Министерство науки и высшего образования Российской Федерации

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

Кафедра информатики и компьютерных технологий

ИНФОРМАТИКА

АЛГОРИТМИЗАЦИЯ И ОСНОВЫ ПРОГРАММИРОВАНИЯ

Методические указания к лабораторным работам

САНКТ-ПЕТЕРБУРГ
2021

УДК 681.142.2 (073)

ИНФОРМАТИКА. Алгоритмизация и основы программирования: Методические указания к лабораторным работам / Санкт-Петербургский горный университет. Сост.: *Т.В. Сарапулова, О.В. Косарев, А.А. Кочнева,* СПб, 2020. 72 с.

Методические указания предназначены для выполнения лабораторных работ в рамках изучения дисциплины «Информатика». В состав указаний входят теоретический материал, описание технологии решения задач средствами VBA, приведены задания для выполнения лабораторных работ, список рекомендуемой литературы, контрольные вопросы для самопроверки.

Научный редактор доц. *А.Б. Маховиков*

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

ВВЕДЕНИЕ

Цель работы – изучить основы алгоритмизации и возможности языка программирования Visual Basic for Applications для решения различного рода 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.

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.

Любой программист должен стремиться составить надежную программу. Надежная программа – это программа, которую легко понять и она может работать без особых проблем.

Составление надежной программы – результат больших усилий и опыта, что потребует много времени и упражнений. Для облегчения такой работы направлены некоторые рекомендации:

– обязательно объявлять переменные явно, для этого в начале каждого программного модуля включать инструкцию *Option Explicit* (все переменные объявлять явно). Это простое действие, возможно, является самым важным в написании надежной программы, потому что предотвращает все досадные ошибки и дефекты, которые происходят из-за неправильно написанных названий переменных и их, как правило, так тяжело выявить;

– не использовать без крайней необходимости тип *Variant*, так как он иногда является причиной некоторых трудноуловимых ошибок в программе;

– не использовать без крайней необходимости тип *Object*, так как он может содержать ссылку на любой тип объектов, а эта гибкость может быть источником проблем. При всякой возможности

объектные переменные объявлять, как конкретный тип, на который они будут ссылаться;

– все объявления переменных помещать в начало процедуры (модуля) и располагать их по одному объявлению на строку;

– не писать в одной строке несколько операторов, даже если возможно поместить в одной строке несколько операторов через двоеточие, потому что такую программу будет затруднительно читать;

– проверять корректность данных, получаемых процедурами в качестве аргументов и введенных пользователями. Прежде чем использовать данные в программе их необходимо проверять, для этого целесообразно использовать, например, функции проверки типов (IsNumeric, IsArray и другие).

Следуя этим рекомендациям можно гарантированно сократить время разработки и отладки программы.

Возможные типы ошибок

Все возникающие в программе ошибки можно разделить на три основных типа:

– ошибки выполнения (*runtime errors*), возникающие в процессе выполнения программы. Если не предусмотрен механизм их перехвата, то происходит аварийная остановка программы с выдачей стандартного описания такой ошибки;

– логические ошибки (*bugs*), приводящие к неправильной работе программы, выражающиеся в получении неправильных результатов. Ошибки такого типа не приводят к аварийной остановке программы. Устранение таких ошибок возможно лишь исправлением алгоритма программы, который должен разрабатываться программистом на одном из первых этапов решения поставленной задачи;

– синтаксические ошибки (*syntax errors*), возникающие в процессе набора текста программы при нарушении правил синтаксиса языка VBA. Каждая строка текста программы в процессе ее набора анализируется редактором.

– рекомендуется набирать текст программы в нижнем регистре, тогда после перехода на следующую строку редактора введенный текст изменится в соответствии с синтаксисом языка VBA (не-

которые буквы отобразятся в верхнем регистре). Если этого не произошло, необходимо проверить набранную строку программы.

ЛАБОРАТОРНАЯ РАБОТА № 1.
АЛГОРИТМИЗАЦИЯ РЕШЕНИЯ ЗАДАЧ

Цель работы: усвоить понятия: алгоритм как фундаментальное понятие информатики, способы описания, основные типы алгоритмов, освоить принципы решения задач с использованием основных алгоритмических конструкций.

Теоретические сведения

Алгоритм – понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение цели.

Любой алгоритм должен обладать следующими **свойствами:**

– **определенностью** – за конечное число шагов либо должен быть получен результат, либо доказано его отсутствие;

– **результативностью** – обязательным получением некоторого результата (числа, таблицы, текста, звука, изображения и т. д.) или сигнала о том, что данный алгоритм неприменим для решения поставленной задачи;

– **массовостью** – возможностью получения результата при различных исходных данных для некоторого класса сходных задач;

– **формальностью** – отвлечение от содержания поставленной задачи и строгое выполнение некоторого правила, инструкции;

– **дискретностью –** возможностью разбиения алгоритма на отдельные элементарные действия.

Существуют следующие **формы представления алгоритма**:

– словесная (вербальная) на неформальном языке;

– на языках программирования;

– графическая.

Словесная форма представления алгоритма имеет ряд недостатков. Для сложных алгоритмов описание становится слишком громоздким и не наглядным. Эта форма представления обычно используется лишь на начальных стадиях разработки алгоритма.

Алгоритм, записанный на **языке программирования**, называется *программой*.

Графическая форма представления алгоритмов является более наглядной и строгой. Алгоритм изображается в виде последовательности связанных между собой блоков, каждый из которых

соответствует выполнению одного или нескольких операторов. Такое графическое представление называется **блок-схемой алгоритма.**

Условные графические обозначения символов, используемых для составления блок-схемы алгоритма, стандартизированы.

Некоторые, часто используемые обозначения, приведены в рисунке 1.

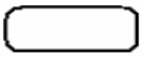
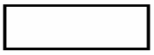
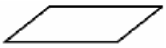
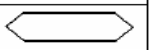
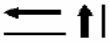
Название блока	**Обозначение**	**Название блока**	**Обозначение**
Начало или конец алгоритма		Решение	
Процесс (действие или серия действий)		Предопределенный процесс (вспомогательный алгоритм)	
Ввод/вывод данных		Модификация (заголовок цикла)	
Линии потока		Комментарии	

Рис. 1. Условные графические обозначения символов

Отдельные блоки алгоритмов соединяются между собой линиями потоков, которые проводятся параллельно внешней рамке чертежа. Направления линий потока сверху вниз и слева направо принимаются за основные и, если линии потоков не имеют изломов, стрелками не обозначаются. Обратные направления линий потока помечаются стрелкой.

«Процесс» (этап вычисления) изображается прямоугольником, внутри которого записывается набор действий. Ромбом изображается «решение», внутри которого осуществляется проверка условия. Ввод исходных данных и вывод результатов изображаются параллелограммами, внутри которых пишутся слова «ввод» или «вывод» и перечисляются переменные, подлежащие вводу или выводу.

Представление алгоритма в виде блок-схемы является промежуточным, так как алгоритм в таком виде не может быть непосредственно выполнен ЭВМ, но помогает пользователю при создании (написании) программы для ПК.

Использование блок-схем дает возможность:

– наглядно отобразить базовые конструкции алгоритма;

– сосредоточить внимание на структуре алгоритма, а не на синтаксисе языка;

– анализировать логическую структуру алгоритма;

– преобразовывать алгоритм методом укрупнения (сведения к единому блоку) или детализации – разбиения на ряд блоков;

– использовать принцип блочности при коллективном решении сложной задачи;

– осуществить быструю проверку разработанного алгоритма (на уровне идеи);

– разобрать большее число учебных задач.

Составление блок-схемы алгоритма является важным и в большинстве случаев необходимым этапом решения сложной и большой задачи на ЭВМ, значительно облегчающим процесс составления программ.

Базовые структуры программирования

Выделяют три основные структуры алгоритмов:

1. Линейная.

2. Разветвляющаяся (альтернатива «если–то–иначе» или «если–то»).

3. Циклическая (повторение).

Линейная структура – является основной. Она означает, что действия выполняются друг за другом. Прямоугольник, показанный на рисунке, может представлять, как одну единственную команду, так и множество операторов, необходимых для выполнения сложной обработки данных. Команды записываются с помощью операции присваивания.

Присваивание переменной какого-либо значения или присваивание одной переменной значения другой переменной является наиболее часто выполняемым действием в программе, написанной на любом языке программирования. В языке программирования присваивание является операцией и обозначается как «:=» или «=». Это означает, что при выполнении этой операции происходит не только присваивание значения определенной переменной, но и возвращается некоторое значение.

Разветвляющаяся структура (ветвление) – это структура, обеспечивающая альтернативный выбор в зависимости от заданного условия. Выполняется проверка условия, а затем выбирается один из путей, где *P* – это условие, в зависимости от истинности (Да) или ложности (Нет) которого управление передается по одной из двух ветвей.

Может оказаться, что для одного из результатов проверки условия ничего предпринимать не надо. В этом случае можно применять только один обрабатывающий блок.

Эта структура называется также ЕСЛИ – ТО – ИНАЧЕ. Каждый из путей (ТО или ИНАЧЕ) ведет к общей точке слияния, так что выполнение программы продолжается независимо от того, какой путь был выбран.

