв и цифр. Скобки после имени оставляются пустыми. Имя может быть составлена с помощью букв алфавита русского языка.

Операторы программы могут быть записаны в столбец или в строку и выполняются поочерёдно начиная с первого оператора. Если в строку записаны несколько операторов (ин-

струкций), то они разделяются двоеточием. Оператор можно переносить с одной строки на другую при помощи знака переноса "_" (подчёркивание), который остаётся в верхней строке. Перед ним ставится пробел. Для обеспечения понятности, в текст программы помещаются комментарии. Они начинаются либо со знака « ' » (апостроф), либо с ключового слова Rem:

```
Rem  Решение квадратного уравнения с коэффициентами A, B, C
A=1
B=20: C=100: kopen1=(-B+sqr(B^2-4*A*C))/(2*a): kopen1=(-B-sqr _
(B^2-4*A*C))/(2*a)      ‘ Это продолжение предыдущей строки
```

При составлении программы переменным, константам, функциям и процедурам задаётся имена (идентификаторы). VBA накладывает на имена следующие ограничения:

- имя должна начинаться с буквы
- длина имени не должна превышать 255 символов;
- имя не должно содержать точек, пробелов разделительных символов, знаков операций и специальных: %, $, !, #, $;
- имена не должны повторяться в пределах кода процедуры;
- не следует использовать имена, совпадающие с зарезервированными словами языка VBA (именами операторов, встроенных процедур и функций).

Могут использоваться буквы как латинские, таки русские, в том числе и вперемешку. Строчные и прописные буквы в идентификаторах VBA не различаются.

Объявление переменных и констант

Переменные – это объекты, предназначенные для временного хранения данных. Содержимое переменной может быть изменено в процессе выполнения программы. Данные, хранимые в переменных, могут быть различного типа: числовые данные, строковые, логические и т.д. Приведем основные типы.

Integer (целый) принимает целые значения в пределах от -32788 до 32788. Занимает 2 байта.

Single (вещественный) используется для обозначения вещественных чисел по абсолютной величине принадлежащих промежутку от $3.4*10^{-38}$ до $3.4*10^{38}$. Имеют 6-7 значащих цифр. Занимает 4 байта.

String (строковый) обозначает строки, состоящие из последовательности символов, заключенную в кавычки. Длина строки до 65 тыс. символов. Один символ занимает 1 байт. Например, "сегодня у нас информатика!", "я - студент".

Date/Time (дата/время) используется для обозначения даты или времени. Дата должна находиться в пределах от 1 января 100 года до 31 декабря 9999 года. Занимает 8 байт. Для обозначения дат используется символ #, после которого записывается дата в американском стиле, например, #12/31/2010#. Можно дату записать и так, #2010-12-31#.

Boolean (логический) предназначен для логических величин. Имеет два значения True и False. Они обозначают соответственно истина и ложь. Занимает 2 байта.

Variant (общий) может использоваться для всех типов данных. Занимает 16 байт.

Переменные в программе нужно объявлять. При объявлении переменных указывается, какой тип данных в ней хранится. Декларация переменных может быть явной или неявной. Явная декларация выполняется при помощи оператора объявления переменной:

Dim VarName1 **As** Type1, VarName2 **As** Type2**,..,** и т. д.

где VarNameN - имя переменной, а TypeN - тип её данных. Оператор Dim можно располагать в любом месте программы, но до первого использования объявляемых переменных. Пример объявления переменных:

```
Dim g As Single, h As Single, j As Integer
Dim i As Long
```

Комбинацию As <тип> необходимо указывать для каждой переменной. Если переменная не описана или не указан тип, то получим переменную типа Variant. Переменные этого типа занимают больше места, и программа с ними работает медленнее. Например:

Dim x As Single, y, z As Integer

Получим, что x типа Single, y типа Variant, z типа Integer.

При описании переменной ей автоматически присваивается нулевое значение данного типа. Если это числовая переменная, то она получит значение нуль. Если же логическая переменная, то значение False. Если строковая, то в качестве

начального значения будет пустая строка, т.е. строка, не содержащая никаких символов. Далее это значение можно менять соответствующими операторами, например, присваивания.

Неявная декларация переменных выполняется с помощью символов описания. Такой способ описания используются для более краткого объявления переменных без оператора Dim. Они добавляются в конце имени переменной, что позволяет сразу задать её тип. Целый тип объявляется символом описания – %, вещественный тип – символом !, строковый тип – через $.

Например,

K% = 2018 - переменная целого типа;

Fi! = 3,5; - переменная вещественного типа;

F$ = "инженер" - переменная символьного типа.

Если в начале кода в общей области модуля записать оператор

Option Explicit,

то это будет означать, что все используемые переменные должны быть описаны. Таким образом, будет производиться автоматическая проверка правильности написания имен.

Константа – это именованная область памяти для хранения данных, изменение которых во время работы программы не допускается. Они могут иметь указанные выше типы. Константа описывается ключевым словом **Const**, и при описании константы требуется присвоить ей значение. Примеры описания констант:

Const Pi as Single = 3.14159 описана константа Pi.

Const Name = “Visual Basic” константа строкового типа.

Иногда приходится преобразовывать данные одного типа в другой тип. Особенно часто приходится преобразовывать строки в числа и наоборот. Например, при вводе чисел они первоначально, как правило, воспринимаются как набор символов (цифр). Возникает необходимость распознать в этих символах число, чтобы можно было с ним выполнять математические действия. Для этого предназначена функция

Val(строка),

которая возвращает число. Данная функция превращает указанную в качестве аргумента строку в число, если строка состоит из цифр и одной точки. Иначе получим 0.

Обратное преобразование часто бывает необходимо при выводе результата. Функция

Str(число)

превращает указанное в качестве аргумента число в строку, т.е. воспринимает число как набор символов. Когда найден результат вычислений, и мы хотим этот результат записать, например, в текстовое поле, нужно предварительно превратить его в строку, а уже затем записать в соответствующее поле.

