ИНФОРМАТИКА И ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

РЕШЕНИЕ ИНЖЕНЕРНЫХ ЗАДАЧ СРЕДСТВАМИ MS EXCEL И VBA

Методические указания к лабораторным работам для студентов бакалавриата направления 22.03.01

САНКТ-ПЕТЕРБУРГ
2020

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
Санкт-Петербургский горный университет

Кафедра информатики и компьютерных технологий

ИНФОРМАТИКА И ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

РЕШЕНИЕ ИНЖЕНЕРНЫХ ЗАДАЧ СРЕДСТВАМИ MS EXCEL И VBA

Методические указания к лабораторным работам для студентов бакалавриата направления 22.03.01

САНКТ-ПЕТЕРБУРГ
2020

УДК 681.142.2 (073)

ИНФОРМАТИКА И ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ. Решение инженерных задач средствами MS Excel и VBA: Методические указания к лабораторным работам / Санкт-Петербургский горный университет. Сост.: *А.А. Кочнева, Т.В. Сарапулова*. СПб, 2020. 47 с.

Методические указания предназначены для выполнения лабораторной работы в рамках изучения дисциплины «Информатика и информационно-коммуникационные технологии». В состав указаний входят теоретический материал, описание технологии решения задач средствами MS Excel и VBA, приведены задание для лабораторной работы, список рекомендуемой литературы, контрольные вопросы для самопроверки.

Методические указания предназначены для студентов направления подготовки 22.03.01 «Материаловедение и технологии материалов».

Научный редактор доц. *А.Б. Маховиков*

Рецензент канд. техн. наук *К.В. Столяров* (Корпорация «Телум Инк»)
,

ВВЕДЕНИЕ

Цель работы – изучить возможности MS Excel и встроенного языка программирования Visual Basic for Applications для решения инженерных задач.

Выполнение лабораторной работы позволяет овладеть основными методами, способами и средствами получения, хранения, переработки информации, приобретение навыков работы с компьютером как средством управления информацией; реализовать способность выбрать инструментальные средства для обработки различных данных в соответствии с поставленной задачей, проанализировать результаты расчетов и обосновать полученные выводы.

Электронная таблица MS Excel – мощная и достаточно простая в использовании программа, предназначенная для решения широкого круга научно-технических, планово-экономических, учетно-статистических и других задач, в которых числовая, текстовая или графическая информация с некоторой регулярной, повторяющейся структурой представлена в табличном виде.

Программа MS Excel и встроенный язык программирования Visual Basic for Applications предоставляет богатые возможности создания и изменения таблиц, которые могут содержать числа, тексты, даты, денежные единицы, графику, математические и иные формулы для выполнения вычислений.

1. ЯЗЫК ПРОГРАММИРОВАНИЯ VISUAL BASIC FOR APPLICATIONS

В языке программирования Visual Basic for Applications (VBA) объект рассматривается как совокупность свойств (структур данных, характерных для этого объекта), методов их обработки и событий, на которые данный объект может реагировать и которые приводят, как правило, к изменению свойств объекта.

Объекты могут иметь идентичную структуру и отличаться только значениями свойств. В таких случаях в программе создается новый тип, основанный на единой структуре объекта. Он называется классом, а каждый конкретный объект, имеющий структуру этого класса, называется экземпляром класса.

Алфавит VBA

Как и все языки программирования высокого уровня, Visual Basic имеет свой набор допустимых для использования символов – алфавит. Алфавит языка содержит в себе:

– прописные и строчные буквы латинского алфавита:

A, B, …, Z;

a, b, …, z;

– прописные и строчные буквы русского алфавита:

А, Б, …, Я;

а, б, …, я;

– арабские цифры: 0, 1, …, 9.

Для построения конструкций языка используются также нижеперечисленные специальные символы:

– + сложение;

– - вычитание;

– * умножение;

– / деление;

– \ целочисленное деление;

– ^ возведение в степень;

– ( ) для составления сложных выражений;

– = оператор присваивания или операция отношения

– «равно», используемая в логических выражениях;

– & сцепление строк (конкатенация);

– ‘ комментарий;

– _ разрыв строки исходного кода программы или как связка в идентификаторах;

– > больше;

– < меньше;

– <= меньше или равно (не больше);

– >= больше или равно (не меньше);

– <> не равно;

пробел разделитель слов (операторов) в языке.

Из вышеописанных символов, относящихся к алфавиту языка, конструируются все его слова (предложения). К их числу относятся имена типов, встроенных констант, процедур, функций, операторов, объектов, свойств, методов и др. При этом прописные и

строчные буквы не различаются. Все слова, задействованные в конструкциях языка, являются зарезервированными словами и не могут применяться для других целей.

Переменные и константы

В VBA, как и в других языках программирования высокого уровня, для хранения значений используются два вида данных: переменные и константы.

Переменная – именованная область памяти, используемая для хранения значения, которое можно изменить при выполнении программы. Переменные подразделяются на простые и индексированные (переменные с индексом). Индексированными переменными являются элементы массивов.

Константа – именованная область памяти, используемая для хранения фиксированного значения, которое невозможно изменить при выполнении программы.

Имена переменных и констант составляются на основании следующих правил:

– первым символом всегда должна быть буква или символ подчеркивания "_";

– в составе имени нельзя использовать символы: **!**, **@**, **&**, **$**, **#**, **пробел**;

– в качестве имени нельзя использовать ключевые (зарезервированные) слова, входящие в конструкции языка VBA;

– длина имени не может быть более 255 символов;

– имя нельзя повторять в пределах области его видимости (действия).

Область действия

Область действия (видимости) переменных и констант определяется с помощью ключевых слов Private, Public и Static.

Private – область действия в пределах конкретного модуля. После завершения выполнения модуля память, отведенная под эти переменные, освобождается.

Public – область действия в пределах приложения.

Static – область действия в пределах конкретного модуля и внешних процедур, используемых в данном модуле. После завершения выполнения модуля значения этих переменных сохраняются и могут быть использованы при повторном выполнении модуля.

Переменные и константы, в зависимости от области действия, подразделяются на *глобальные* и *локальные*.

Если переменная или константа описана внутри процедуры, то она является *локальной*, то есть она определена и может использоваться только в пределах данной процедуры.

Если переменная или константа описана вне процедуры (в программе), то она будет *глобальной*. Такая переменная или константа может быть использована в других процедурах, внутренних по отношению к той, где она описана.

Объявление

Переменная объявляется (описывается) с помощью ключевых слов Private, Public, Static, Dim. Чтобы явно указать тип переменной, используется ключевое слово As.

Примеры описания простых переменных:

Private X

Public i As Integer, r As Long, c As Date

Static Строка As String

Dim Y

Dim Z As Single 'Явный способ объявления переменной. Самый простой и надежный

Примеры описания индексированных переменных:

Dim Массив1() As Integer

Dim Массив2(12) As String

Dim Массив3(1 to 20) As Single

Dim Массив4(1 to 5, 1 to 7) As Byte

Константа объявляется с помощью ключевого слова Const. При этом можно указать ее тип, область действия и присвоить ей значение.

Синтаксис объявления:

Const <имя константы> As <тип> = <значение>

или

Const <имя константы> = <значение>

Если в константе явно не указан тип данных, то VBA назначает ей тип, который соответствует присваиваемому значению.

Примеры:

Private Const q = 44,55
Public Const pi = 3,1459
Static Const QWER=2,54
Const y = 34
Const Con As Byte = 34
Const z As Single = -3,8374E-22

Все строковые константы указываются в кавычках.

Примеры:

Const prv As String = "*Язык программирования VBA*"
Public Const prv = "*Язык программирования VBA*"

Время жизни

Переменные и константы, объявленные как Private, сохраняют свое значение только во время выполнения блока кода, в котором они определены.

Переменные и константы, объявленные как Public, сохраняют свое значение и вне блока кода, в котором они определены, то есть до конца выполнения программы.

Переменные и константы, объявленные как Static, сохраняют свое значение и между вызовами процедур.

Переменные и константы, объявленные без ключевых слов Public, Private, Static, сохраняют свое значение согласно месту их объявления (описания).