Циклическая структура (или повторение) предусматривает повторное выполнение некоторого набора действий. Последовательность действий, которые повторяются в цикле, называют **телом цикла**.

Циклические алгоритмы подразделяют на алгоритмы с предусловием, постусловием и алгоритмы с конечным числом повторов (со счетчиком). В алгоритмах с предусловием сначала выполняется проверка условия окончания цикла и затем, в зависимости от результата проверки, выполняется (или не выполняется) так называемое тело цикла.

Пример 1. Определить площадь трапеции по введенным значениям оснований (*a* и *b*) и высоты (*h*).

Запись решения задачи на алгоритмическом языке:

Алгоритм трапеция
Вещественные a, b, h, s
Начало
    Ввод a, b, h
    *s= ((a + b)/2)*h*
Вывод s
Конец

Запись алгоритма в виде блок-схемы (рис. 2):

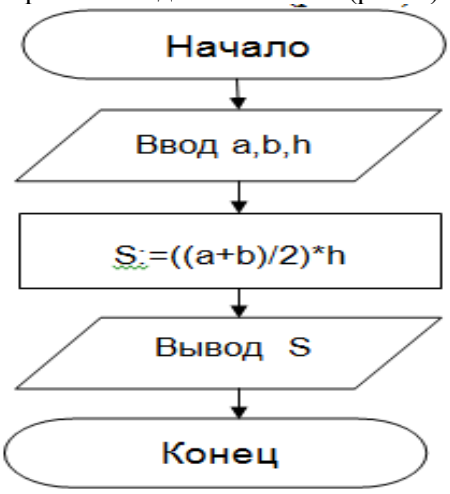


Рис. 2. Блок-схема линейного алгоритма

Пример 2. Определить среднее арифметическое двух чисел, если *a* положительное, иначе частное (*a/b*).

Запись решения задачи на алгоритмическом языке:

Алгоритм числа
Вещественные a, b, c
Начало
Ввод a, b
Если a> 0
тогда c = (a + b) /2
иначе c=a/b
Конец если
Вывод c
Конец

Запись алгоритма в виде блок-схемы (рис. 3):

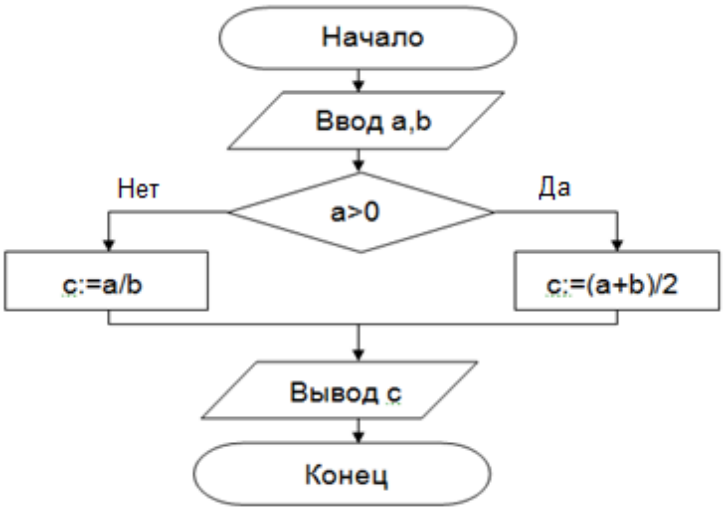


Рис. 3. Блок-схема алгоритма с ветвлением

В алгоритме с предусловием сначала проверяется условие окончания цикла, а затем выполняется тело цикла. Решение задачи нахождения суммы первых десяти целых чисел в данном случае будет выглядеть следующим образом:

Пример 3. Составить алгоритм нахождения суммы целых чисел в диапазоне от 1 до 10.

Запись решения задачи на алгоритмическом языке:

Алгоритм сумма
  Вещественные a, s
Начало
  S=0
  a=1
  Начало цикла
    Пока a <=10
    S=S + a
    a=a+1
  Конец цикла
  Вывод S
Конец

Запись алгоритма в виде блок-схемы (рис. 4):

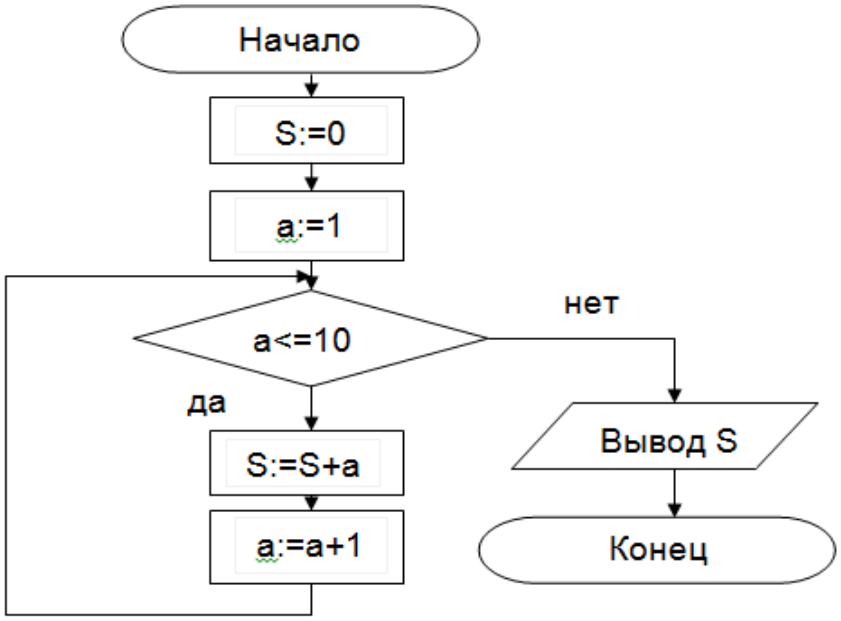


Рис. 4. Циклический алгоритм с предусловием

В алгоритме с постусловием сначала выполняется тело цикла, а затем проверяется условие окончания цикла. Решение задачи нахождения суммы первых десяти целых чисел в данном случае будет выглядеть следующим образом:

Алгоритм сумма
Вещественные a, s
Начало
S=0
A=1
Начало цикла
S=S + a
A=a+1
Пока a <=10
Конец цикла
Вывод S
Конец

Запись алгоритма в виде блок-схемы (рис. 5):

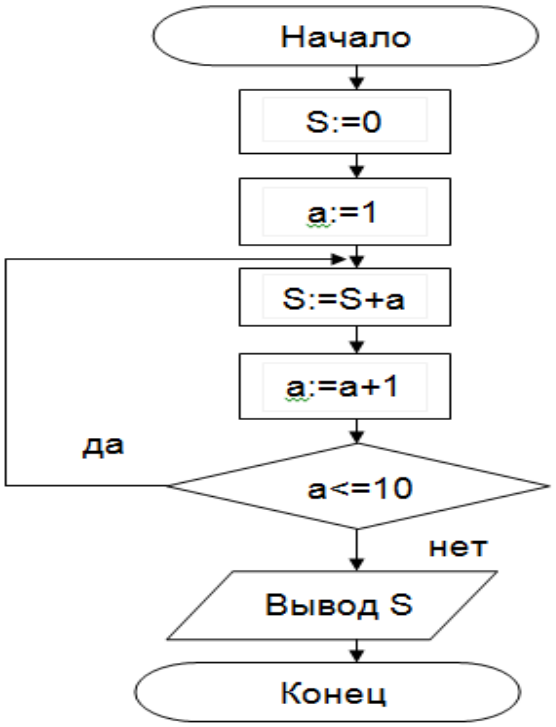


Рис. 5. Циклический алгоритм с постусловием

Решение задачи нахождения суммы первых десяти целых чисел с помощью алгоритма с конечным числом повторов (счетчиком) будет выглядеть следующим образом:

Алгоритм сумма
  Вещественные a, s
Начало
  S=0
  Начало цикла
  Для a=1 до 10
    S=S + a
  Конец цикла
  Вывод S
Конец

Запись алгоритма в виде блок-схемы (рис. 6):

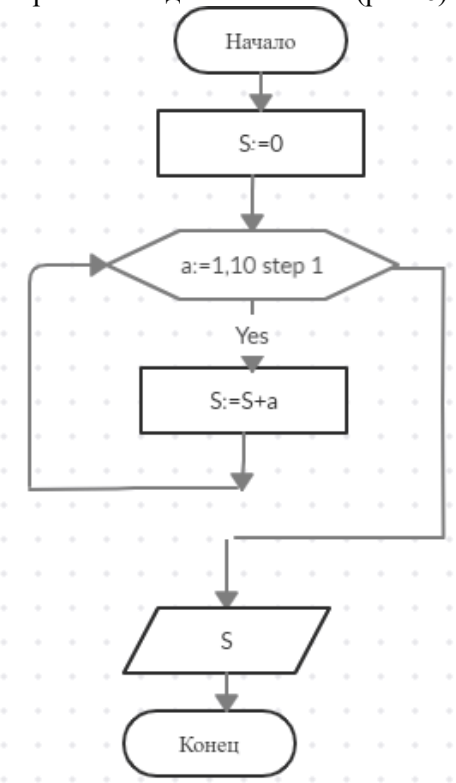


Рис. 6. Циклический алгоритм с помощью алгоритма с конечным числом повторов

Технология выполнения работы

В рамках выполнения работы необходимо составить алгоритм решения задачи в виде блок-схемы **и** записать решения задачи на алгоритмическом языке.