Кроме функций Val и Str в VBA имеются другие функции преобразования типов. Как правило их названия начинаются с буквы С от слово covrrsion. Некоторые из них приведены ниже

Функция	Тип, в который преобразуется выражение
CInt(выражение)	Integer
CSng(выражение)	Single
CSt(выражение)r	String

CDate(выражение)	Date
CBool(выражение)	Boolean
CVar(выражение)	Variant

Ввод и вывод данных с помощью встроенных диалоговых окон

Ввод и вывод данных могут быть выполнены следующими способами:

1) с помощью встроенных диалоговых окон InputBox и MsgBox;
2) с помощью объекта Cells(номер строки, номер столбца);
3) с помощью файла прямого доступа.

Окно ввода информации выводится на экран с помощью функции InputBox, которая имеет ряд необязательных параметров тип String. Первые два из них соответствуют пояснящей надписи на окне и названию окна.

Предположим, мы хотим, чтобы окна называлось «Ввод данных» и пояснительная надпись была «Введите номер варианта n». Тогда соответствующий оператор можно записать так:

```
n = InputBox("Введите номер варианта n ", "Ввод данных")
```

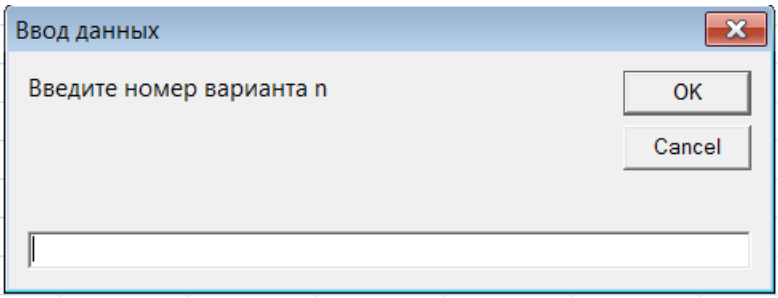


Рис. 2.1. Окно функции InputBox

При вызове функции InputBox откроется окно, приведенное на рис.2.1. Если ввести в текстовое поле некоторое значение и нажать кнопку OK, то переменная n примет введенное значение. Если же нажать кнопку Cancel, то функция ничего не вернет, т.е. значение переменной n не изменится. В обоих случаях окно закроется.

Следует помнить, что функция InputBox возвращает результат типа String. Поэтому, если переменная n не была заранее объявлена числовым типом, при использовании этой функции для ввода числовых данных необходимо дополнительно воспользоваться функцией Val для преобразования полученной строки в число, например,

n = Val(InputBox("Введите номер группы", "Ввод данных")).

Простейшим способом вывода результата является использование окно сообщений MsgBox. Оно выводится на экран с помощью оператора

MsgBox строка,

где строка – это выводимый результат, записанный в виде строки.

При выполнении оператора появится окно сообщений с указанной строкой в виде сообщения. Например, при выполнении оператора

MsgBox "Лабораторная работа № 1" & Chr(13) & “Студент Заманов И.А.”

откроется окно, изображенное на рис.2.2. В данном примере после названия процедуры MsgBox используется объединение

нескольких строковых величин в одну при помощи операции объединения строк &.

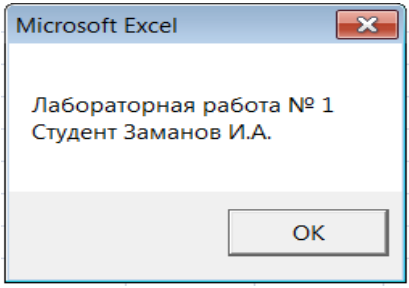


Рис. 2.2. Окно сообщений

В списке вывода оператора MsgBox, после символьного выражения "Лабораторная работа № 1", записана встроенная функция Chr(13). Функция Chr() имеет один аргумент в виде целого числа и возвращает символ из таблицы кодирования Windows по номеру – аргументу функции. Функция Chr(13) равносильна нажатию клавиши “Enter“, поэтому символьное выражение “Студент Заманов И.А.” напечатолось с новой строки.

Заметим, что если вместо строки записать некоторое выражение другого типа, то сначала будет найдено значение этого выражения, затем оно преобразовано в тип String, и полученный результат будет выведен в окне. Например, при выполнении операторов

```
Sub P1()
Dim a As Boolian
n!=5
a = True
MsgBox n^2+1 & Not a
```

End Sub

получим окно, изображенное на рис. 2.3.

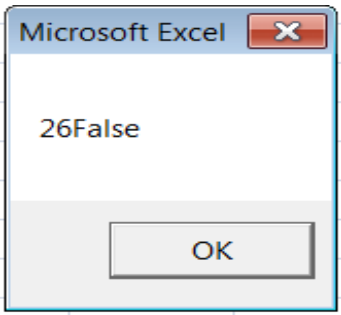


Рис. 2.3.Окно вывода

Запись и считывание данных из ячеек рабочего листа Excel

Обмен данными между программой и таблицей Excel может быть произведен с помощью оператора присваивания. Для этого можно использовать коллекцию ячеек таблицы Cells или диапазон ячеек Range. Для доступа из программы к значениям, хранящимся в ячейках некоторого листа, применяется одна из двух следующих комбинаций:

Sheets("НазваниеЛиста"**).Range(**"ЯчейкаИлиДиапазонЯчеек"**).Value**

или

Sheets("НазваниеЛиста")**.Cells**(НомерСтроки, НомерСтолбца)**.Value**

Диапазон ячеек в **Range** задаётся при помощи начальной и конечной ячеек, разделённых двоеточием. В качестве номеров строк и столбцов в **Cells** могут быть только ненуле-

вые целочисленные параметры: числа, переменные, выражения и т. д.