Таблица 1

Типы переменных и констант

Тип данных	**Описание**	**Диапазон**
Byte	целое число	От 0 до 255
Integer	целое число	От -32768 до 32767
Long	длинное целое число	От -2147483648 до 2147483647
Single	число с плавающей запятой одинарной точности	а) для отрицательных чисел: от -3,402823E38 до -1,401298E-45 б) для положительных чисел: от 1,401298E-45 до 3,402823E38
Double	число с плавающей запятой двойной точности	а) для отрицательных чисел: от -1,79769313486231E308 до -4,94065645841247E-324 б) для положительных чисел: от 4,94065645841247E-324 до 1,79769313486231E308
Currency (денежный)	число с фиксированной десятичной точкой	от -922.337.203.685.477.5808 до 922.337.203.685.477.5807
String	строка символов	от 0 до 147483647 символов
Variant	универсальный	Значения любого типа
Boolean	логический	True или False
Date	дата	от 1.01.100 до 31.12.9999
Object	объект	Ссылка на любой объект

Примечание: для дробных чисел существует экспоненциальная форма записи, например, $1{,}456 \cdot 10^{23}$. В VBA число, записанное в этой форме, будет выглядеть так: 1,456E23, где E обозначает основание 10, а после E указывается степень числа. Запись этого же числа в отрицательной степени будет выглядеть так: 1,456E-23.

Управляющие конструкции

Операнды, операции, выражения

Операнды – это данные, над которыми выполняются какие-либо операции. В качестве операндов могут быть использованы константы, переменные, функции.

Операция – это элементарное действие над операндами.

Выражение – это последовательность операндов, отделенных друг от друга знаками операций. В выражении возможно присутствие парных круглых скобок. В зависимости от типа операндов

и используемых операций выражения делятся на *арифметические, логические и строковые* (*текстовые*) выражения.

Арифметическим называется выражение, содержащее операнды только числового (вещественного и/или целого) типа (числа) и знаки арифметических операций (табл. 2).

Таблица 2

Знак	Операция
-	Знак числа (смена знака)
^	Возведение в степень
*	Умножение
/	Деление
\	Целочисленное деление
mod	Остаток от деления по модулю
+	Сложение
-	Вычитание

Примеры: *Y = X*10, F = (x + y)/(3 + x^2), G = Sin(x).*

Выражение, содержащее хотя бы один знак операции отношения (табл. 3) или логической операции (табл. 4), называется *логическим*. Результатом выполнения такого выражения является логическая константа со значением либо True (истина), либо False (ложь). Логические выражения получаются в результате использования операций отношения (сравнения) либо логических операций. Операнды операций отношения должны быть одного типа (сравнимы между собой).

Таблица 3

Знак	Операция	Примеры (при значениях a = 6, b = 3)	Результат
<	Меньше	b<7	True
>	Больше	"Вася">"Маша"	False
<=	Меньше или равно	a+b<=12	True
>=	Больше или равно	b^2>=9	True
<>	Не равно	a<>b	True
=	Равно	B=10–a	False

Таблица 4

Знак	Операция	Пример
not	Логическое отрицание (НЕ)	not A
and	Логическое умножение (И)	A and B
or	Логическое сложение (ИЛИ)	A or B
xor	Исключающее ИЛИ	A xor B

Таблица 5

Знак	Пример	Результат
&	"Крас"&"ное"	"Красное"
+	"Чер"+"ное"	"Черное"

Строковое (*текстовое*) выражение (табл. 5) может содержать операнды только символьного (текстового или строкового) типа. В языках программирования, в том числе и в VBA, имеется только одна строковая операция, которая носит название *конкатенация* или *слияние* (*сцепление*) строк. Эта операция обозначается знаками «&» (коммерческое И – амперсанд) или «+» (плюс).

Приоритет выполнения операций

Приоритет операций (табл. 6) определяет порядок их выполнения в выражениях. Выражения вычисляются слева направо в порядке приоритета операций. Если в выражении имеются парные круглые скобки, тогда в первую очередь вычисляется выражение, расположенное в самых внутренних скобках.

Таблица 6

Приоритет	Операция
1	Вызов функций и выражения в скобках
2	^ (возведение в степень)
3	- (смена знака числа)
4	* (умножение), / (деление), \ (целочисленное деление), mod (деление по модулю)
5	+, – и & (сложение, вычитание и конкатенация)
6	<, <=, =,<>,>,>=
7	not
8	and
9	or
10	xor

Встроенные функции

В VBA имеется большой набор встроенных функций, использование которых существенно упрощает программирование. Эти функции можно разделить на следующие основные категории:

- математические функции (табл. 7);
- функции проверки типов (табл. 8);
- функции преобразования типов (табл. 9);

- функции обработки строк (табл. 10);
- функции времени и даты (табл. 11);
- функции выбора (табл. 12).

Таблица 7

Имя функции	Математическая интерпретация
Abs(x)	|x| (модуль числа x)
Atn(x)	Arctg x (арктангенс x)
Cos(x)	Cos x (косинус x)
Exp(x)	Экспонента (e в степени x)
Fix(x)	Отбрасывает дробную часть числа x
Int(x)	Округляет вещественное число x до целого в меньшую сторону
Log(x)	Ln x (натуральный логарифм x)
Rnd()	Генерирует случайное число от 0 до 1
Rnd(x)	Генерирует случайное число от 0 до x
Sgn(x)	Sign x – знак числа (сигнум x)
Sin(x)	Sin x (синус x)
Sqr(x)	Корень квадратный числа x
Tan(x)	Tg x (тангенс x)

Таблица 8

Имя функции	Проверка
IsArray (переменная)	Является ли переменная массивом?
IsDate (переменная)	Является ли переменная датой?
IsEmpty(переменная)	Инициализирована ли переменная?
IsError(переменная)	Является ли переменная кодом ошибки?
IsNull(переменная)	Является ли переменная пустой (Null)?
IsNumeric(переменная)	Является ли переменная числом?
IsObject(переменная)	Является ли переменная объектом?

Таблица 9

Имя функции	Тип, в который преобразуется выражение
CBool(Выражение)	Boolean (логический)
CByte(Выражение)	Byte (байтовый)
CCur(Выражение)	Currency (денежный)
CDate(Выражение)	Date (дата)
CDbl(Выражение)	Double (число с плавающей запятой двойной точности)
CInt(Выражение)	Integer (целое число)
CLng(Выражение)	Long (длинное целое число)
CSng(Выражение)	Single (число с плавающей запятой одинарной точности)
CStr(Выражение)	String (строка)
CVar(Выражение)	Variant (вариант)

Таблица 10

Имя функции	Описание
Mid(<строка>,<начало>[,<длина>])	Возвращает из строки подстроку указанной длины, начиная с заданного символа. Если длина не указана, то возвращается вся подстрока, начиная от заданного символа
Left(<строка>,<длина>)	Возвращает из заданной строки подстроку указанной длины, начиная с левого края строки
Right(<строка>,<длина>)	Возвращает из заданной строки подстроку указанной длины, начиная с правого края строки
Len(<строка>)	Возвращает длину указанной строки
Lcase(<строка>)	Преобразует в заданной строке все прописные буквы в строчные
String(<количество>,<символ>)	Повторяет заданный символ указанное количество раз
InStr(<начало>,<исходная строка>,<искомая подстрока>,<тип сравнения>)	Ищет подстроку в заданной строке, начиная с указанного символа. Тип сравнения: 0 – с учетом регистра (vbBinaryCompare); 1 – без учета регистра (vbTextCompare)
Trim(строка>)	Удаляет пробелы из начала и конца заданной строки
Ltrim(<строка>)	Удаляет все пробелы из начала заданной строки
Rtrim(<строка>)	Удаляет все пробелы из конца заданной строки
Space(<количество>)	Повторяет пробел указанное количество раз
Ucase(<строка>)	Преобразует в заданной строке все строчные буквы в прописные
Asc(<Символ>)	Возвращает код символа
Chr(<Код>)	Возвращает символ по его коду
Str(<Число>)	Преобразует заданное число в его строковое представление
Val(<Строка>)	Преобразует строку, представляющую собой число, в число

Таблица 11

Имя функции	Возвращаемое значение
Date	Возвращает значение типа Variant(Date), содержащее текущую системную дату
Time	Возвращает значение типа Variant(Date), содержащее текущее время по системным часам компьютера
Now	Возвращает значение типа Variant(Date), то есть текущую дату и время по системному календарю и часам компьютера
Hour, Minute, Second	Возвращает значение типа Variant(Date), содержащая целое число, представляющее часы, минуты и секунды в значении времени. Пример: ВР=#4:35:17 PM# Час = Hour(ВР) Минута = Minute(ВР) Секунда = Second(ВР)
Day, Month, Year	Возвращает значение типа Variant(Integer), содержащее целое число и представляющее день, месяц и год в значении даты. Синтаксис: Day(Дата), Month(Дата), Year(Дата)

Таблица 12

Имя функции	Возвращаемое значение
Iif	Возвращает одну из альтернатив. Синтаксис: Iif(expr, truepart, falsepart), где: expr – проверяемое значение; truepart – возвращаемое значение или выражение, если expr имеет значение true; falsepart – возвращаемое значение или выражение, если expr имеет значение false
Choose	Возвращает значение, выбранное из списка аргументов. Синтаксис: Choose(индекс, вариант_1, вариант_2, …, вариант_n). Функцию Choose можно использовать для выбора одного из возможных значений, представленных в виде списка.