Задание 1. Составить алгоритм решения **трех** задач на алгоритмическом языке и с помощью блок-схем, используя конструкцию линейного алгоритма.

Варианты заданий:

1. Вычислить площадь поверхности и объем усеченного конуса по следующим формулам:

$$S = \pi\,(R + r)\,l + \pi R^2 + \pi r^2\,;$$
$$V = (1/3)\,\pi\,(R^2 + r^2 + Rr)\,h\,.$$

Примечание: π =4*Atn(1)

2. Вычислить координаты центра тяжести трех материальных точек с массами m_1, m_2, m_3 и координатами:

$$(x_1,y_1),\ (x_2,y_2),\ (x_3,y_3)$$

по формулам:

$$x_c = (m_1x_1 + m_2x_2 + m_3x_3) / (m_1 + m_2 + m_3);$$
$$y_c = (m_1y_1 + m_2y_2 + m_3y_3) / (m_1 + m_2 + m_3).$$

3. Вычислить площадь треугольника со сторонами a, b, c по формуле Герона:

$$S = \sqrt{p\,(p-a)(p-b)(p-c)}\,,$$

где p – полупериметр, вычисляемый по формуле $(a + b + c)/2$.

4. Вычислить координаты точки, делящей отрезок a_1a_2 в отношении n_1/n_2 по формулам:

$$x = (x_1 + \gamma x_2)/(1+\gamma)\,;$$
$$y = (y_1 + \gamma y_2)/(1+\gamma)\,,$$

где $y = n_1/n_2$.

5. Вычислить медианы треугольника со сторонами a, b, c по формулам:

$$m_a = 0.5\sqrt{2b^2 + 2c^2 - a^2}\,;$$
$$m_b = 0.5\sqrt{2a^2 + 2c^2 - b^2}\,;$$
$$m_c = 0.5\sqrt{2b^2 + 2b^2 - c^2}\,;$$

6. Вычислить площадь круга и длину окружности по введенному значению радиуса.

7. Вычислить площадь S и периметр L эллипса по введенным значениям полуосей a и b:

$$S := \pi \cdot a \cdot b\,;$$
$$L = 2 \cdot \pi \cdot \sqrt{\tfrac{1}{2}(a^2 + b^2}\,.$$

8. Вычислить объем V и площадь боковой поверхности цилиндра S по введенным значениям радиуса основания R и высоты цилиндра H.

$$V = \pi \cdot R^2 \cdot H ;$$
$$S = 2 \cdot \pi \cdot R \cdot H .$$

9. Вычислить объем V и площадь боковой поверхности конуса S по введенным значениям радиуса основания r, высоты h и образующей l:

$$V = \frac{1}{3}\pi \cdot r^2 \cdot h ;$$
$$S = \pi \cdot r \cdot l .$$

10. Вычислить объем V и площадь поверхности S сферы по введенному значению радиуса r:

$$V = \frac{4}{3} \cdot \pi \cdot r^3 ;$$
$$S = 4 \cdot \pi \cdot r^2 .$$

11. Дано целое четырехзначное число. Используя операции div и mod, найти сумму его цифр.

12. Дана сторона равностороннего треугольника. Найти площадь этого треугольника и радиусы вписанной и описанной окружностей.

13. Даны координаты трех вершин треугольника ($x1$, $y1$), ($x2$, $y2$), ($x3$, $y3$). Найти его периметр и площадь.

14. Дана длина окружности. Найти площадь круга, ограниченного этой окружностью.

15. Дана площадь круга. Найти длину окружности, ограничивающей этот круг.

16. Вычислить площадь поверхности и объем усеченного конуса по следующим формулам:

$$S = \pi\ (R + r)\ l + \pi R^2 + \pi r^2 ;$$
$$V = (1/3)\ \pi\ (R^2 + r^2 + Rr)\ h .$$

17. Вычислить координаты центра тяжести трех материальных точек с массами m_1, m_2, m_3 и координатами:

$$(x_1,y_1),\ (x_2,y_2),\ (x_3,y_3)$$

по формулам:

$$x_c = (m_1x_1 + m_2x_2 + m_3x_3) / (m_1 + m_2 + m_3);$$
$$y_c = (m_1y_1 + m_2y_2 + m_3y_3) / (m_1 + m_2 + m_3).$$

18. Вычислить площадь треугольника со сторонами *a*, *b*, *c* по формуле Герона:

$$S = \sqrt{p\,(p-a)(p-b)(p-c)}\ ,$$

где *p* – полупериметр, вычисляемый по формуле (*a* + *b* + *c*)/2.

19. Вычислить координаты точки, делящей отрезок a_1a_2 в отношении n_1/n_2 по формулам:

$$x = (x_1 + \gamma x_2)/(1+\gamma)\ ;$$
$$y = (y_1 + \gamma y_2)/(1+\gamma)\ ,$$

где $y = n_1/n_2$

20. Вычислить медианы треугольника со сторонами a, b, c по формулам:

$$m_a = 0.5\sqrt{2b^2 + 2c^2 - a^2}\ ;$$
$$m_b = 0.5\sqrt{2a^2 + 2c^2 - b^2}\ ;$$
$$m_c = 0.5\sqrt{2b^2 + 2b^2 - c^2}\ ;$$

21. Вычислить площадь круга и длину окружности по введенному значению радиуса.

22. Вычислить площадь *S* и периметр *L* эллипса по введенным значениям полуосей *a* и *b*:

$$S := \pi \cdot a \cdot b\ ;$$
$$L = 2 \cdot \pi \cdot \sqrt{\tfrac{1}{2}(a^2 + b^2}\ .$$

23. Вычислить объем *V* и площадь боковой поверхности цилиндра *S* по введенным значениям радиуса основания *R* и высоты цилиндра *H*.

$$V = \pi \cdot R^2 \cdot H\ ;$$
$$S = 2 \cdot \pi \cdot R \cdot H\ .$$

24. Вычислить объем V и площадь боковой поверхности конуса S по введенным значениям радиуса основания r, высоты h и образующей l:

$$V = \frac{1}{3}\pi \cdot r^2 \cdot h\ ;$$
$$S = \pi \cdot r \cdot l\ .$$

25. Вычислить объем V и площадь поверхности S сферы по введенному значению радиуса r:

$$V = \frac{4}{3} \cdot \pi \cdot r^3\ ;$$
$$S = 4 \cdot \pi \cdot r^2\ .$$

Задание 2. Составить алгоритм решения **трех** задач на алгоритмическом языке и с помощью блок-схем, используя конструкцию алгоритма с ветвлением.

Варианты заданий:

1. Составить программу для решения квадратного уравнения $ax^2 + bx + c = 0$.

2. Определить максимальное четное число из двух введенных.

3. Определить, можно ли из отрезков с длинами x, y и z построить треугольник.

4. Ввести два числа a и b. Большее число заменить утроенным произведением, меньшее – полосуемой.

5. Если среди трех чисел a, b, c имеется хотя бы одно четное, то найти максимальное число, иначе – минимальное.

6. Определить, в каком квадранте находится точка с координатами x и y и вывести номер квадранта на экран.

7. Найти квадрат наибольшего из двух чисел a и b. Вывести на экран число 1, если наибольшим является число a, число 2 – если наибольшим числом является b.

8. Определить, попадает ли точка с координатами x и y в круг радиусом R. Если точка попадает в круг, вывести на экран единицу, в противном случае – ноль.

9. Написать алгоритм решения задачи, которая решает уравнение $ax + b = 0$ относительно x для любых чисел a и b, введенных с клавиатуры. Все числа считаются действительными.

10. Написать алгоритм решения задачи, которая определяет, лежит ли точка $A(x, y)$ внутри некоторого кольца («внутри» понимается в строгом смысле, т.е. случай, когда точка А лежит на границе кольца, недопустим). Центр кольца находится в начале координат. Для кольца заданы внутренний и внешний радиусы *r1*, *r2*. Координаты x и y вводятся с клавиатуры.

11. Даны две переменные целого типа: A и B. Если их значения не равны, то присвоить каждой переменной произведение этих значений, а если равны, то присвоить переменным нулевые значения.

12. Даны две переменные целого типа: A и B. Если их значения не равны, то присвоить каждой переменной минимальное из этих значений, а если равны, то присвоить переменным нулевые значения.

13. Даны целочисленные координаты точки на плоскости. Если точка не лежит на координатных осях, то вывести 0. Если точка совпадает с началом координат, то вывести 1. Если точка не совпадает с началом координат, но лежит на оси *OX* или *OY*, то вывести соответственно 2 или 3.

14. Даны вещественные координаты точки, не лежащей на координатных осях *OX* и *OY*. Вывести номер координатной четверти, в которой находится данная точка.

15. Дано целое число, лежащее в диапазоне от –999 до 999. Вывести строку – словесное описание данного числа вида «отрицательное двузначное число», «нулевое число», «положительное однозначное число» и т.д.

Задание 3. Составить алгоритм решения **трех** задач на алгоритмическом языке и с помощью блок-схем, используя конструкцию циклического алгоритма.