Если данная комбинация находится **слева** от знака равенства в операторе присваивания, то происходит **запись данных в ячейку**. Если данная комбинация находится **справа** от знака равенства в операторе присваивания, то происходит **чтение данных из ячейки**. Например, результат выполнения программы, приведенной на рис. 2.4, показан на рис. 2.5. Как видим, данный способ выделения коллекции ячеек позволяет достаточно просто выполнять ввод –вывод данных, а также дает возможность использовать эти адреса ячеек в арифметических и логических выражениях.

```
Sub P2()
Dim d As String
d = "A1:C2"
Sheets("Лист1").Range(d).Value = 3
Sheets("Лист1").Range("a3").Value =100+Sheets("Лист1").Cells(1,1)^3
Sheets("Лист1").Cells(3, 2) = Cells(3, 1) + 1000
End Sub
```

Рис. 2.4. Листинг программы.

	A	B	C
1	3	3	3
2	3	3	3
3	127	1127	
4			
5			

Рис. 2.5. Результат выполнения программы

Если процедура написана для определенного листа, то метод Select можно не использовать (рис. 2.6). При этом результаты не изменятся (рис. 2.5) и будут помещены на этом же листе.

```
Sub P2()
Dim d As String
d = "A1:C2"
Range(d) = 3
Range("a3") = 100 + Cells(1, 1) ^ 3
Cells(3, 2) = Cells(3, 1) + 1000
End Sub
```

Рис. 2.6. Листинг программы метод Select

Арифметические выражения

Программа линейной структуры обычно имеет следующую структуру:

Sub Имя()
<объявление переменных и констант>
<ввод исходной информации>
<последовательность операторов присваивания>
<вывод результатов вычислений>
End Sub

Оператор присваивания предназначен для изменения значений переменных и имеет следующий вид

имя переменной = выражение.

В правой части оператора присваивания чаще всего используется арифметическое выражение, которое записывается по правилам конкретного языка программирования. Чтобы арифметическое выражение давало правильный результат необходимо помнить о приоритете выполнения действий. Тут все как в элементарной математике:

- сначала вычисляются функции, если они есть;
- затем выполняется возведение в степень,
- потом умножение и деление;
- в последнюю очередь — сложение и вычитание.

Действия одинаковой очередности выполняются слева направо.

Имеется ряд встроенных математических функций. Каждая функция имеет имя и аргумент. Имя функций состоит из трех латинских букв, аргумент заключается в скобки. Между именем и скобками не должно быть никаких символов. Некоторые функции приведены в таблице 1.

Таблица 1

№	Название функции	Математическое определение	Запись на VBA
1.	Синус	sin x	SIN(X)
2.	Косинус	cos x	COS(X)
3.	Тангенс	tg x	TAN(X)
4.	Арктангенс	arctg x	ATN(X)
5.	Показательная функция	e^x	EXP(X)
6.	Натуральный логарифм	ln x	LOG(X)
7.	Десятичный логарифм	lg x	LOG(X)/LOG(10)
8.	Абсолютная величина	|x|	ABS(X)
9.	Квадратный корень	$\sqrt{x}$	SQR(X)
10.	Целая часть числа	[x]	INT(X)

11.	Датчик случайных чисел	-	RND(X)
12.	Число π	π	4*ATN(1)

Примеры записи арифметических выражений:

$\sqrt[3]{x^2} + tg|y|$ X^(2/3)+TAN(ABS(Y))

$\frac{a^3 + e^{-k}}{\sqrt{y} + \ln^2 x}$ (A^3+EXP(-K))/(SQR(Y)+LOG(X)^2)

Задание 2.1. Составить программу для вычисления

$$z = \frac{\sqrt[3]{x+y}}{1+\cos^2 x}, \text{ где}$$

$$y = \ln n^3 + \frac{1}{n+m}, \qquad x = e^{\sqrt{m}} + \sin^3 n^2.$$

Порядок выполнения работы.

1. Откройте вкладку VBA, например, через Alt+F11.

2. Создайте новый модуль **Insert/ Module** и наберите программу

```
Sub Пример1()
Dim n As Integer, m As Integer
Dim x As Single, y As Single, z As Single
n=InputBox(“n=”,“Введите значения  n”)
m=InputBox(“m=”,“Введите значения m”)
x=exp(sqr(m))+sin(n^2)^3
y=log(n^3)+1/(n+m)
z=(x+y)^(1/3)/(1+cos(x)^2)
```

```
MsgBox “z=” & Format(z,”##0.###”)
End Sub
```

3. Запускаем программу на выполнение (F5). Например, при значениях параметров n=1, m=2 получим z=0,793 (рис. 2.7).

4. Окончательную программу и ответ переписываем в тетрадь.

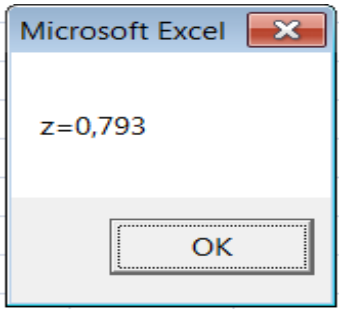


Рис. 2.7. Результат выполнения программы

Задание 2.2. Составить программу для вычисления площади треугольника по заданным значениям сторон a,b,c. Использовать метод Герона

$S = \sqrt{p(p-a)(p-b)(p-c)}$, где $p = (a+b+c)/2$.