Операторы альтернативы (ветвления)

Как и в любом другом языке программирования, в VBA можно проверять условия и выполнять действия в соответствии с результатами проверки этих условий. Для данной цели применяются следующие операторы (инструкции) принятия решения, позволяющие организовать в программе ветвление.

Условный оператор

IF <условие> THEN <оператор (код)>

Такая языковая конструкция позволяет выполнить один или несколько операторов в случае истинности проверяемого условия. Применяется однострочный или блочный вариант записи условного оператора. Если необходимо выполнить более одной строки кода, нужно использовать блочный вариант с ключевым словом End IF. С помощью такой инструкции реализуется базовый алгоритм неполного ветвления.

Синтаксис:

IF <условие> Then <оператор (код)>
IF <условие> Then
<блок кода>
End IF

Примеры:

```
IF x<10 Then z=0

IF x>10 Then
    z=2
    z=z+x
End IF
```

Инструкция, реализующая базовый алгоритм полного ветвления, позволяет определить два блока операторов. Первый выполняется, когда условие истинно, а второй, когда оно ложно.

Пример:

```
IF x<>0 Then
    y=Sin(x)/x
   Else
    y=1
End IF
```

Пример ветвления по четырем направлениям:

IF <условие 1> Then
<блок кода 1>
ElseIF <условие 2> Then
<блок кода 2>
ElseIF <условие 3> Then
<блок кода 3>

Else <блок кода 4>

End IF

В блоке IF допускается любое количество предложений ElseIF, но ни одно из них не может находиться после предложения Else. Однако с точки зрения методологии структурного программирования уровень вложенности оператора IF не должен превышать трех.

Пример:

```
IF x=-1.57 Then
        y=-1
       ElseIF x=0 Then
              y=0
             ElseIF x=1.57 Then
                    y=1
                   Else y=Sin(x)
End IF
```

Оператор выбора

При выборе для выполнения одного из нескольких операторов (блоков операторов) целесообразно и удобно использовать инструкцию Select Case. С помощью этого оператора в языке реализована алгоритмическая конструкция множественного выбора.

Синтаксис:

```
Select Case<переменная или выражение>
                Case <значение 1>
                        <оператор (блок операторов) 1>
                 Case <значение 2>
                        <оператор (блок операторов) 2>
                 Case <значение 3>
                        <оператор (блок операторов) 3>
End Select
```

Пример использования оператора выбора варианта в подпрограмме-функции:

```
Function PR(ByVal S As Single,
        ByVal P As Integer)As Single
         Select Case P
                        Case 0
                                PR=S*0
                        Case 1
                                PR=S*0.10
                        Case 2
                                PR=S*0.15
                        Case 3
                                PR=S*0.20
         End Select
End Function
```

Допускается вложенность инструкций Select Case. При этом каждой вложенной инструкции Select Case должна соответствовать инструкция End Select.

Операторы циклов

DO … LOOP

С помощью такой инструкции в языке реализована базовая алгоритмическая конструкция повторения, и она позволяет многократно выполнить любой блок кода.

Существует несколько вариантов записи этого оператора, но в каждом из них проверяется условие, управляющее работой цикла, и по результатам проверки определяется необходимость его продолжения. Результатом вычисления логического выражения (управляющего условия) будет одна из констант True или False.

Циклы с предусловием

DO WHILE <условие> … LOOP

Оператор DO WHILE <условие> … LOOP позволяет проверить условие перед началом цикла и выполнять цикл до тех пор, пока оно имеет значение True. Как только условие, управляющее рабо-

той цикла, примет значение False, выполнение тела цикла прекратится.

Пример:

```
Dim X As Integer 'Описание переменной X целого типа
X=0 ‘Начальное значение переменной X
DO WHILE X<=10 'Пока X<=10, цикл будет повторяться
  X=X+1 'Изменение значения переменной X
LOOP 'Конец цикла
```

Другой вариант записи инструкции такого цикла:

WHILE <условие>… WEND

Пример:

```
X=0
WHILE X<12
  Y=Cos(X)
  X=X+1
WEND
```

DO UNTIL <условие> … LOOP

Оператор Do Until <условие> … Loop позволяет проверить условие перед началом цикла и выполнять тело цикла до тех пор, пока условие принимает значение False. Как только управляющее условие примет значение True, выполнение тела цикла прекратится.

Пример:

```
Dim X As Integer 'Описание переменной X целого типа
X=0 'Начальное значение переменной X
Do Until X>10 'До тех пор, пока X<=10, цикл повторяется
  X=X+1 'Изменение значения переменной X
Loop 'Конец цикла
```

Циклы с постусловием

DO … LOOP WHILE <условие>

Если операторы цикла необходимо выполнить хотя бы раз, то для этой цели нужно использовать цикл с постусловием. Инструкция Do … Loop While <условие> позволяет проверить условие после выполнения операторов тела цикла. Цикл будет повторяться до тех пор, пока выражение в условии цикла имеет значение True.

Как только условие цикла примет значение False, выполнение тела цикла прекратится.

Пример:

```
Dim X As Integer
X=0
Do
  X=X+1
Loop While X<=10 'До тех пор, пока X<=10, цикл повторяется
```

DO … LOOP UNTIL <условие>

В отличие от предыдущего этот цикл будет выполняться до тех пор, пока значение управляющего условия равно False.

Пример:

```
Dim X As Integer
X=0
Do
  X=X+1
Loop Until X>10 'Как только переменная станет больше десяти, выполнение цикла прекратится
```

Цикл по счетчику

FOR … NEXT

Цикл с определенным количеством повторений выполняется от начального до конечного значения параметра с заданным шагом.

Пример:

```
Dim X As Integer
For X=1 To 10 Step 1 'Повторять цикл для X,
        изменяющегося от 1 до 10 с шагом 1
  Beep 'Звук (тело цикла)
Next X 'Конец цикла
```

Exit For или Exit Do 'Досрочный выход из цикла

FOR EACH … NEXT

Цикл For Each … Next предназначен для перебора всех элементов из заданного массива или набора объектов.

Пример использования цикла для обработки массива:

```
Sub Mas6()
 Dim i As Integer
 Dim j As Integer
 Dim x(1 To 5, 1 To 5) As Single
 Dim S As Single
 Worksheets("Лист1").Activate
 For i=1 To 5
   For j=1 To 5
     x(i,j)=Cells(i,j)
   Next j
 Next i
 For Each e In x
   S=S+e
 Next e
 MsgBox("S=") & CStr(S)
End Sub
```

Вложенные циклы

Совокупность простых циклов, вложенных один в другой, называется сложным (вложенным) циклом. При конструировании сложных циклов необходимо руководствоваться следующими правилами:

– нельзя войти во внутренний цикл, минуя вход внешнего цикла;

– имена параметров простых циклов не должны повторяться в конструкции сложного цикла;

– простые циклы не должны пересекаться в конструкции сложного цикла, то есть окончание внешнего цикла не должно предшествовать окончанию внутреннего цикла.

Примеры:

```
For i=1 to n
  For j=1 to m
    A(i,j)=Int(Sin(j*i)*100)
  Next j
Next i
```

```
Do
  X=1
  Z=0
  Do
    S=Int(Rnd(x)*100)
    Z=Z+S
    X=X+1
  Loop Until X>=20
  Zsr=Z/20
Loop Until Zsr>=25
```

Подпрограммы-процедуры и подпрограммы-функции

Подпрограмма – это блок кода между инструкциями Sub и End Sub или Function и End Function.

Подпрограмма-процедура – это блок кода, заключенный между инструкциями Sub и End Sub. Обычно подпрограмму-процедуру принято называть процедурой.

При написании программы нужно учесть одно правило: «Внутри одной процедуры не может быть описана другая процедура».