Варианты задания:

1. Найти сумму чисел, кратных трем, в диапазоне от 0 до 50.
2. Найти сумму первых десяти чисел, кратных пяти.
3. Найти произведение четных чисел в диапазоне от 2 до 30.

4. Вводятся положительные числа. Прекратить ввод, когда сумма введенных чисел превысит 100.

5. Требуется найти сумму чисел, кратных 7, в диапазоне от 0 до 100. Вывести на экран сумму чисел и их количество.

6. Определить количество целых чисел, кратных 3 (от 3 и далее), дающих в сумме число, превышающее 200.

7. Вводятся 10 чисел. Вывести на экран суммы положительных и отрицательных чисел и их количество.

8. Вывести на экран значения функции $y=sin(x)$ для $0 \le x \le 180$ с шагом в 10.

9. Подсчитать площади десяти кругов с радиусами от 1 см с шагом 2 см и вывести значения площадей на экран.

10. Вводятся положительные числа. Прекратить ввод чисел, когда их сумма превысит 100. Результат вывести на экран.

11. Вводятся числа. Прекратить ввод чисел, когда сумма положительных чисел превысит 100. Результат вывести на экран.

12. Вывести на экран значения функции $y=sin(x)$ для $0 \le x \le 180$ с шагом в 12.

13. Вывести на экран таблицу перевода километров в мили в диапазоне от 2 до 20 километров с шагом 2 км.

14. Вы положили в банк 1500 рублей. Определить, сколько денег будет на вашем вкладе через 1 год, если каждый месяц вклад увеличивается на 0.76 % от суммы предыдущего месяца.

15. Решив заняться легкой атлетикой, вы пробежали в первый день 2 км. Сколько километров Вы пробежите за 2 недели, если каждый день вы увеличиваете дистанцию на 10% от предыдущего дня?

Отчет, подготовленный в MS Word, должен содержать:

1. Титульный лист.

2. Цель работы.

Для каждой задачи

3. Условие задачи.

4. Алгоритм, написанный на алгоритмическом языке и с помощью блок-схемы. Блок-схему выполнить в специализированной программе в соответствии с ГОСТ 19.701-90.

Контрольные вопросы:

1. Что такое алгоритм?
2. Свойства алгоритма.
3. Способы записи алгоритма.
4. Основные элементы блок-схемы.
5. Виды алгоритмов.
6. Отличительные особенности алгоритмов с предусловием и постусловием.

ЛАБОРАТОРНАЯ РАБОТА №2.
ПРОГРАММИРОВАНИЕ ЗАДАЧ СРЕДСТВАМИ VBA

Цель работы: научиться использовать язык программирования Visual Basic for Applications (VBA) для разработки модулей в Excel на основе линейных алгоритмов.

Теоретические сведения

Электронная таблица Excel содержит встроенную систему программирования на VBA. Для того, чтобы активизировать VBA можно воспользоваться одним из следующих путей. Самый простой путь – находясь в редакторе Microsoft Excel нажать комбинацию клавиш Alt+F11. Второй вариант – щелкнув правой кнопкой мыши на ярлыке любого рабочего листа из всплывающего меню можно выбрать пункт «Исходный текст». В любом случае открывается окно, которое имеет вид, показанный на рис.7.

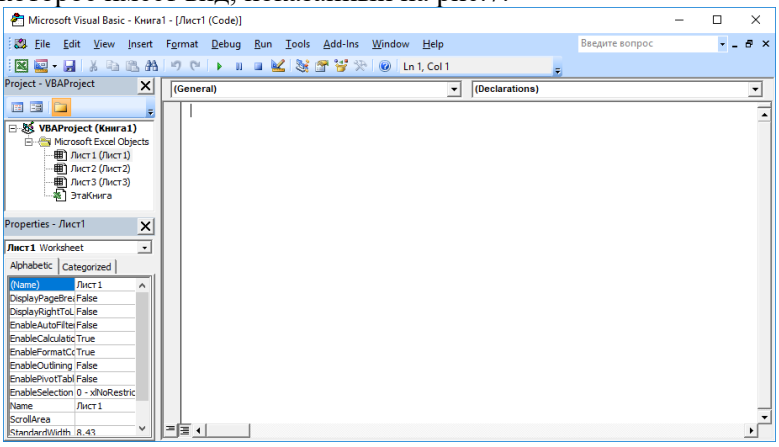


Рис. 7. Окно проекта

Это окно, в свою очередь, состоит из нескольких окон.

Окно проекта, расположенное в левом верхнем углу экрана, представляет иерархическую структуру рабочих листов, модулей и форм данной программы (проекта).

Окно свойств, расположенное ниже окна проекта, содержит перечисление свойств текущего объекта. Окно свойств состоит из двух частей – списка объектов и рабочей части – собственно свойств выбранного объекта.

В рабочей части перечислены имена и значения всех свойств объекта. Две закладки в верхней части позволяют переупорядочивать имена свойств по алфавиту или по категориям. Пользователь, создавая объект, может установить для него необходимые значения свойств.

Окно редактора исходного текста программы занимает большую часть в правой части экрана. Это окно тесно связано с окном проекта. При выборе какого-либо объекта в окне проекта, в окне редактора исходного текста автоматически будет появляться программный код, связанный с этим объектом (так называемый ***модуль объекта***). Написание программ в окне редактора существенно упрощается за счет его интеллектуальных возможностей. Так, например, редактор сам контролирует синтаксическую правильность вводимых операторов и выдает сообщения об ошибках, дописывает за программиста те участки программного кода, которые однозначно должны фигурировать в данном контексте и так далее. Кроме того, редактор автоматически распознает и выделяет разными цветами синтаксические конструкции VBA. Редактор кода обладает еще одной очень полезной особенностью. Если текстовый курсор расположить на каком-либо служебном слове VBA, имени процедуры, свойства или метода и нажать клавишу «F1», то на экране появится окно со справочной информацией об этом объекте, слове или методе. Обычно в справке приводится и пример правильного использования данного программного кода. Один только недостаток – справка по VBA не переведена на русский язык. Но при даже незначительных знаниях английского – всегда можно оперативно получить помощь.

Основные синтаксические конструкции VBA

К основним конструкциям языка VBA относятся переменные, константы и служебные слова. Из этих основных, базовых для любого алгоритмического языка конструкций, строятся в свою очередь более сложные конструкции – операторы, процедуры, функции и, в конечном итоге, программы.

Переменные. Переменной называется область оперативной памяти, предназначенная для хранения какого-либо значения. Содержимое этой области памяти (значение переменной) может ме-

няться во время выполнения программы. Каждая переменная имеет собственное имя, которое назначается программистом и формируется по следующим правилам:

– первым символом имени обязательно должна быть буква;

– имя может содержать только буквы, цифры и знак подчеркивания («_»);

– длина имени не должна превышать 255 символов;

– в одной и той же подпрограмме (процедуре или функции) не могут быть объявлены две переменные с одним и тем же именем.

Каждая переменная при создании должна получить определенный тип. Тип переменной характеризует длину, способ представления и диапазон изменения тех значений, которые могут храниться в переменной. В языке VBA возможны типы переменных, приведенные в таблице 1.

Таблица 1.

Типы переменных языка VBA

Тип переменной	Способ описания	Размер (Байт)	Диапазон значений в языке VBA
Байт	Byte	1	Целые числа от 0 до 255
Логический тип	Boolean	2	Только значения True (истина) или False (ложь)
Целое число	Integer	2	От -32768 до +32767
Длинное целое	Long	4	От -2 147 483 648 до +2 147 483 647
Число с плавающей запятой одинарной точности	Single	4	От $-1{,}401298*10^{-45}$ до $+3{,}042823*10^{38}$
Число с плавающей запятой двойной точности	Double	8	От $-4{,}940655645584*10^{-324}$ до $+3{,}042823*10^{38}$
Дата и время	Date	8	От 1 января 100г. до 1 января 9999г.
Любой объект	Object	4	Хранит указатель на любой объект, в том числе и на объекты Microsoft Excel
Строка символов	String	Длина строки	От 0 до 65000 символов в строке
Произвольный тип данных	Variant		Может хранить любое значение из описанных выше

Переменные могут быть объединены в так называемые массивы.

Массив – это группа однотипных переменных, которые названы одним общим именем. Массивы можно считать одной из разновидностей обычных (простых) переменных. Отличие состоит в том, что в массивах можно хранить не одно, а несколько значений. Доступ к заданному элементу массива осуществляется с помощью номера этого значения в массиве – так называемого индекса. Синтаксис объявления массива отличается от синтаксиса объявления переменной тем, что здесь требуется указать также размерность массива и границы изменения индексов.

Константы. Для повышения наглядности при написании программ вместо использования какого-либо постоянного значения часто используют *константы*. Применение констант облегчает восприятие текста программы, а так же значительно упрощает отладку программы.

В VBA существуют константы двух типов:

В*строенные константы.* Список этих констант можно просмотреть в окне просмотра объектов. Их количество определяется используемым приложеним. Например, в Microsoft Excel к таким константам относятся *True*, *False*, *Null* и еще множество других, описывающих свойства объектов. Например, xlMaximized, xlMinimized и другие выражения, упомянутые в п. 1.2.1, являются именно константами Excel. Об этом, кстати, говорит и префикс «xl» перед каждой такой константой.