Порядок выполнения работы такой же. Результат вычислений запишется в ячейках A1, B1.

```
Sub Geron()
Dim a, b, c, p, s As Double
a = Val(InputBox("Введите a"))  'ввод значения переменной a
b = Val(InputBox("Введите b"))  'ввод значения переменной b
c = Val(InputBox("Введите c"))  'ввод значения переменной c
'P - полупериметр,S - площадь
p = (a + b + c). 'вычисляет полупериметр
s = Sqr(p * (p - a) * (p - b) * (p - c)) ). 'вычисляет площадь
Cells(1, 1) = "Площадь=" 'вывод текста в ячеку A1
Cells(1, 2) = s. 'вывод значения периметра в ячеку B1
End Sub
```

Рис. 2.8. Листинг программы вычисление площади треугольника

Работа с данными типа дата/время

Дата определяется следующим образом. Имеет некоторая точка отсчета (базовая дата) - это 30 декабря 1899 года. Эта дата принимается за нуль. Каждая другая дата получается, как количество дней, предшествующих этой дате или прошедших с этой даты, т.е. по сути, она выражается целым числом.

Время рассматривается как некоторая часть суток. Если сутки - это единица, то время - это часть единицы. Один час - это 1/24 дня, одна минута - 1/1440 дня, секунда - 1/86400 дня.

Таким образом, дата и время в сумме дают некоторое вещественное число, у которого целая часть определяет дату, а дробная часть - время. Поэтому над датами и временем можно производить определенные арифметические действия с помощью обычных знаков действий. Можно, например, вычитать одну дату из другой, добавлять к дате или вычитать числа для изменения ее значения. Например, если необходимо опреде-

лить количество дней между двумя датами, просто необходимо вычесть более раннюю дату из более поздней даты. Поскольку это значения типа Date, VBA «знает», что целью вычисления является получение разности в днях между двумя этими датами. Аналогично, если необходимо определить дату через 50 дней после определенной даты, необходимо прибавить 50 к этой дате.

При записи дат используется символ # и возможны два формата. Например, дату 21 января 2018 года можно записать в двух вариантах: #1/21/2018# или #2018-01-21#.

Приведем пример записи даты со временем: #2018-01-21 13:45:20#.

Существуют встроенные функции для работы с датами и временем.

Контрольные вопросы:

1. Дать описание структуры программы.
2. Как объявлять переменные и константы в программе?
3. Как осуществляется ввод и вывод данных с помощью встроенных диалоговых окон?
4. Как осуществляется запись и считывание данных из ячеек рабочего листа Excel?
5. Как осуществляется работа с данными типа дата/время?

3. ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ РАЗВЕТВЛЯЮЩЕЙСЯ СТРУКТУРЫ

Разветвляющиеся программы составляются с использованием операторов безусловного и условного перехода, а также с помощью оператора множественного выбора. В подобных программах последовательность выполнения операций заранее

не определена и ставится в зависимость от результатов проверки заданных условий.

Оператор безусловного перехода предназначен для передачи управления на указанную строку внутри процедуры. Он имеет вид

GOTO M ,

где M –номер строки или метка. Метка представляет собой небольшое натуральное число или последовательность символов, которая заканчивается символом двоеточия. Она ставится в начале строки перед оператором и предназначена для идентификации оператора. Действие оператора GoTo состоит в передаче управления меченой строке.

Оператор условного перехода предназначен для передачи управления в зависимости от выполнения некоторых условий. Условные операторы могут использоваться для организации разветвлений и циклов. Имеются строковые и блоковые (структурные) условные операторы.

Строковой оператор имеет две формы:

1) IF «условие» THEN «оператор»

2) IF «условие» THEN «оператор1» ELSE «оператор2»

При выполнении оператора сначала проверяется условие. В случае его выполнения, работает оператор, записанный после слова THEN. В противном случае выполняется оператор, записанный после слова ELSE. Если ELSE отсутствует, то управление передается на следующий оператор.

Блоковый условный оператор имеет следующий вид:

1 вариант

IF «условие» THEN
«Блок1»
[ELSE
«Блок2»]
ENDIF

2 вариант

IF «условие1» THEN
«Блок1»
[ELSE IF «условие2» THEN
«Блок2»]
…
ELSE
«БлокK»
ENDIF

Здесь параметры (Блок1, Блок2,…) включают в себя один или более операторов в одной или более строках. Квадратные скобки означают, что заключенные в них команды могут опускаться.

Пример 3.1. Вычислить

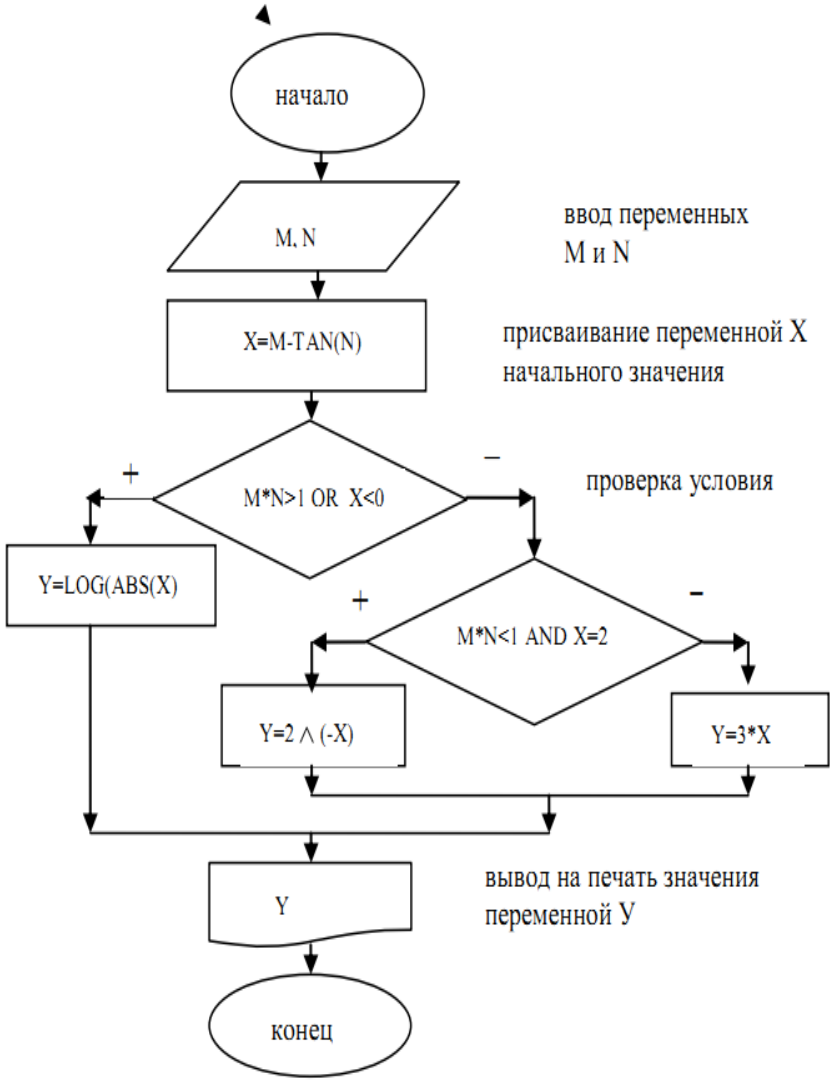
$$y = \begin{cases} \ln|x|, \text{ если } mn > 1 \text{ или } x < 0; \\ 2^{-x}, \text{ если } mn < 1 \text{ и } x = 2; \\ 3x, \quad \text{во всех остальных случаях}; \end{cases}$$