Синтаксис:

```
Sub <имя> (ByVal <параметр 1> As <тип>,
        ByVal <параметр 2> As <тип>,
        ByRef <параметр 3>,
        ByRef <параметр 4>)
  <блок кода процедуры>
End Sub
```

В скобках указываются необходимые параметры, если параметров нет, то просто пустые парные скобки. Например, напишем программу, выводящую на экран окно с приветствием:

```
Sub Программа_Привет()
  MsgBox("ПРИВЕТ")
End Sub
```

Параметры, указанные в скобках заголовка процедуры, называются *формальными*. Параметры, указанные в списке оператора вызова процедуры, называются *фактическими параметрами*.

Ключевые слова ByVal и ByRef определяют способ передачи значений параметров. ByVal используется для указания, что аргумент передается по значению. ByRef – аргумент передается по ссылке. Значения фактических параметров, передаваемых по способу ByVal, не могут изменяться в теле процедуры во время ее выполнения, то есть во время выполнения процедуры в программе сохраняются неизменными последние значения переменных. Значения фактических параметров, передаваемых по способу ByRef, изменяются в случае их изменения в вызываемой процедуре.

Вызов процедуры из другой процедуры можно произвести несколькими способами. Первый способ: <Имя процедуры> <Список фактических параметров>. Список должен соответствовать списку формальных параметров, заданному в заголовке процедуры, по количеству и типу.

Пример: *qwer x,y,s 'оператор вызова процедуры*

Если требуется использовать несколько процедур с одинаковыми именами, расположенными в разных модулях, то при их вызове перед именем процедуры через точку необходимо указывать имя модуля, в котором расположена процедура.

Синтаксис:

<Имя модуля>.<Имя процедуры> <Список
фактических параметров>

Второй способ вызова процедуры реализуется с помощью инструкции Call.

Синтаксис:

Call <Имя процедуры> (<Список фактических
параметров>)

В отличие от первого способа здесь список фактических параметров заключается в скобки.

Пример: *Call qwer(x,y,s)*

Подпрограмма-функция – это блок кода, заключенный между инструкциями Function и End Function. В ней может быть реализован любой алгоритм, но при этом функция обязательно возвращает какое-нибудь значение. Значение возвращается через имя функции.

Синтаксис:

```
Function <имя функции> (ByVal <параметр>
                As <тип>) As <Тип>
  <код функции>
End Function
```

Пример:

```
Function f(ByVal x As Single) As Single
  f=Sin(x^2)+Cos(3*x)
End Function
```

Оператор вызова функции состоит из имени функции и списка фактических параметров, заключенных в скобки.

Пример: y=f(x) 'Оператор вызова функции

При необходимости можно указать область видимости процедуры или функции.

Private Sub Программа_Привет() – закрытая процедура. Возможен вызов из модуля, где она находится, то есть подпрограмма доступна для других процедур только того модуля, в котором она описана.

Public Sub Программа_Привет() – открытая процедура. Возможен вызов из любого модуля, то есть подпрограмма доступна для всех других процедур во всех модулях.

Static Sub Программа_Привет() – указывает, что значения локальных переменных процедуры сохраняются в промежутках времени между вызовами этой процедуры.

Private Function f(ByVal x As Single, ByVal y As Single) As Single – закрытая функция. Возможен вызов из модуля, где она находится.

Public Function f(ByVal x As Single, ByVal y As Single) As Single – открытая функция. Возможен вызов из любого модуля.

Ввод-вывод с помощью диалоговых окон

В сокращенном варианте инструкции ввода-вывода имеют вид:

InputBox(<"сообщение">)
MsgBox(<"сообщение">)

Массивы

Массив – набор однотипных переменных с одним именем, каждая из которых называется элементом массива и имеет свой номер (индекс).

Массивы могут быть: одномерные (для нумерации элементов используется один индекс), двумерные (для нумерации элементов используются два индекса: номер строки, номер столбца) и *N*-мерные. Число измерений может достигать 60.

Кроме того, по способу выделения оперативной памяти для хранения элементов массивы подразделяются на статические и динамические.

Статические массивы

Статическим называется массив с заранее известным количеством элементов. Синтаксис описания (объявления) статического массива:

Dim <Имя массива>(<верхняя граница>) As <Тип>

По умолчанию значение нижней границы равно нулю.

Dim <Имя массива>(<Нижняя граница>
To <Верхняя граница>) As <Тип>

Примеры:

```
Dim a(10) As Single 'Одномерный массив
    с начальной границей, равной 0
Dim S(3 To 5) As String 'Одномерный массив
    с явно заданными границами
Dim Z(1 To 3, 1 To 5) As Byte 'Двумерный массив
```

Для задания по умолчанию нижней границы массива, равной 1, используется инструкции Option Base 1, которая задается в начале модуля.

Динамические массивы

Динамическим называется массив, размерность которого определяется в ходе выполнения программы. Синтаксис описания массива: Dim <Имя массива>() As <Тип>

Размерность массива устанавливается и изменяется с помощью инструкции ReDim <Имя массива>(<размерность>)

Для установки и изменения размерности массива без потери его содержимого применяется инструкция

ReDim Preserve<Имя массива>(<размерность>)

Для определения параметров динамического массива используются функции:

LBound(<Имя>[,<Размерность>])

Эта функция возвращает нижнюю границу указанной размерности массива.

Ubound(<Имя>[,<Размерность>])

Данная функция возвращает верхнюю границу указанной размерности массива. Если размерность не указана, то подразумевается значение, равное 1.

Array(<Список аргументов>)

С помощью такой инструкции создается массив типа Variant. Список аргументов представляет разделенный запятыми список значений, присваиваемых элементам массива.

Пример:

Dim День As Variant

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

IsArray(<Имя переменной>)

Эта функция используется для проверки факта, является ли переменная типа Variant массивом. Она возвращает значение True, если переменная является массивом, и False в противном случае.

Erase(<Список массивов>)

С помощью этой инструкции повторно инициализируются элементы статических массивов и освобождается память, отведенная для динамических массивов. Список представляет собой имена очищаемых массивов, разделенных запятой. В статических массивах их элементам вместо чисел присваиваются нулевые значения, а

строки переменной длины становятся пустыми. В массивах типа Variant каждому элементу присваивается значение Empty.

Основные объекты MS Excel

К числу основных объектов MS Excel, которые описываются в этом разделе, относятся следующие: рабочая книга (Workbook) и семейство рабочих книг (Workbooks), рабочий лист (Worksheet) и семейство рабочих листов (Worksheets), диапазон ячеек или ячейка (Range).

После объекта, обычно через точку «.», указывается свойство или метод. Иногда, чтобы добраться до определенного объекта, нужно пройти иерархию вышестоящих объектов.

Пример:

Workbooks("Книга1.xls").Worksheets("Лист1").Activate

Свойства и методы, которые обеспечивают ссылку на нужный объект в иерархии объектов, называются семействами (наборами).

Семейство WorkBooks("Книга1") обеспечивает доступ к рабочей книге. В скобках указывается имя книги.

Семейство WorkSheets("Лист1") обеспечивает доступ к рабочему листу. В скобках указывается имя листа.

Семейство Range(“диапазон”) обеспечивает доступ к диапазону ячеек или к ячейке. В скобках указывается диапазон ячеек или имя ячейки.

Семейство Cells(№ строки, № столбца) обеспечивает доступ к ячейке. В скобках указываются координаты ячейки.

Примеры:

WorkBooks("Книга1")
WorkSheets("Лист1")
Range("A1")
Range("A1:B10")
Cells(2,3)
Cells(k,i+1)

Объект Workbook и семейство Workbooks

В иерархии MS Excel объект Workbook (рабочая книга) идет сразу после объекта Application и представляет собой файл рабочей книги. Рабочая книга хранится либо в файлах формата XLS (стандартная рабочая книга) или XLA (полностью откомпилированное приложение). Свойства и методы рабочей книги позволяют работать с файлами. Этот объект входит в семейство (набор) Workbooks.

Ссылку на объект можно получить через следующие свойства.

Workbooks(<Индекс>) возвращает объект по индексу в наборе.

Workbooks("<Имя>") возвращает объект по имени в наборе.

ActiveWorkbook возвращает ссылку на активную книгу в момент выполнения команды.

ThisWorkbooks возвращает ссылку на книгу, в которой находится текст исполняемого модуля.

Свойства

ActiveSheet возвращает активный лист книги. Для получения имени листа используется свойство Name объекта Sheet.

Примеры:

MsgBox("Имя активного листа" & ActiveSheet.Name)

или

MsgBox(ActiveWorkbook.ActiveSheet.Name) – выводит имя активного рабочего листа в диалоговом окне.