Пользовательские константы объявляются с помощью служебного слова *Const*. В момент объявления пользовательским константам присваиваются удобные для восприятия имена и значения. Например, часто используемое в программе число π можно описать в виде константы Const Pi = 3.141592, а затем идентификатор Pi многократно использовать в программе. Наглядность текста такой программы существенно повышается.

Операторы являются основными конструкциями языка. Именно с помощью операторов реализуется алгоритм работы программы. В современных языках существует огромное количество самых разнообразных операторов. Ниже кратко описаны основные операторы языка VBA.

Арифметические операторы предназначены для выполнения основных арифметических операций над операндами, в качестве которых могут выступать как переменные, так и числа, и константы. Знаками этих операторов являются общепринятые математические знаки +, -, *, /. Например, оператор + выполняет операцию сложения двух чисел или выражений, являющихся операндами. Большинство арифметических операторов VBA требуют наличия двух операндов. Оператор присваивания, задаваемый необязательным служебным словом Let, используется в том случае, когда требуется присвоить значение выражения переменной или свойству.

Ее синтаксис имеет вид: [Let] <Имя Переменной> = <выражение>

Элементы синтаксиса инструкции Let содержат элементы, представленные в таблице 2.

Таблица 2.

Элементы синтаксиса инструкции Let

Элемент	Описание
Let	Необязательный элемент. Ключевое слово Let обычно опускают
<Имя Переменной>	Обязательный элемент, представляет собой имя переменной или свойство, удовлетворяющее правилам именования переменных
<Выражение>	Обязательный элемент, определяющий значение, присваиваемое переменной или свойству

Значение выражения может быть присвоено только в том случае, если типы переменной и выражения совместимы. Например, нельзя присвоить числовой переменной значение выражения, которое является строкой.

В Visual Basic for Applications имеются стандартные окна для ввода данных в программу и отображения результатов, т.е. вывода данных. Окно сообщений (*MsgBox*) и окно ввода (*InputBox*) позволяют организовать простейший диалог с пользователем.

Синтаксис окна ввода: *Переменная = InputBox*("текст", "заголовок окна")

Этот оператор выводит на экран диалоговое окно, которое содержит текст – сообщение и поле для ввода данных.

Синтаксис окна сообщений MsgBox: *MsgBox(сообщение, кнопки+ значки, заголовок окна)*

Параметр *сообщение* содержит текст, который выводится в окне диалога.

Параметр *кнопки* указывает число и тип отображаемых кнопок в окне сообщения.

Меню редактора VBA

После входа в редактор Visual Basic, а также при изменении макроса в меню вы автоматически окажетесь в редакторе. Состав и назначение основных пунктов меню редактора представлены ниже.

File – команды сохранения изменений в проекте и вывода на экран и печать исходного кода макросов.

Edit – команды управления исходным кодом в окне Code, а также объектами в формах.

View – команды, позволяющие выводить или убирать с экрана различные окна самого редактора VBA.

Insert – команды вставки в проект различных объектов: процедур, модулей, форм, классов и пр.

Format – команды, используемые при создании пользовательских диалоговых окон. Они позволяют выравнивать объекты в форме по отношению друг к другу, настраивать размеры и внешний вид элементов управления и другие операции.

Debug – команды тестирования и отладки кода. Позволяют запускать код с любой точки, отслеживать ход выполнения по шагам, видеть значения, прерывать программу в нужном месте.

Run – команды запуска программного кода на выполнение, прерывания, возобновление работы, а также возврата прерванной программы в исходное состояние.

Tools – команды, позволяющие выбрать макрос для выполнения или получения доступа к внешним библиотекам макросов. Доступ к диалоговому окну Option (параметры) редактора и окну свойств VBA.

Add-Ins – одна команда Add-Ins Manager для вывода диалогового окна, в котором можно загружать, выгружать, регистрировать или определять поведение программ – дополнений (надстроек).

Основными парадигмами объектно-ориентированного программирования на языке VBA являются объект, свойство, метод, событие, класс и семейство объектов.

Объект – это инкапсуляция данных вместе с кодом, предназначенным для их обработки.

Семейство – объект, содержащий несколько других объектов того же типа:

Worksheets ("Лист 1") – рабочий лист с именем Лист1;

Worksheets (1) – первый лист рабочей книги.

Класс – это проект, на основе которого будет создан объект, т.е. класс определяет имя объекта, его свойства и действия, над ним выполняемые. А каждый объект, свою очередь, является экземпляром класса.

Методы – это действия, выполняемые над объектом.

Объект. Метод – синтаксис метода

Примеры:

Application. Quit – закрыть объект Application.

Worksheets ("Лист1"). Chartobjects.Delete – удалит все диаграммы с листа "Лист1".

Свойства – это атрибут объекта, определяющий его характеристики: размер, цвет, положение на экране или состояние (доступность, видимость).

Для изменения характеристик меняют его свойства:

Объект. Свойство=Значение свойства

Пример: Worksheets.Visible = False

Есть свойства, возвращающие объект:

ActiveCell возвращает активную ячейку активного листа активной рабочей книги.

ActiveWindow – активное окно.

Свойства: ActiveCell, ActiveWindow. ActiveCell и Application. ActiveCell возвращают одну и ту же активную ячейку.

События – это действия, распознаваемые объектом.

Суть программирования на VBA и заключается в том, чтобы на событие получить отклик.

Пример 1. Написать программу, которая вычисляет периметр треугольника.

Визуальное представление формы для ввода данных представлено на рисунке 8, результата – рис.9.

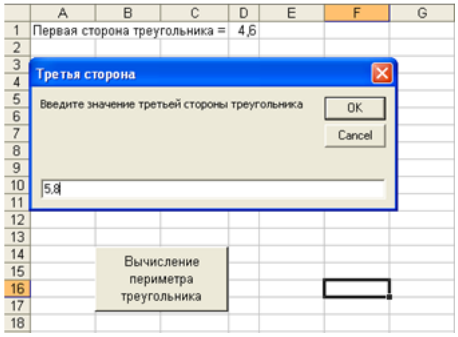


Рис. 8. Ввод исходных данных в процессе выполнения программы

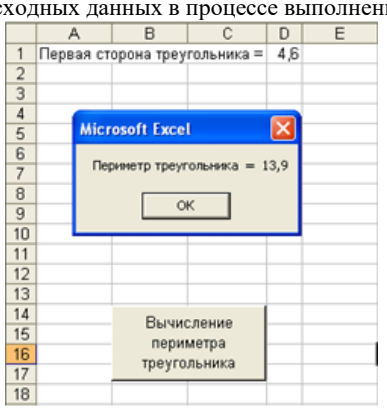


Рис. 9. Вывод результата в процессе выполнения программы

Программный код

```
Option Explicit
Sub Perimeter()
    Dim A, B, P As Single
    A = 3.5
    B = InputBox("Введите значение третьей стороны треугольника", "Третья сторона")
    P = Cells(1, 4) + A + B
    MsgBox "Периметр треугольника =" & P
End Sub
```

Пример 2. Вычислите значение квадратного корня из суммы трех переменных.

Визуальное представление формы для ввода данных представлено на рисунке 10, результата – рис. 11.

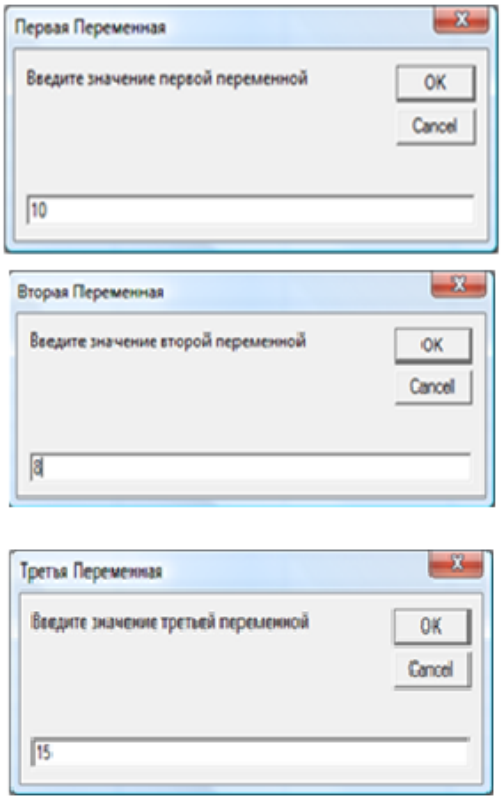


Рис 9. Ввод исходных данных в процессе выполнения программы

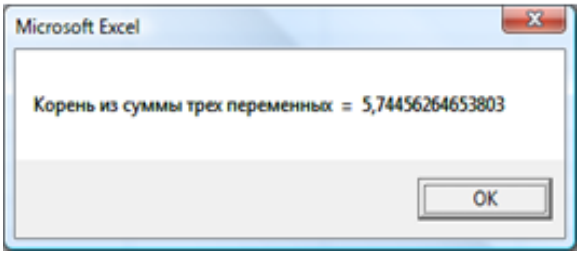


Рис. 10. Вывод результата в процессе выполнения программы

Программный код

```
Option Explicit
Sub root_of_number()
```

```
    Dim A, B, C, S As Single
    Dim R As Double
    A = InputBox("Введите значение первой переменной", "Первая Переменная")
    B = InputBox("Введите значение второй переменной", "Вторая Переменная")
    C = InputBox("Введите значение третьей переменной", "Третья Переменная")
    S = CSng(A) + CSng(B) + CSng(C)
    R=Sqr(S)
    MsgBox "Корень из суммы трех переменных = “& R
End Sub
```

Пример 3. Вычислить *Y*, задав значения переменным:

$$Y = \frac{\sqrt{5 \cdot x^4 + 2 \cdot \sin^2 \beta}}{\cos\left(\left|\alpha^3 - 3 \cdot tg\beta\right|\right)}$$

Программный код

```
Private Sub Main()
    Dim x, b, a, res As Single
    x = Cells(3, 1)
    b = Cells(3, 2)
    a = Cells(3, 3)
    res = (Sqr(5*x^4+2* Sin(b)^2))/Cos(Abs(a^3-3*Tan(b)))
    Cells(5, 2) = res
End Sub
```

Визуальное представление решения задачи представлено на рисунке 12.