где x= m-tgn

Программа может иметь следующий вид:

```
Sub Пример31()
Dim n As Integer, m As Integer
Dim x As Single, y As Single
n=InputBox(“n=”,“Введите значения  n”)
m=InputBox(“m=”,“Введите значения m”)
        x=m - tan(n)
        If  m*n>1  Or  x<0  Then
        y=log(abs(x))
        ElseIf  m*n<1  And x=2  Then
        y=2^(-x)
        Else
        y=3*x
        EndIf
MsgBox “y=” & y
End Sub
```

Блок-схема программы:



Оператор множественного выбора SELECT CASE.

Иногда приходится делать выбор одного действия из целой группы действий на основе проверки нескольких различных условий. В таких случаях удобно использовать специально предназначенный для этого оператор выбора Select Case. Оператор выбора имеет следующий вид:

```
SELECT CASE тест_выражение
CASE условие_1
Блок_операторов_1
[CASE условие_2
Блок_операторов_2]
. . . . . .
[CASE ELSE
Блок_операторов_n]
END SELECT
```

Здесь **тест_выражение** — любое числовое или строковое выражение, которое в процессе выполнения программы принимает то или иное значение. После слово CASE надо указать условие в одном из трех форматов:

1) CASE константа_1, константа_2, ...
2) CASE IS знак_отношения константа
3) CASE константа_1 TO константа_2

Константы в условии должны быть того же типа, что и тест_выражение в SELECT CASE. В качестве знака отношения в формате 2) можно использовать любой из знаков операций отношений, перечисленных в алфавите VBA.

Алгоритм множественного выбора заключается в следующем. В начале работы оператора вычисляется тест выражение. Оно может возвращать значение любого типа, например, логическое, числовое или строковое. Далее проверяется, удовлетворяет ли это значение одному из указанных в CASE условий. Если значение удовлетворяет какому-то условию, выполняется блок операторов, следующий за данным CASE. Если ни одно условие не удовлетворяется, выполняется блок операторов, следующий за CASE ELSE.

Пример 3.2.

```
Sub Пример32()
Dim x As Single
x=InputBox("x=","Введите значения  x")
Select Case x
Case 2, 7
MsgBox  "X равно 2 или 7"
Case IS >50
MsgBox  "X БОЛЬШЕ 50"
Case -13 TO 0.5
MsgBox  "X  не меньше -13, но не больше 0.5"
Case Else
MsgBox  "Ни одно условие не выполняется"
End Select
End Sub
```

Контрольные вопросы:

1. Описать операторы безусловного и условного перехода?
2. Описать оператор множественного выбора?
3. Описать алгоритм разветвляющейся структуры?

4. ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ

Многие вычисления приходится проводить несколько раз. Например, циклические алгоритмы применяются при составлении таблицы значений функций, вычислении суммы и произведений, обработки массивов. Циклы, в которых число повторений заранее известно, называются арифметическими циклами. Циклы, в которых известно только условие завершения цикла, а число повторений цикла заранее неизвестно, называются итерационными циклами.

Следует отметить, что циклические вычисления можно организовать при помощи условных операторов. Структура организации цикла с помощью операторов IF и GOTO выглядит следующим образом:

```
    X = A.
20 REM     Тело цикла
    . . . . . .
    X =X+H
    IF  X <= B. THEN  GOTO  20
```

Здесь X – управляющая переменная цикла; A – начальное значение переменной X, B. – конечное значение переменной X, H - шаг изменения переменной X.

Однако существуют специальные операторы, значительно упрощающие построение программ с циклами. В VBA есть богатый выбор средств организации циклов, которые можно разделить на две основные группы:

1. Циклы с параметром For … Next

2. Циклы с условием Do … Loop

Цикл For … Next используется для организации арифметических циклов. Циклы типа Do … Loop используются в тех случаях, когда заранее неизвестно сколько раз должно быть повторено выполнение блока операторов, составляющего тело цикла. Такой цикл продолжает свою работу до тех пор, пока не будет выполнено определенное условие.

Цикл с параметром FOR – NEXT. Структура цикла с параметром выглядит следующим образом:

```
FOR  X=A  TO  B  STEP  H
REM     Тело цикла
   . . . . . .
NEXT  X
```

Здесь X – управляющая переменная (параметр) цикла; A,B,H – начальные и конечные значения параметра и шаг его изменения. Если H=1, то конструкцию STEP H можно опустить.