ActiveDialog – возвращает активное диалоговое окно.

ActiveChart – возвращает активную диаграмму.

Sheets – возвращает семейство всех листов книги.

Worksheets – возвращает семейство всех рабочих листов книги.

Пример:

```
Sub xjfchd()
 For Each s In ActiveWorkbook.Sheets
          MsgBox (s.Name)
 Next s
End Sub
```

Charts – возвращает семейство всех диаграмм книги, которые не внедрены в рабочие листы.
Count – возвращает число объектов семейства Workbooks (количество открытых приложением книг).
FullName – возвращает полное имя рабочей книги.

Пример:

MsgBox (ActiveWorkbook.FullName)

Name – возвращает имя активной рабочей книги.

Пример:

MsgBox (ActiveWorkbook.Name)

Path – возвращает путь к файлу рабочей книги.

Пример:

MsgBox (ActiveWorkbook.Path)

Методы

Метод Activate активизирует рабочую книгу так, что ее первый рабочий лист становится текущим (доступным для работы).

Пример:

WorkBooks("Книга1").Activate

или

Workbook.Activate

Метод Close обеспечивает закрытие рабочей книги. Close SaveChanges FileName – закрывает книгу. Параметр SaveChanges позволяет управлять сохранением изменений в рабочей книге. Если его значение равно True, то изменения сохраняются, если – False , то не сохраняются. Параметр FileName – строка, содержащая имя файла рабочей книги.

Пример:

WorkBooks("Книга1").Close
WorkBooks("Книга1").Close SaveChanges:=True
Filename:= "Книга2"

New Window предназначен для открытия указанной книги в новом окне.

Пример:

WorkBooks("Книга1").NewWindow

Save служит для сохранения изменений в рабочей книге.

Пример:

WorkBooks ("Книга1").Save

SaveAs Filename – используется для сохранения книги под другим именем (в другом файле).

SaveAsCopy – предназначен для сохранения рабочей книги в другом файле, оставляя ее открытой с прежним именем.

Примеры:

WorkBooks ("Книга1").SaveAs FileName:= "kdjf.xls"
ActiveBook.SaveAsCopy FileName:= "Моя книга"

Open Filename – позволяет открыть рабочую книгу с именем, указанным в параметре FileName.

Пример:

Workbooks.Open FileName:= "Книга1.xls"

События

Основные события объекта Workbook приведены в табл. 13.

Таблица 13

Событие	Когда возникает событие
BeforeClose	При закрытии рабочей книги
BeforePrint	Перед печатью рабочей книги
BeforeSave	Перед сохранением рабочей книги
NewSheet	При добавлении нового листа
Open	При открытии рабочей книги
SheetActivate	При активизации рабочего листа

Объект Worksheet и семейство WorkSheets

В иерархии MS Excel объект Worksheet идет сразу после объекта Workbook, представляет рабочий лист книги и входит в семейство (набор) Worksheets.

Ссылку на объект можно получить с помощью команды Worksheets(Index). Она возвращает ссылку на объект по индексу в наборе, в качестве индекса может выступать имя листа или его номер в наборе.

Примеры:

Worksheets("Лист1").Activate
Worksheets(1).Activate

Activesheet – возвращает ссылку на активный лист.

Пример: *Activesheet.Range ("a1")=1*

Свойства

Свойство Name позволяет работать с именем рабочего листа.

Пример:

Worksheets(1).Name="Итоги"

ActiveCell – возвращает активную ячейку активного рабочего листа.

Cells – возвращает ссылку на диапазон ячеек листа.

Cells(<строка>,<столбец>) – возвращает ссылку на ячейку с указанными координатами.

Columns(<столбец>) – возвращает ссылку на столбец. В качестве параметра могут быть заданы имя или номер столбца.

Пример:

Worksheets(1).Columns("a")=1

или

Worksheets(1).Columns(1)=1

Rows(<строка>) – возвращает ссылку на строку. В качестве параметра может быть использован номер строки.

Пример:

Worksheets(1).Rows(1)=1

Range(<Диапазон ячеек>) – возвращает ссылку на указанный диапазон ячеек.

UsedRange – возвращает ссылку на используемый диапазон листа.

Пример:

Worksheets("Лист1").UsedRange.Value=1

Count – возвращает количество листов в книге.

Visible – определяет отображение рабочего листа в книге. Его допустимые значения:True – рабочий лист выводится на экран; False – рабочий лист невидим (скрыт), но его можно отобразить на экране с помощью последовательности команд *Формат, Лист, Отобразить (Format, Sheet, Show)*; xlVeryHidden – рабочий лист скрыт и его можно отобразить на экране только программно.

Примеры:

```
Sub Пусто()
    Worksheets("Лист3").Visible=False
End Sub
```

UsedRange – возвращает диапазон, то есть объект Range, содержащий данные.

Пример:

```
Worksheets(1).UsedRange.Clear ‘Очищается
```

диапазон первого рабочего листа с данными

Методы

Метод Activate позволяет активизировать рабочий лист.

С помощью метода Evaluate выражения преобразуются в объекты или в значения.

Пример:

```
Sub Ввод_формул()
  Dim Значение As Single
  Dim Fx As String
  Ячейка InputBox("Введите имя ячейки") 'Ввод
        адреса ячейки в диалоговом окне
        ЗначениеEvaluate(Ячейка)
  Value 'Считывание значения из ячейки
  MsgBox Cstr(Ячейка)& "="&CSng(Значение)
      ‘Вывод значения в диалоговое окно
  Fx=InputBox(“Введите формулу (функцию)”)
      ‘Ввод функции в диалоговом окне
  ЗначениеEvaluate(Fx) 'Вычисление значения
                функции
  MsgBox CStr (Ячейка)& "="&CSng(Значение)
      ‘Вывод значения в диалоговом окне
End Sub
```

Объект Range

В иерархии MS Excel объект Range (диапазон) идет сразу после объекта Worksheet и является одним из ключевых объектов VBA. Он не входит в состав никакого семейства объектов.

Объект Range описывает диапазон ячеек рабочего листа. При работе с ним имеются три способа записи ссылок (табл. 14) на ячейки рабочего листа: относительная адресация (начало координат, задающее нумерацию строк и столбцов, связывается с объектом, вызвавшим Range), абсолютная и смешанная адресация.

Таблица 14

Стили ссылок	Способы записи адресов ячеек
A1	Имя ячейки состоит из имени столбца (их 256 – от A до IV) и номера строки (от 1 до 65536). Пример: A1, C2 – *относительный адрес*
R1C1	Ссылка задается индексом строки и индексом столбца. Пример: R1C1, R2C2, R1C3 – *абсолютный адрес*
A1	Признаком абсолютной адресации является символ «$», предшествующий номеру строки (A$12 – *смешанный адрес*), имени столбца ($A12 – *смешанный адрес*), тому и другому (A12 – *абсолютный адрес*)
R1C1	Можно указать смещение по отношению к активной ячейке. Смещение приводится в квадратных скобках, причем знак указывает на направление смещения. Пример: R2C3 ⇒ R[1]C[-1] ⇒ R3C2

Записанный в таком виде адрес ячейки рабочего листа является лишь частью полного адреса, который в общем случае включает имя рабочего листа и имя файла рабочей книги. При задании полного адреса за именем листа следует знак «**!**», а имя файла рабочей книги заключается в квадратные скобки.

Примеры:

A1 – относительная ссылка на ячейку A1 активного рабочего листа.

Лист2!A1 – относительная ссылка на ячейку A1 рабочего листа Лист2 активной книги.

[ВсепроВсе.xls]Лист2!A1 – относительная ссылка на ячейку A1 рабочего листа Лист2 книги ВсепроВсе.xls текущего рабочего каталога.

Если в диапазоне указываются только имена столбцов или строк, то объект Range задает диапазон, состоящий из указанных столбцов или строк.

Примеры:

Range("A:C") **–** задает диапазон столбцов A, B, C.

Range("2:4") **–** задает диапазон строк 2, 3, 4.

Так как ячейка является частным случаем диапазона, то объект Range позволяет также работать и с ней. Альтернативным способом работы с ячейкой является объект Cells (ячейки).

Свойства

Свойство Formula позволяет установить формулу в ячейке. Формула задается в виде строки.

FormulaArray – устанавливает формулу массива ячеек. Формула задается в виде строки. В качестве ссылок на ячейки используется стиль A1. Формула массива – это формула, которая в качестве исходных данных использует диапазон ячеек и возвращает одно или несколько значений.