Рис.12. Визуальное представление решения задачи

Второй вариант решения

Программный код

```
Sub Main1()
    Dim x, b, a, res As Single
    x = -3
    b = 2
    a = 0.5
    res = (Sqr(5*x^4+2* Sin(b)^2))/Cos(Abs(a^3-3*Tan(b)))
    MsgBox "res=" & CStr(res)
End Sub
```

Пример 4. Вычислить *f*, задав значения переменным:

$$f = \frac{x}{\left(4{,}8x^3 - y\right)^3} + \frac{1}{4}\left(t - \frac{3}{x}\right)$$

Программный код

```
Private Sub Main2()
    Dim x, y, t, res As Single
    x = Cells(3, 9)
    y = Cells(3, 10)
    t = Cells(3, 11)
    res = (x/(4.8*x^3-y)^3)+1/4*(t-3/x)
    Cells(5, 10) = res
End Sub
```

Визуальное представление решения задачи представлено на рис. 8.

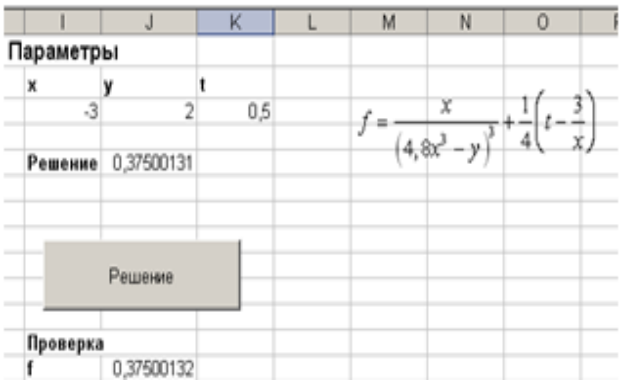


Рис.13. Визуальное представление решения задачи

Второй вариант решения

Программный код

```
Sub Main3()
    Dim x, y, t, res As Single
    x = -3
    y = 2
    t = 0.5
    res = (x/(4.8*x^3-y)^3)+1/4*(t-3/x)
    MsgBox "res=" & CStr(res)
End Sub
```

Технология выполнения работы:

1. Выполнить все примеры (4 примера).
2. Разработать алгоритм для написания программ в виде блок-схемы (3 задачи по вариантам).
3. Написать программы (3 задачи по вариантам).
4. Протестировать программы с разными исходными данными.
5. Написать программы для задания 1 лабораторной работы №1 (3 задачи).
6. Протестировать их с разными исходными данными.

Итого: 4 примера, 3 задачи (+ блок-схема для каждой) и 3 задачи из задания №1 ЛР№1 – 10 задач.

7. Создать отчет.

При выполнении работы необходимо учесть следующие моменты:

1. В программе значение одной из переменных должно быть записано в ячейке электронной таблицы.

2. Значение одной из переменных задается в программе или вводится с помощью окна-диалога.

3. Все типы переменных и констант, которые используются в программе, должны быть объявлены и отражать смысловое значение переменных или констант.

4. Результат работы программы должен иметь наглядный вид.

5. Для запуска макроса создать кнопку с соответствующей надписью.

Варианты заданий:

Задание 1. Составить блок-схему и написать программный код согласно условию задачи.

1. Вычислите значение квадратного корня из суммы трех переменных.

2. Вычислите значение суммы обратных величин трех переменных.

3. Найдите сумму первой и второй переменных и разность первой и третьей переменной.

4. Найти объем цилиндра.

5. Вычислить площадь прямоугольника.

6. Вычислите значение куба, квадрата и обратной величины переменных.

7. Найдите площадь круга по формуле $S=\pi R^2$.

8. Найти расстояние от точки *M* до начала координат.

9. Найти сторону квадрата, площадь которого равна площади прямоугольника.

10. Найти площадь ромба.

11. Определите произведение трех переменных.

12. Найти гипотенузу треугольника.

13. Найти остаток от деления двух действительных чисел.

14. Найти сумму квадратов трех чисел.

15. Найти среднее арифметическое трех введенных чисел.

Задание 2. Вычислить для своего варианта *Y*, задав значения переменным (рис. 14):

1. $Y = \dfrac{\ln x - bx^2 \cdot ctga^2}{2a^3 \cdot \cos^4 \beta}$
2. $Y = \dfrac{2e^x - tg^3 \alpha}{\alpha^3 - b \cdot \cos\beta}$
3. $Y = \dfrac{3 \cdot \sin^2 \alpha - 2\cos^2 \beta}{b^{\alpha} \cdot ctg^2 \beta}$
4. $Y = \dfrac{a^5 \cdot \sqrt{\cos^2 \alpha + 2.8}}{2 \cdot tg^2 \beta - \cos^3 \alpha}$
5. $Y = \dfrac{b^5 - a^2 x^2 + 2 \cdot tg^3 \beta}{\arccos x - 2 \cdot \sin \beta}$
6. $Y = \dfrac{\sec^2 \beta - \cos^2 \alpha}{2 \cdot e^{|\alpha+\beta|} + 1.3 ctg\beta}$
7. $Y = \dfrac{a^3 - b^2 \cdot \arcsin^3 \beta}{\sqrt{a^2 + \sin^4 |\beta + x|}}$
8. $Y = \dfrac{b^3 - 7 \cdot tg|e + \beta|}{2 \cdot \ln x^4 - 5 \cdot tg\alpha^3}$
9. $Y = \dfrac{2 \cdot |\lg x^2 - \ln x^4|}{7.5 \cdot \cos\beta - 2.3\alpha}$
10. $Y = \dfrac{5\cos\alpha + arctg^3 \beta}{e^{\alpha} \cdot |a^2 + b^2|}$
11. $Y = \dfrac{5x^2 - 2 \cdot \ln x^6 \cdot \sin\alpha}{2\cos^3 \beta + 1.7}$
12. $Y = \dfrac{2 \cdot \cos^3 \beta - e^{\alpha}}{ctg^2 |\alpha - \beta|}$
13. $Y = \dfrac{\alpha \cdot tg^3 \beta + 2 \cdot x}{\sqrt{\alpha^2 \cdot \cos^4 \beta + \alpha^4}}$
14. $Y = \dfrac{e^x \cdot |\ln x^2 + 3.8|}{ctg^3 x - \cos\beta}$
15. $Y = \dfrac{3a^2 \cdot |\cos\alpha - tg^3 \beta|}{\sqrt{2 \cdot \sin^2 \alpha + \ln x^2}}$

Рис.14. Лабораторная работа 2. Задание 2

Задание 3. Вычислить для своего варианта *f*, задав значения переменным (рис. 15):

1. $f = \dfrac{1}{1+\dfrac{1}{y+\dfrac{t^3}{x+\dfrac{1}{t-x}}}}$

2. $f = \dfrac{x-y}{3\cdot x^2 - \left(4-\dfrac{y}{t}\right)}$

3. $f = \dfrac{9x}{y^2+5} - \dfrac{(3{,}2x-t)^2}{x-y\cdot t}$

4. $f = \dfrac{x^4}{4} - \dfrac{2{,}8(x+t)}{(x-t)\left(y-\dfrac{3}{t}\right)}$

5. $f = \left(\dfrac{4{,}8+x}{10-t^2}\right)^3 + \dfrac{1}{2}\cdot y^3$

6. $f = \dfrac{y^5}{8{,}3-x} - \dfrac{9-t}{\left(\dfrac{x}{y}+9\right)^4}$

7. $f = \dfrac{\dfrac{1}{4t^3-x} - \dfrac{1}{2{,}4\cdot y}}{t-y\cdot x}$

8. $f = \left(\dfrac{3{,}8+y}{2+t}\right)^3 + \dfrac{1}{2}\cdot x^2$

9. $f = \dfrac{x}{2+x^2} - \dfrac{2}{3}\cdot(t-y)^3$

10. $f = \left(t-\dfrac{y}{3}\right)^4 + \dfrac{9x}{x^3+0{,}5}$

11. $f = \dfrac{t^4}{4{,}9-x} + \dfrac{1}{(yt-1)^4}$

12. $f = \dfrac{t}{x^3(y^4-x)} + \dfrac{1}{2}xy^2$

13. $f = -\dfrac{29x^2}{(1-y)^2} + \left(\dfrac{t}{x+5}\right)^3$

14. $f = \dfrac{8t}{x^4} - \dfrac{(8{,}8x-t)^2}{18+y}$

15. $f = \dfrac{(x-9t)^4}{y-15{,}6} + \dfrac{x}{4-t^3}$

Рис.15. Лабораторная работа 2. Задание 3

Отчет, подготовленный в MS Word, должен содержать:

1. Титульный лист.
2. Цель работы.