После завершения цикла управление передается команде, следующий за оператором NEXT.

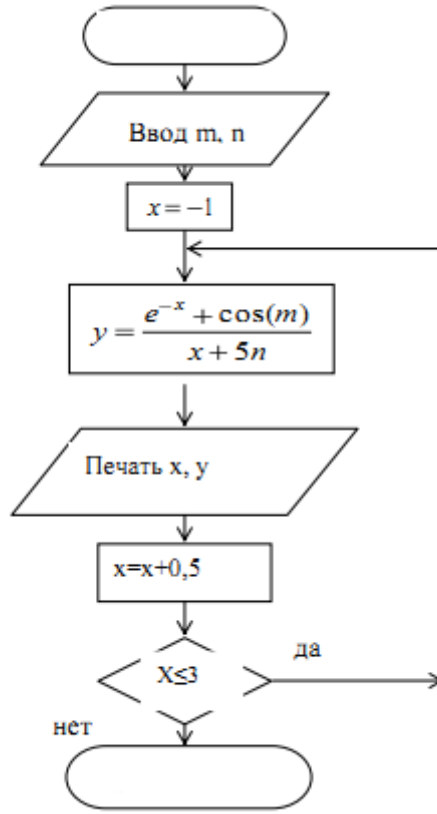
С помощью FOR – NEXT можно организовать вложенные циклы – каждый со своими FOR, NEXT и счетчиком.

Пример 4.1. Составить таблицу значений функции $y = \frac{e^{-x} + \cos(m)}{x + 5n}$ для аргумента x, изменяющегося от -1 до 3 с шагом 0,5.

Программа может иметь следующий вид:

```
Sub Табулирование функции()
Dim m As Integer
Dim n As Integer
Dim r As Integer
Dim x As Single, y As Single
        Cells(1,1)=”Значение аргумента”
        Cells(1,2)=”Значение функции”
        m=6
        n=3
        k=2
For x=-1 To 3 Step 0.5
        y=(exp(-x)+cos(m))/(x+5*n)
        Cells(k,1)=x
        Cells(k,2)=y
        k=k+1
Next x
End Sub
```

Блок-схема программы:



Универсальный цикл DO…LOOP (ДЕЛАТЬ…ЦИКЛ). Конструкция цикла с условием имеет четыре формата:

1вариант

DO WHILE логич_выражения
<блок_команд>
. . . . . .
LOOP

Указанная конструкция работает следующим образом:

Блок_команд (тело цикла) выполняется до тех пор, пока значение логич_выражения - «Истина». Если при входе в цикл значение логич_выражения – «Ложь», блок_команд не выполняется ни разу.

Пример 4.2. Составить программу для вычисления корня уравнения $x^2 + 2x - 3 = 0$ в интервале $[0;2]$ с точностью $\varepsilon = 0{,}0001$.

Решение. Для решения используем метод Ньютона. Предварительно найдем значения первой и второй производных: $y' = 2x + 4;\ y'' = 2$. Значения функции в граничных точках интервала равны $y(a) = y(0) = -3,\ y(b) = y(2) = 5$. За начальное приближение корня выбираем правую границу $b = 2$, так как вторая производная положительна и выполняется условие $y(b)y''(b) > 0$.

Программа может иметь следующий вид:

```
Sub Метод Ньютона()
Dim y, dydx, Eps, x, x1 As Double
x=2
Eps=0.1
        Do While Eps>0.0001
y=x^2+2*x-3
        dydx=2*x+4
        x1=x-y/dydx
        Eps=Abs(x1-x)
```

```
MsgBox “Приближения=” & Eps
x=x1
        Loop
MsgBox “Корень уравнения x=” & x
End Sub
```

2 вариант

```
DO UNTIL логич_выражения
 <блок_команд>
 . . . . . .
 LOOP
```

Блок_команд (тело цикла) выполняется до тех пор, пока значение логич_выражения – «Ложь». Если при входе в цикл значение логич_выражения – «Истина», блок_команд не выполняется ни разу.

3 вариант

```
DO
 <блок_команд>
  . . . . . .
   LOOP WHILE логич_выражение
```

Блок_команд (тело цикла) выполняется до тех пор, пока значение логич_выражения – «Истина». При первом входе в цикл значение логич_выражения не проверяется, поэтому блок_команд будет выполнен хотя бы один раз.

4 вариант

```
DO
<блок_команд>
. . . . . .
 LOOP UNTIL логич_выражение
```

Блок_команд (тело цикла) выполняется до тех пор, пока значение логич_выражения – «Ложь». При первом входе в цикл значение логич_выражения не проверяется, поэтому блок_команд будет выполнен хотя бы один раз.

Чтобы не произошло «зацикливания» в теле цикла величины, входящие в условие, должны меняться. Альтернативный способ выхода из цикла представляет инструкция Exit Do.

Контрольные вопросы.

1. Алгоритмы циклической структуры?
2. Описать цикл с параметром FOR – NEXT?
3. Циклы с условием Do … Loop?

5. РАБОТА С МАССИВАМИ

Массив - это переменная, в которой хранится одновременно несколько значений одинакового типа, доступ к которым осуществляется по индексу (порядковому номеру). Таким образом, массив - это совокупность однотипных индексированных данных. Например, список фамилий студентов, численные данные о среднесуточной температуре за месяц, буквы русского алфавита и т.д.

Массивы объявляются с помощью оператора Dim (от англ. "Dimension"), где указывается имя, размер и размерность массива:

Dim имя (размер1, размер2,…) **As** тип данных

Здесь указанные в скобках величины размер1, размер2,… задают размеры массива – количество индексов и максимально допустимое значение для каждого конкретного индекса. По умолчанию индексирование элементов массива начинается с нуля. Например, Dim D(3) As Integer

Это означает, что в одномерном массиве D будет храниться 4 элемента D(0), D(1), D(2) и D(3) типа Integer.