Пример:

```
Sub Prima1()
  With Worksheets("Лист1")
   For i = 1 To 3
     For j = 1 To 3
       Cells(i,j) = Int(Rnd(i*j)*100)
     Next j
   Next i
   Range("D1:F3").FormulaArray="=MINVERSE(A1:C3)"
  End With
End Sub
```

FormulaR1C1 – устанавливает формулу в ячейке. Формула задается в строковом виде стилем ссылок R1C1.

Пример:

```
Sub Prima1()
  With Worksheets(“Лист1”)
   For i = 1 To 3
     For j = 1 To 3
       Cells(i,j) = Int(Rnd(i*j)*100)
     Next j
   Next i
   Range("D1:F3").FormulaR1C1="=MINVERSE(R1C1:R3C3) "
  End With
End Sub
```

Свойство HasArray возвращает значение True, если указанная ячейка является частью массива. Массивом на рабочем листе называется именованный диапазон ячеек.

Свойство HasFormula возвращает значение True, если в указанной ячейке содержится формула.

Примеры:

MsgBox Worksheets(1).Range("A2").HasArray – выводимое значение True или False

MsgBox Worksheets(1).Range("A3").HasFormula – выводимое значение True или False

Свойство Text возвращает содержимое ячейки в виде строки. Используется только для чтения.

Пример:

```
MsgBox Worksheets(1).Range("A2").Text
```

Свойство Value возвращает значение из ячейки или устанавливает значение в ячейку (в ячейки).

Примеры:

x=Range("C1").Value – значение из ячейки C1 присваивается переменной x

Range("A1:B4").Value=12 – в диапазон ячеек A1:B4 устанавливается число 12

Инструкция With используется для указания текущего объекта. Внутри ее можно указывать, начиная с точки, только свойства и методы при обращении к текущему объекту.

Синтаксис:

```
With <объект>
  .<свойства и методы>
End With
```

Пример:

```
Sub qwe()
   Dim a As Single
   Dim b As Single
   With Worksheets("Лист1")
     a=.Range("A1").Value
     b=.Range("B1").Value
     .Range("C1").Value=a+b
```

```
    .Range("D1").Formula= "=A1+B1"
  End With
End Sub
```

Команда Set предназначена для закрепления объекта за переменной. Переменная должна быть типа Object или с типом объекта, который за ней будет закреплен.

Примеры:

```
Sub Prima1()
  Dim Lst As Object
  Set Lst=Workbooks("Книга1.xls").Worksheets("Лист1")
          'За переменной Lst закрепляется
          рабочий Лист1 рабочей Книги1
  Lst.Cells(1,1)=10  'В ячейку А1 рабочего Листа1 устанавливается значение 10
End Sub
```

Методы

Метод Activate позволяет активизировать указанный диапазон ячеек.

С помощью метода AddComment <текст примечания> добавляются примечания к ячейке (пункт *Добавить примечание* контекстного меню).

Методом AutoFill <источник> производится автозаполнение диапазона данными из указанных ячеек.

Пример:

```
Sub Prima3()
  Worksheets(1).Activate
  For i = 1 To 10
    Cells(i,1)=i
  Next i
  Range("B1").Formula= "=sin(A1) "
  Range("B1").AutoFill Range("B1:B10")
End Sub
```

Метод Clear очищает указанный диапазон ячеек.

ClearContents – очищает формулы и значения, содержащиеся в ячейках, представляемых объектом Range. При этом очищается только содержимое ячеек, форматирование же их сохраняется.

Метод ClearFormats удаляет все форматирование ячеек, сохраняя неизменными хранящиеся в них данные.

Метод Sort Key1:=<ячейка> Order1:=<порядок> Orientation:=<направление> позволяет осуществить сортировку указанного диапазона ячеек. Key1 – ключ (строка или столбец), по которому будет производиться сортировка. Order1 – задает порядок сортировки и может принимать два значения: xlAscending – по возрастанию (в алфавитном или хронологическом порядке); xlDescending – по убыванию (в обратном алфавитному или в ретроспективном порядке). Orientation – указывает направление сортировки: xlSortRows – сортировка данных в строке; xlSortColumns – сортировка в столбце.

2. ОБРАТНАЯ ГЕОДЕЗИЧЕСКАЯ ЗАСЕЧКА

Сущность обратной геодезической засечки состоит в определении координат дополнительной точки *M* путем измерения на этой точке углов между направлениями как минимум на три исходных пункта с известными координатами. На практике для получения надежного контроля и повышения точности определения координат искомой точки используют многократную обратную засечку не менее чем по четырем исходным пунктам (рис. 1). В этом случае решение обратной засечки выполняют независимо по двум комбинациям исходных пунктов. В нашем случае первой комбинацией является треугольник «дом-ель-водокачка», второй – «ель-водокачка-волейбол».

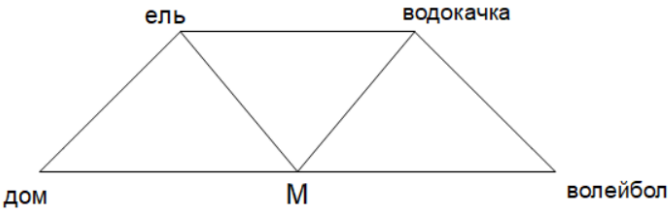

Рис. 1. Схема обратной геодезической засечки

Координаты вычисляются по формулам Юнга:

$$tgO = \frac{(Y_B - Y_C) * ctg\beta_2 - (Y_A - Y_B) * ctg\beta_1 + X_C - X_A}{(X_B - X_C) * ctg\beta_2 - (X_A - X_B) * ctg\beta_1 - Y_C + Y_A} \quad (1)$$

$$N = (Y_B - Y_C)(ctg\beta_2 - tg\theta) - (X_B - X_C)(1 + ctg\beta_2 tgO) \quad (2)$$

$$X = \frac{N}{1 + tg^2 O} \quad (3)$$

$$\Delta Y = \Delta X * tgO \quad (4)$$

$$X_p = X_в + \Delta X \quad (5)$$

$$Y_p = Y_в + \Delta Y \quad (6)$$

В решении задачи были задействованы такие процессы, как линейный, разветвляющийся и циклический.

РЕШЕНИЕ ЗАДАЧИ В MICROSOFT OFFICE EXCEL

Решение задачи в Microsoft Office Excel начинается с ввода исходных данных в ячейки на Лист 1 (рис. 2-3).

	A	B	C	D	E	F
1	№ пункта	Пункт	Угол	Угол		
2				градусы	минуты	секунды
3	1	Дом				
4			β1	56	12	54
5	2	Ель				
6			β2	63	49	46
7	3	Водокачка				
8			β3	69	9	1
9	4	Волейбол				
10						

Рис. 2. Исходные данные

13	Проверка ввода допустимых значений углов:			
14		градусы	минуты	секунды
15	β1	56	12	54
16	β2	63	49	46
17	β3	69	9	1

Рис. 3. Проверка ввода

Вспомним геометрию обратной геодезической засечки и условия, которые накладываются на значения градусов (30-150), минут (0-60) и секунд (0-60). Создадим проверку этих условий с помощью формул «*=ЕСЛИ (И (D4<150; D4>30); D4;"нет решения")*» для углов и «*=ЕСЛИ (И (E4<600; E4>0); E4;"нет решения")*» для минут и секунд.

Необходимо минуты и секунды сделать долями градусов, для этого следует использовать следующую формулу: «*=D4+E4/60+F4/3600*» (для угла $\beta 1$). Затем следует перевести зна-

чения углов из градусной в радианную меру для дальнейших вычислений (рис. 4). Сделать это можно с помощью такой встроенной функции, как «*РАДИАНЫ*». Для угла β1 формула выглядит вот так: «*=РАДИАНЫ(G4)*». Затем вычисляем *ctg(β1)* по формуле «*=1/TAH(H4)*». Те же действия выполняем и для других углов.

D	E	F	G	H	I
Угол			Значение угла	Значение угла(рад)	ctg угла
градусы	минуты	секунды			
56	12	54	56,215	0,981	0,669
63	49	46	63,829	1,114	0,491
69	9	1	69,150	1,207	0,381

Рис. 4. Значения углов в радианной мере и котангенсы углов

Затем вычисляем такие промежуточные величины (рис. 5), как *tg(O)* и *N*. Для того, чтобы не запутаться в значениях, посчитаем отдельно числитель и знаменатель *tg(O)*. Числитель вычисляется по формуле «*=(K5-K7)*I6-(K3-K5)*I4+J7-J3*», знаменатель «*=(J5-J7)*I6-(J3-J5)*I4-K7+K3*» для первого треугольника.