Для каждой задачи

2. Условия задачи
3. Алгоритм в виде блок-схемы. Блок-схему выполнить в специализированной программе в соответствии с ГОСТ 19.701-90.
4. Исходный текст программы.
5. Вводимые данные и результаты – фрагменты экрана с диалоговыми окнами Excel.

Контрольные вопросы:

1. Какие встроенные типы данных вы знаете?
2. Как описываются переменные?
3. Как объявляют константы?
4. Какие операции языка VBA вы знаете?
5. Какие операторы языка VBA вы знаете?
6. Перечислите операторы управления
7. Укажите операторы, используемые для ввода данных с рабочего листа.
8. Как заменить ArcSin в VBA?
9. Укажите формулу, которая заменит ArcCtg в VBA.

ЛАБОРАТОРНАЯ РАБОТА №3. СОЗДАНИЕ, ТЕСТИРОВАНИЕ, ОТЛАДКА, ОЦЕНКА И АНАЛИЗ ПОЛУЧЕННОГО РЕЗУЛЬТАТА ПРОГРАММЫ С ПРИМЕНЕНИЕМ ОПЕРАТОРА ВЕТВЛЕНИЯ

Цель работы: научиться использовать язык программирования Visual Basic for Applications (VBA) для разработки модулей в Excel на основе условных алгоритмов.

Теоретические сведения.

VBA – это язык объектно-ориентированного программирования.

Основными парадигмами являются объект, свойство, метод, событие, класс и семейство объектов.

Объект – это инкапсуляция данных вместе с кодом, предназначенным для их обработки.

Семейство – объект, содержащий несколько других объектов того же типа.

Классы – это проект, на основе которого будет создан объект, т.е. класс определяет имя объекта, его свойства и действия, над ним выполняемые. А каждый объект, свою очередь, является экземпляром класса.

Методы – это действия, выполняемые над объектом.

Свойства – это атрибут объекта, определяющий его характеристики: размер, цвет, положение на экране или состояние (доступность, видимость).

События – это действия, распознаваемые объектом.

Структура редактора VBA

Интерфейс VBA состоит из следующих основных компонентов: окно проекта, окно свойств, окно редактирования кода, окна форм, меню и панели инструментов.

В окне проекта (VBAProject) представлена иерархическая структура файлов форм и модулей текущего проекта (рис. 16).

В проекте автоматически создается модуль для каждого рабочего листа и для всей книги. Кроме того, модули создаются для каждой пользовательской формы, макросов и классов. По своему предназначению модули делятся на два типа: ***модули объектов*** **и** ***стандартные.***

К **стандартным модулям** относятся те, которые содержат макросы. Такие модули добавляются в проект командой **Вставка, Модуль** (Insert, Module).

К **модулям объектов** относятся модули, связанные с рабочей книгой, рабочими листами, формами, и модули класса.

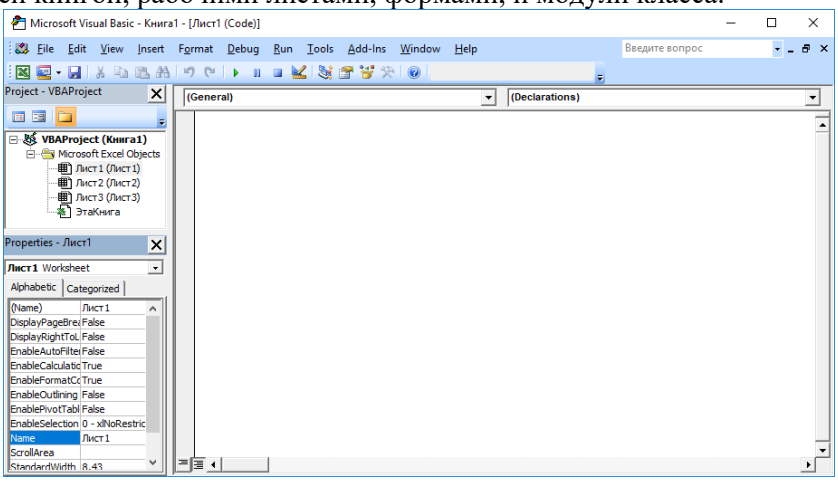


Рис. 16. Окно проекта

Интеллектуальные возможности редактора кода

Например, набирая код Range("A1"). После ввода точки на экране отобразится список компонентов, которые логически завершают данную инструкцию. Двойной щелчок на выбранном элементе из этого списка или нажатие клавиши <Tab> вставляет выбранное имя в код программы.

Автоматическое отображение на экране сведений о процедурах, функциях, свойствах и методах после ввода их имени можно выводить на экран нажатием комбинации клавиш <Ctrl>+<!>.

Если после набора строки и нажатия клавиши <Enter> строка выделяется **красным цветом**, то это как раз и указывает на наличие **синтаксической ошибки** в набранной строке. Эту ошибку необходимо найти и исправить.

Если курсор расположить на ключевом слове языка VBA, имени процедуры, функции, свойства или метода и нажать **клавишу <F1>,** то на экране появится **окно со справочной информацией** об этой функции.

Окно редактирования форм (UserForm)

Для создания диалоговых окон, разрабатываемых приложений в VBA, используются формы.

Форма в проект добавляется с помощью команды **Вставка, Форма** (Insert, Form) или нажатием кнопки **Вставить UserForm** (Insert UserForm). В результате на экран выводится незаполненная форма с панелью инструментов **Панель элементов** (Toolbox) (рис. 17).

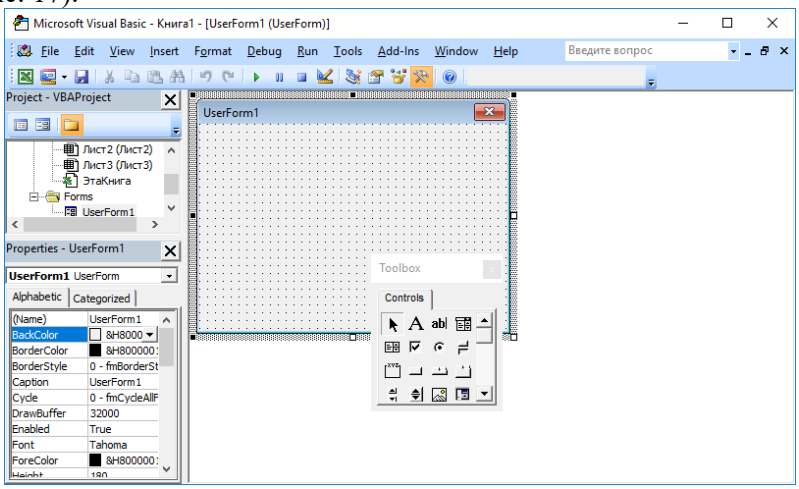


Рис. 17. Окно редактирования форм и панель инструментов Панель элементов

Используя панель инструментов **Панель элементов** из незаполненной формы, можно сконструировать любое требуемое для приложения диалоговое окно. Размещение нового управляющего элемента в форме осуществляется следующей последовательностью действий:

1. Щелкните значок того элемента, который вы собираетесь разместить в форме.

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

3. Нажмите левую кнопку мыши и, не отпуская ее, растяните появившийся прямоугольник до требуемых размеров.

4. Отпустите кнопку мыши. Элемент управления на нужном место создан.

Размеры формы и расположенных на ней элементов управления можно изменять. Технология изменения размеров стандартная для Windows: выделить изменяемый элемент, разместить указатель мыши на одном из размерных маркеров и протащить его при нажатой левой кнопки мыши так, чтобы объект принял требуемые размеры. Окно редактирования форм поддерживает операции буфера обмена.

Таким образом, можно копировать, вырезать и вставлять элементы управления, расположенные на поверхности формы. Для облегчения размещения и выравнивания элементов управления используется **сетка.** Активизировать ее можно с помощью вкладки **Общие** (General) диалогового окна **Параметры** (Options), вызываемого командой **Сервис, Параметры** (Tools, Options), там же устанавливается шаг сетки. Кроме того, команды меню **Формат** (Format) автоматизируют и облегчают процесс выравнивания элементов управления как по их взаимному местоположению, так и по размерам (рис. 18).