В качестве стандартного значения нижней границы индекса может использоваться не только ноль. Чтобы изменить это стандартное значение, нужно воспользоваться оператором Option Base. Например, если в начале модуля разместить оператор

Option Base 1,

то индексирование элементов массивов по умолчанию будет начинаться не с нуля, а с единицы.

Другим способом изменения базового индекса является использование ключевого слова To при объявлении массива. Например, одномерном массив G из трех элементов может быть объявлен так:

Dim G (1 To 3) As Integer

Обычно элементы массива содержат значения одного и того же типа. Если же необходимо, чтобы в массиве содержались данные разных типов, при объявлении массива нужно указать тип Object:

Dim F(2) As Object

Элементы такого массива могут содержать значения разных типов:

F(0) = "Салимов"
F(1) = 30
F(2) = 3.14157

При создании двумерных массивов нужно указать количество строк и столбцов, например

Dim B (1 To 2, 1 To 2) As Single

Если размер массива заранее неизвестно, то можно создать динамический массив. Динамические массивы создаются с помощью оператора Dim без указания его размера. Затем их размер устанавливается с помощью оператора ReDim во время выполнения процедуры. Например:

Dim Gruppa( ) As String - объявляет динамический массив Gruppa;

Redim Gruppa (1 To 25) – изменяет размер массива до 25 элементов;

Dim X As Variant –объявляет переменную типа Variant;

ReDim X(20) As Integer – создает массив из 20 целых чисел в Variant.

Ввод значений массива осуществляется в цикле поэлементно. В качестве переменной цикла участвует индекс массива. Если индексов несколько, то используются вложенные циклы в том же количестве. В теле цикла размещают оператор ввода, который непосредственно присваивает элементам массива соответствующие значения.

Если результатом выполнения программы является массив, то вывод опять надо производить в цикле.

Пример 5.1. В первой строке таблицы Excel записаны десять вещественных чисел. Требуется найти сумму этих чисел и вывести в ячейку A2. Кроме того, требуется найти квадраты этих чисел и вывести в третью строку.

Программа может иметь следующий вид:

```
Sub macro()
' описание массивов
Dim a(1 To 10) As Single, b(1 To 10) As Single, s As Single
' ввод значений
```

```
For k=1 To 10
a(k) = Cells(1, k)
Next
' вычисление суммы
s=0
For k=1 To 10
s=s + a(k)
Next k
‘вывод суммы
Cells(2, 1) = s
' вычисление квадратов
For k=1 To 10
b(k) = a(k)^2
Next
' вывод квадратов
For k=1 To 10
Cells(3, k) = b(k)
Next
End Sub
```

Контрольные вопросы.

1. Дать определение массиву.
2. Как массивы объявляются в программе?
3. Описать процесс ввода значений массива?

6. ФУНКЦИИ И ПРОЦЕДУРЫ

Когда некоторые последовательности действий выполняются многократно с разными исходными данными, их целесообразно оформлять в виде так называемых процедур. В VBA имеются два типа процедур - процедура типа Function и про-

цедура общего вида Sub. Процедуру Function обычно называют пользовательскими функциями или просто функциями.

Процедуры и функции представляют собой относительно самостоятельные фрагменты программы, оформленные особым образом и снабженные именем. Упоминание этого имени в тексте программы называется вызовом процедуры (функции). В отличие от процедуры, имя функции выступает также в качестве переменной и используется для возвращения значения в точку вызова функции.

Пользовательская функция

Пользовательские функции описываются следующим образом

Function *Имя*(формальные_параметры As тип) As тип
тело функции
Имя = результат
End Function

Здесь *Имя* - имя функции, которая определяется аналогично имени переменной. В функциях именно *Имя* функции используется в качестве выходного параметра. Поэтому обязательно должен присутствовать оператор, присваивающий результат имени функции. В противном случае функция выведет нулевое значение. Для каждого параметра, а также для функции желательно указывать его тип. В противном случае будет использоваться тип Variant.

Для использования функция должна быть вызвана. Вызов производится по аналогии с вызовом стандартных функций, т.е. имеет вид

Имя(фактические_параметры)

Фактические параметры по количеству, порядку и типу должны соответствовать формальным параметрам. Вызов функции может быть использован либо в операторе некоторой процедуры VBA, либо в формулах при непосредственной работе с таблицей Excel.

Пример 6.1. Составить таблицу значений функции $y = 5*\sin(3*x)$ для аргумента x, изменяющегося от a=0 до b=4 с шагом h=0,2.

Приведем код макроса, в котором эта функция используется. Предположим, что макрос предназначен для табулирования функции, причем все данные (концы отрезка и шаг) считываются с соответствующих ячеек второго столбца, а значения аргумента и функции выводятся соответственно в третий и четвертый столбцы

```
Sub Macro()
a = Cells(1,2)
b = Cells(2,2)
h = Cells(3,2)
k = 1
For x = a To b + h/2 Step h
Cells(k,3) = x
Cells(k,4) = f(x)
k = k+1
Next x
End Sub
Function f(x As Single) As Single
f=5*sin(3*x)
End Function
```

Если просмотреть список функций Excel, то наша функция f(x) окажется в разделе *Определенные пользователем*. Эту функцию можно использовать наравне с другими при записи в ячейку формул. Например, мы можем записать в некоторую ячейку следующую формулу: = f(A2).