20	№	Числитель	Знаменатель	tgΘ	N
21	1	-37,879	31,235	-1,21	54,06
22	2	22,084	92,146	0,24	32,11
23					

Рис. 5. Промежуточные значения

В ячейках *K*, *J* находятся координаты исходных точек (рис. 6).

ΔX	ΔY	X	Y
21,88	-26,53	115423,5	121272,2
30,37	7,28	115423,5	121272,2

Рис. 6. Вычисление приращений и координат

Следующим шагом вычисляются приращения координат *dX*, *dY* и самих координат.

На этом этапе необходимо сравнить значения координат, полученные из двух треугольников. Если они отличаются более, чем на 0.3 м, то в вычислениях имеется ошибка. В нашем случае значения по *X* и *Y* не отличаются (рис. 7). Следовательно, вычисления выполнены верно.

X	Y
115423,5	121272,2
115423,5	121272,2
115423,5	121272,2

Рис. 7. Конечные значения координат

РЕШЕНИЕ ЗАДАЧИ В VISUAL BASIC FOR APPLICATIONS

На этом этапе требуется создать программу для вычисления координат искомой точки *M* путем ввода исходных координат четырех точек и значений градусов, минут и секунд для трех углов.

Для начала создадим форму (рис. 8) для ввода данных (*Insert – Userform*).

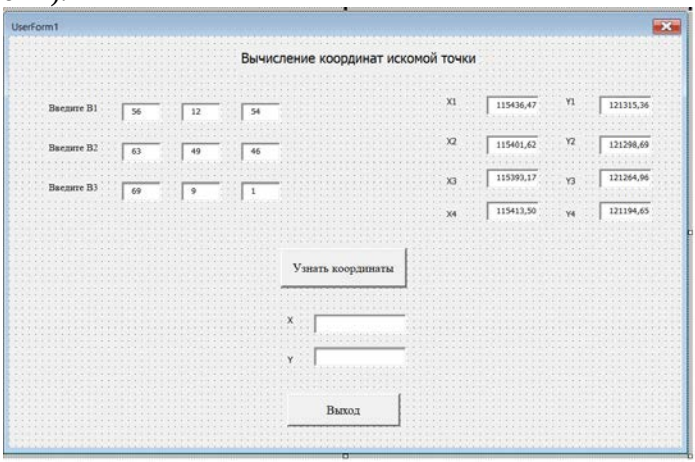

Рис. 8. Основная форма

Кликнув дважды на кнопку «*Узнать координаты*», мы перейдем в модуль, в котором необходимо написать программу для вычисления координат. Первым действием напишем команду Option Explicit, которая осуществляет проверку ввода переменных, для того, чтобы не забыть объявить какую-либо переменную. Следующим шагом является описание всех переменных, которые будут использоваться при написании кода (рис. 9).

CommandButton1

```
Option Explicit 'проверка объявления переменных
Private Sub CommandButton1_Click()
' Объявление переменных
' Численная переменная arrayG - массив размерностью 3*3
Dim arrayG(1 To 3, 1 To 3) As Single
' Численная переменная arrayG - массив размерностью 3*3
Dim arrayG1(1 To 3, 1 To 3) As Single
' Численная переменная arrayG - массив размерностью 4*2
Dim arrayK(1 To 4, 1 To 2) As Single
' Численные переменные B1, B2, B3 для хранения значений углов
Dim B1 As Single, B2 As Single, B3 As Single
' Численные переменные ctg1, ctg2 для хранения значений котангенсов углов
Dim ctg1 As Single, ctg2 As Single, ctg3 As Single
' Численные переменные tg01, tg02 для хранения значений тангенсов O (тета)
Dim tg01 As Single, tg02 As Single
' Численные переменные N1, N2 для хранения значений некоторой переменной
Dim N1 As Single, N2 As Single
' Численные переменные dX, dY для хранения значений приращений по X и Y
Dim dX1 As Single, dX2 As Single
Dim dY1 As Single, dY2 As Single
' Численные переменные Xo, Y0 для хранения значений координат
Dim Xo1 As Single, Xo2 As Single
Dim Yo1 As Single, Yo2 As Single
' Численные переменные x, y для хранения усредненных значений координат
Dim x As Single, y As Single
'Переменная i целого типа для перечисления индексов массивов
Dim i As Integer
```

Рис. 9. Описание переменных

Все переменные (кроме *i*) описаны как *Single*, потому что этот тип позволяет использовать дробные числа, но не перезагружает код так, как это сделал бы тип *Double*. Для описания переменной *i* мы используем тип Integer, потому что эта переменная будет использоваться для перечисления индексов массивов, и, следовательно, она должна быть целым числом.

```
CommandButton1

'Запись массива углов из textbox
'Присвоение значений переменной i
 For i = 1 To 3
 arrayG(i, 1) = CSng(UserForm1.Controls("TextBox" & CStr(0 + i)))
 arrayG(i, 2) = CSng(UserForm1.Controls("TextBox" & CStr(3 + i)))
 arrayG(i, 3) = CSng(UserForm1.Controls("TextBox" & CStr(6 + i)))
 Next
 'Запись массива координат из textbox
 'Присвоение значений переменной i
 For i = 1 To 4
 arrayK(i, 1) = CSng(UserForm1.Controls("TextBox" & CStr(9 + i)))
 arrayK(i, 2) = CSng(UserForm1.Controls("TextBox" & CStr(13 + i)))
 Next
```

Рис. 10. Запись значений в массивы

Для удобства считывания данных с UserForm1 создадим цикл (рис. 10) для считывания значений градусов, минут, секунд, а также цикл (рис. 11) для считывания значений координат. Данные углов будут записаны в массив *arrayG*, а координаты в массив *arrayK*. Функция *Controls* требуется для того, чтобы указать на конкретный элемент управления, в нашем случае – это *TextBox*. С помощью функции *Cstr* считаем значения из нужных *TextBox*, так как по умолчанию там стоит значение *String*. С помощью функции *CSng* преобразуем считываемые значения в тип *Single*.

Далее требуется проверить условия геометрии. Необходимо, чтобы значения градусов (*arrayG(i,1)*) были в диапазоне от 30 до 150, значения минут (*arrayG(i,2)*) и секунд (*arrayG(i,3)*) в диапазоне от 0 до 60. Для этого создадим циклы (Приложение А). Первая строчка цикла присваивает значения для переменной i, вторая задает условие, третья говорит о том, что будет, если истинность условия нарушена. Четвертая говорит об окончании выполнения условия, пятая заканчивает цикл.

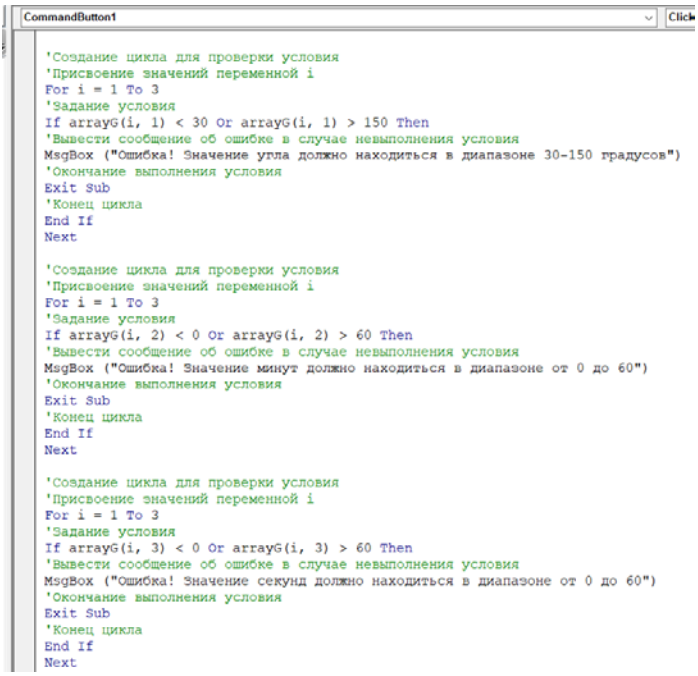

Рис. 11. Проверка условия

В случае несоблюдения какого-либо условия функция *MsgBox* должна выведет на экран сообщение об ошибке.

Следующим этапом переводим значения углов из градусной меры в радианную. Для этого значения градусов требуется умножить на π и разделить на 180. Функция числа π в VBA представлена, как *Atn(1)*4*. Как известно из геометрии, $tg(\pi/4) = 1$. Соответственно, $\pi/4=arctg(1)$, а $\pi=4arctg(1)$. После перевода (рис. 12) суммируем элементы одной строки для получения значения угла.

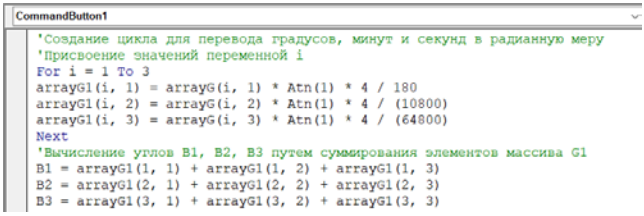

Рис. 12. Вычисление значений углов в радианной мере

Далее мы вычисляем промежуточные величины (рис. 13) такие, как *ctg(β)*, который вычисляется с помощью функции обратной тангенсу. При вычислении *tg(O)* и *N* используем массив координат *arrayK*.

```
'Вычисление котангенсов углов
ctg1 = 1 / Tan(B1)
ctg2 = 1 / Tan(B2)
ctg3 = 1 / Tan(B3)
'Вычисление значений tgO
tg01 = ((arrayK(2, 2) - arrayK(3, 2)) * ctg2 - (arrayK(1, 2) - arrayK(2, 2)) * ctg1 + (arrayK(3, 1) - arrayK(1, 1))) /
tg02 = ((arrayK(3, 2) - arrayK(4, 2)) * ctg3 - (arrayK(2, 2) - arrayK(3, 2)) * ctg2 + (arrayK(4, 1) - arrayK(2, 1))) /
'Вычисление значений N
N1 = (arrayK(2, 2) - arrayK(3, 2)) * (ctg2 - tg01) - (arrayK(2, 1) - arrayK(3, 1)) * (1 + ctg2 * tg01)
N2 = (arrayK(3, 2) - arrayK(4, 2)) * (ctg3 - tg02) - (arrayK(3, 1) - arrayK(4, 1)) * (1 + ctg3 * tg02)
```

Рис. 13. Вычисление промежуточных значений

Вычисляем приращения координат *dX, dY* и значения координат из двух треугольников (рис. 14).

```
'Вычисление приращений dX и dY
dX1 = N1 / (1 + (tg01) ^ 2)
dX2 = N2 / (1 + (tg02) ^ 2)
dY1 = dX1 * tg01
dY2 = dX2 * tg02
'Вычисление координат путем суммирования исходных координат и вычисленных приращений
Xo1 = arrayK(2, 1) + dX1
Xo2 = arrayK(3, 1) + dX2
Yo1 = arrayK(2, 2) + dY1
Yo2 = arrayK(3, 2) + dY2
```

Рис. 14. Вычисление приращений и координат

Последним действием мы усредняем полученные координаты, округляем их до десятых с помощью функции *Format* и записываем в *TextBox18* и *TextBox19* (рис. 15).

```
'Усреднение координат
x = (Xo1 + Xo2) / 2
y = (Yo1 + Yo2) / 2
'Запись полученных значений x и y в форму
'Округление до десятых
TextBox18 = Format(x, "0.0")
TextBox19 = Format(y, "0.0")
End Sub
```

Рис. 15. Получение конечных координат

Кнопка «Выход» на Форме осуществляет выход из формы с помощью функции *Unload* (рис. 16).

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

Рис. 16. Кнопка «Выход»

Полученные нами результаты расходятся на 0.1 м по оси *Y*, что является допустимым в условиях обратной геодезической засечки (рис. 17).

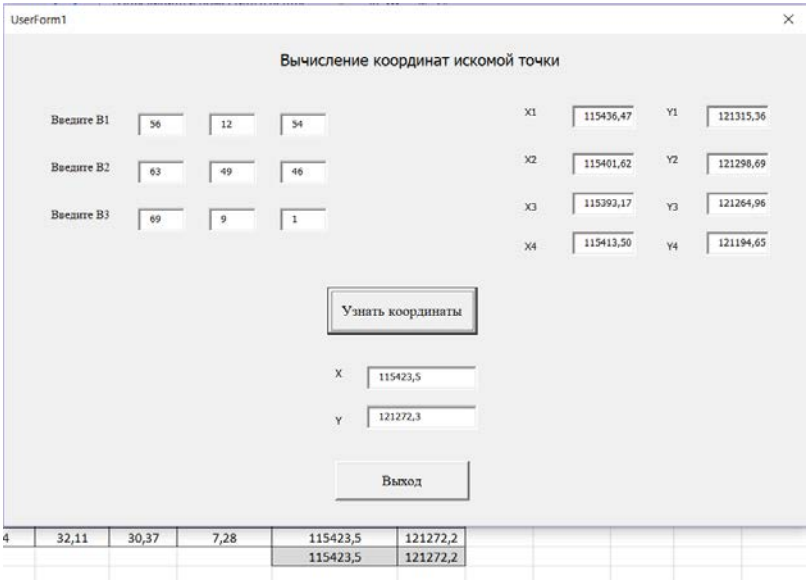

Рис. 17. Сравнение результатов в Microsoft Excel и VBA

ЗАДАНИЕ ДЛЯ ЛАБОРАТОРНОЙ РАБОТЫ

Подобрать индивидуальную задачу, связанную с профессиональной деятельностью, и согласовать с преподавателем. Задача должна включать возможность использовать условия и циклы, построение графиков. Реализовать решение задачи в MS Excel и VBA.

Отчет о выполнении работы должен представлять собой документ MS Word. Обязательно приводятся условие задач и исходные данные.

Все расчеты и результаты должны быть представлены как в режиме представления данных, так и в режиме представления формул. В заключение дается анализ полученных результатов с подробными пояснениями.

Для защиты работы необходимо знать ответы на контрольные вопросы и уметь сделать выводы по полученным результатам.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите опцию, требующую явного описания всех используемых переменных и констант.
2. С помощью какого ключевого слова объявляются переменные и константы, сохраняющие свои значения до конца выполнения программы?
3. С помощью каких ключевых слов определяется область действия переменных?
4. Какого типа переменной могут присваиваться числовое или строковое значения?
5. Где хранится код программы во время ее выполнения?
6. С помощью какого ключевого слова осуществляется явное определение типа переменной?
7. С помощью какой функции можно вычислить длину строки?
8. Какое имя у функции, выделяющей целую часть вещественного числа с округлением?

РЕКОМЕНДУЕМЫЙ БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Грошев, А.С. Информатика [Электронный ресурс]: учебник для вузов / А.С. Грошев; Берлин: Директ-Медиа. – Москва, 2015. – 484 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=428591. – Загл. с экрана.

2. Грошев, А.С. Информационные технологии [Электронный ресурс]: лабораторный практикум / А.С. Грошев; Берлин: Директ-Медиа. – Москва, 2015. – 285 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=434666. – Загл. с экрана.

3. Уокенбах, Джон. Microsoft Excel. Библия пользователя. – М.: ООО «ИД Вильямс», 2015. – 1040 с.

4. Головин, Г.А. Геодезия. Топографические съёмки: Методические указания к учебной практике по геодезии // Г.А. Головин, Ю.Н. Корнилов, А.А. Боголюбова / Санкт-Петербургский горный университет. – СПб, 2016. – 81 с.

СОДЕРЖАНИЕ

ИНФОРМАТИКА И ИНФОРМАЦИОННО-КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

РЕШЕНИЕ ИНЖЕНЕРНЫХ ЗАДАЧ СРЕДСТВАМИ MS EXCEL И VBA

Методические указания к лабораторным работам для студентов бакалавриата направления 22.03.01

Сост.: *А.А. Кочнева, Т.В. Сарапулова*

Печатается с оригинал-макета, подготовленного кафедрой информатики и компьютерных технологий

Ответственный за выпуск *А.А. Кочнева*

Лицензия ИД № 06517 от 09.01.2002

Подписано к печати 28.05.2020. Формат 60×84/16.
Усл. печ. л. 2,7. Усл.кр.-отт. 2,7. Уч.-изд.л. 2,5. Тираж 75 экз. Заказ 328. С 34.

Санкт-Петербургский горный университет
РИЦ Санкт-Петербургского горного университета
Адрес университета и РИЦ: 199106 Санкт-Петербург, 21-я линия, 2