Как правило, функция возвращает некоторое значение. Поэтому ее вызов записывают в выражении, как в приведенном примере. Однако может оказаться, что в конкретной ситуации возвращаемое значение не представляет интереса. Это возможно, например, в тех случаях, когда в теле функции присутствуют операторы вывода. Тогда вызов можно записать так:

Имя фактические_параметры

Здесь скобки не используются.
Рассмотрим код

```
Function summa(x As Single, y As Single) As Single
c = a + b
Cells(1,1) = c
summa = c
End Function
```

Эта функция вычисляет сумму двух чисел и возвращает ее, но вместе с тем она записывает эту сумму в ячейку A1. Эту функцию используем в макросе

```
Sub macro()
summa 4, 5
End Sub
```

При выполнении макроса в ячейку A1 будет записано число 9.

Процедура общего вида

У функции пользователя имеется один недостаток – она возвращает в основную программу лишь одно вычисленное значение, присвоенное к имени подпрограммы. Процедура Sub позволяет вычислять любое количество значений, поэтому она называется процедурой общего вида. Для возвращения в основную программу вычисленных результатов применяются дополнительные параметры в списке аргументов. Эти процедуры имеют вид

Sub *Имя*(параметры As тип)
тело процедуры
End Sub

При составлении программ у нас основными процедурами были макросы. Макрос – это процедура типа Sub без параметров. В VBA их можно выполнять непосредственно, не используя для этого другие процедура.

Здесь, в отличии от функции пользователя, параметры могут являться как входными , так и выходными. Они записываются в любом порядке

Вызов процедуры может быть выполнен двумя способами:

1) *Имя* параметры
2) Call *Имя* (параметры)

В первом случае скобки не используются. Параметры - это фактические параметры. Если параметр входной, то в ка-

честве фактического параметра, как и у функции, могут выступать константы, переменные или выражения соответствующего типа. Если же параметр выходной, то в качестве фактического параметра могут выступать только переменные.

Заметим, что один и тот же параметр может одновременно играть и роль входного параметра, и роль выходного параметра. В этом случае параметр называется обновляемым и в качестве фактического параметра должна выступать переменная, которой предварительно следует присвоить некоторое значение.

Фактические параметры по количеству, порядку, типу и статусу (входной, выходной, обновляемый) должны соответствовать формальным параметрам.

Рассмотрим код

```
Sub ss(a, b, c)
c = a + b
b = a * b
End Sub
Sub macro()
a = 3
b = 4
ss a, b, c
Cells(1, 1) = b
Cells(2, 1) = c
End Sub
```

Здесь мы имеем две процедуры Sub. Первая процедура ss – это процедура общего вида. У нее три параметра. Первый параметр – это входной, второй – обновляемый, а третий – вы-

ходной. Вторая процедура – это макрос. При выполнении макроса в указанные ячейки будут выведены числа 12 и 7.

Пример 6.2. Вычислить определитель n-го порядка Программа может иметь следующий вид:

```
Sub Вычисление определителя()
Dim a(25, 25) As Double
Dim i, j, n As Integer
n = Val(InputBox("Порядок определителя n="))
For i = 1 To n
For j = 1 To n
a(i, j) = Cells(i, j)
Next j
Next i
Call opr(a(), n)
End Sub
Sub opr(a(), n)
s = 1
For k = 1 To n - 1
For i = k + 1 To n
c = a(i, k) / a(k, k)
For j = k + 1 To n
a(i, j) = a(i, j) - c * a(k, j)
Next j
Next i
Next k
For i = 1 To n
s = s * a(i, i)
Next i
```

```
MsgBox ("Определитель=" + Str(s))
End Sub
```

Пример 6.3. Вычислить значение функции $y = \frac{e^{2,5} + e^{10}}{2}$.

Решение. Функцию $y = e^x$ можно аппроксимировать многочленом n- ой степени или конечной суммой вида $S = 1 + x + \frac{x^2}{2} + ... + \frac{x^n}{n!} \approx e^x$. Сумму ряда оформим в виде подпрограммы.

Программа может иметь следующий вид:

```
Sub Вычисление выражений с exp(x) ()
Dim s1, s2, x1, x2, y As Double
x1 = 2.5
Call R(x1, s1)
x2=10
Call R(x2, s2)
y=(s1+s2)/2
MsgBox ("Ответ:" + Str(s))
End Sub
Sub R(x,s)
Dim i, n As Integer
Dim s, a, x As Double
n=30
s = 1
a=x
```

```
For i = 1 To n
s=s+a
a=a*x/(i+1)
Next i
MsgBox ("Сумма ряда exp(x) =" + Str(s))
End Sub
```

Контрольные вопросы:

1. Опишите пользовательскую функцию.
2. Опишите процедуру общего вида.
3. Что означают формальные и фактические параметры?

ЛИТЕРАТУРА

1. Кудинов Ю. И., Пащенко Ф. Ф. Основы современной информатики: Учебное пособие. 5-е изд. – СПб.: Издательство «Лань», 2019. – 256 с.

2. Кудинов Ю. И., Пащенко Ф. Ф., Келина А. Ю. Практикум по основам современной информатики. – СПб.: Издательство «Лань», 2011. – 352 с.

3. Экономическая информатика: учебное пособие / коллектив авторов; под редакции Д.В. Чистова. – М.: КНОРУС, 2009. – 512 с.

4. Андреева Н. М., Василюк Н. Н., Пак Н. И., Хеннер Е. К. Практикум по информатике: Учебное пособие. – 2-е изд. – СПб.: Издательство «Лань», 2019. – 248 с.

5. Биллиг В.А. Основы офисного программирования и язык VBA. – Издательство: Национальный Открытый Университет «ИНТУИТ», 2016. – 708 с.

СОДЕРЖАНИЕ

Введение …… 3
1. Создание и запись макроса……. 5
2. Ввод и вывод информации. Программирование алгоритмов линейной структуры……..... 16
3. Программирование алгоритмов разветвляющейся структуры 33
…… 40
4. Программирование алгоритмов циклической структуры.. 46
5. Работа с массивами…….. 49
6. Функции и процедуры……. 58
7. Литература……. 59
8. Содержание……