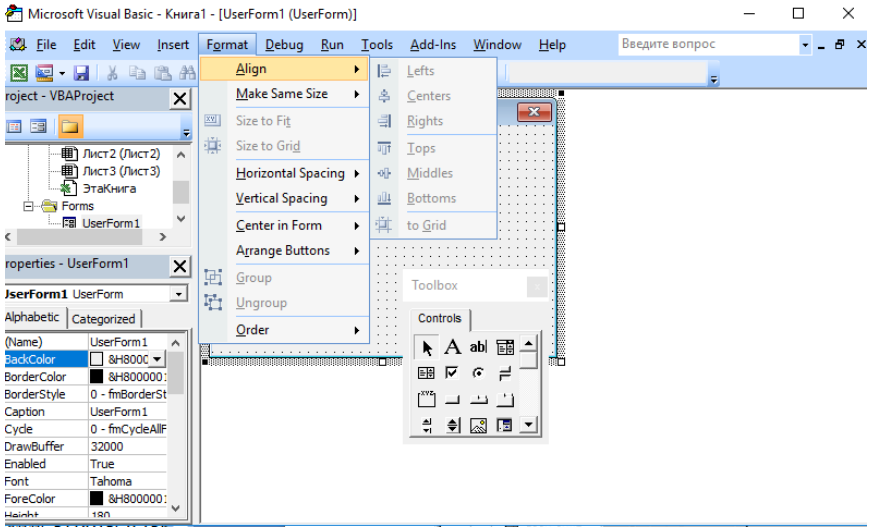


Рис. 18. Команды меню Формат

Окно свойств

В окне свойств перечисляются основные установки свойств выбранной формы или элемента управления.

Используя это окно, можно просматривать свойства и изменять их установки. Для просмотра свойств выбранного объекта надо либо щелкнуть кнопку **Окно свойств** (Properties Window) либо выбрать команду **Вид, Окно свойств** (View, Properties Window) (рис. 19).

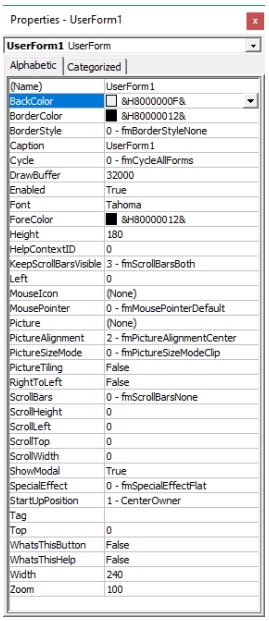


Рис. 19. Окно свойств

Окно свойств состоит из двух составных частей: верхней и рабочей.

В верхней части окна свойств располагается раскрывающийся список, из которого можно выбрать любой элемент управления текущей формы или саму форму.

Рабочая часть состоит из двух вкладок: **По алфавиту** (Alphabetic) и **по категориям** (Categorized), отображающие набор свойств в алфавитном порядке или по категориям. В обоих вкладках свойство Name (имя элемента управления) будет первым. Изменяются значения свойств одним из следующих способов: вводом с клавиатуры значения свойства в соответствующее поле; значения большинства свойств можно выбрать из раскрывающегося списка.

Раскрывающийся список активизируется щелчком в соответствующем поле окна свойств.

Окно Просмотр объектов (Object Browser)

Окно **Просмотр объектов** (Object Browser) вызывается командой **Вид, Просмотр объектов** (View, Object Browser) или нажатием кнопки **Просмотр объектов** (Object Browser) (рис. 20). В этом окне приведен список всех объектов, которые имеются в системе и которые можно использовать при создании проекта.

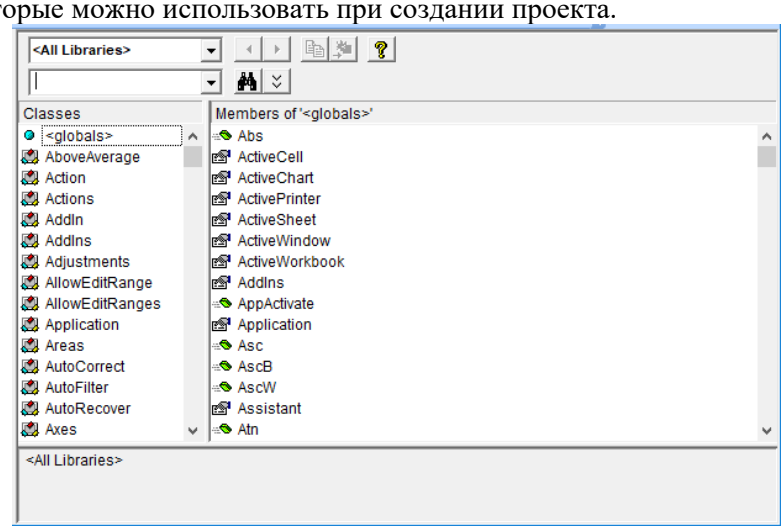


Рис. 20. Окно Просмотр объектов

Окно **Просмотр объектов (Object** Browser) состоит из трех основных частей:

1. Раскрывающегося списка **Проект/Библиотека** (Project/Library) в левом верхнем углу окна. В этом раскрывающемся списке можно выбрать различные проекты и библиотеки объектов. В частности, библиотеки объектов Excel, VBA, Office и VBAProject (объекты пользовательского проекта). Выбор в списке строки **<Все библиотеки>** (<All Libraries>) отображает список объектов всех библиотек.

2. Списка **Классы** (Classes). После выбора из раскрывающегося списка **Проект/Библиотека** (Project/Library) просматриваемой библиотеки, например **VBA,** все классы объектов выбранной библиотеки выводятся в списке **Классы** (Classes).

3. Списка **Компоненты** (Members). После выбора класса из списка **Классы** (Classes) просматриваемой библиотеки, например FileSystem, все компоненты выбранного класса выводятся в списке **Компоненты** (Members). При выделении строки в этом списке в нижней части окна **Просмотр объектов** (Object Browser) приводится дополнительная информация о выбранном компоненте. Кроме того, если нажать на кнопку **Справка** (Help), расположенную на панели инструментов в правой верхней части окна **Просмотр объектов** (Object Browser), то на экране отобразится окно **Справочник Visual_Basic** (Microsoft Visual Basic Help) с подробной информацией о выделенном компоненте.

Оператор условного перехода If.

1-я форма оператора If:

If условие_then
операторы
End If

Логика работы: если значение *условия истина*, то выполняются операторы, иначе текст программы за end If.

2-я форма оператора If:

If условие then
операторы 1
Else
операторы 2
End If

Логика работы: если значение *условия* истина, то выполняются операторы 1, иначе выполняются операторы 2.

В условии могут применяться следующие операции сравнения: < – меньше, > – больше, <= – меньше-равно, >= – больше-равно, = – равно, <> – неравно, и логические операции: not – логическое **не**, and – логическое **и,** or – логическое **или**.

Пример 1. Написать код программы, вычисляющей значение *Z.*

$$z = \begin{cases} (x-y)^x, \text{если } x > 0.1 \text{ и } y > 0.1 \\ \sin(x), \text{если } x < 0.1 \text{ и } y < 0.1 \\ x^2 / y, \text{если } x = 0.1 \text{ и } y = 0.1 \end{cases}$$

Программный код

```
Sub Условия()
    Dim x, y, Z As Double
    x = Range("A2").Value
    y = Range("B2").Value
    If x > 0.1 And y > 0.1 Then
    Z = (x - y) ^ x
       ElseIf x < 0.1 And y < 0.1 Then
       Z = Sin(x)
       ElseIf x = 0.1 And y = 0 Then
          If y = 0 Then
          MsgBox "Ошибка! Деление на ноль!"
          Else: Z = (x * x) / y
          End If
    End If
    Cells(2, 4).Value = Z
End Sub
```

Рассмотрим результат работы программы при разных значениях x и y (рис. 21, 22, 23).

	A	B	C	D
1	x	y		Z
2	-0,5	0		-0,47943
3				

Рис. 21. Результат вычисления функции $y= sin(x)$ при $x<0.1$ и $y<0.1$

	A	B	C	D
1	x	y		Z
2	3	1		8
3				

Рис. 22. Результат вычисления функции $y= (x - y)^x$ при $x>0.1$ и $y>0.1$

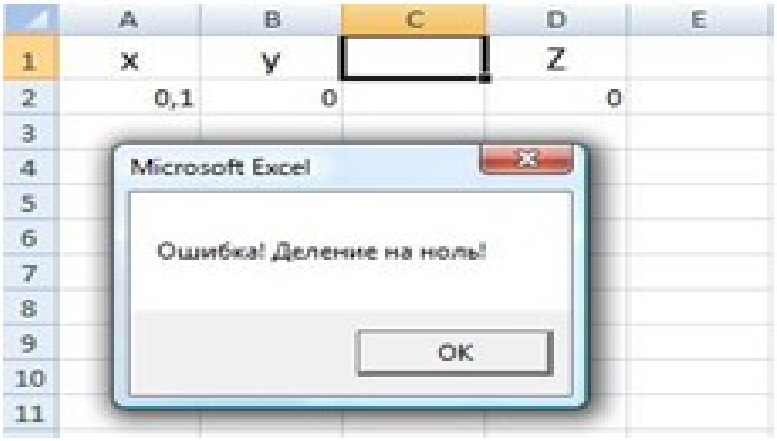


Рис. 23. Результат вычисления функции $y=x^2/y$ при $x=0.1$ и $y=0$

Пример 2. Написать программу вычисления функции.

$$z = \begin{cases} x + y, \text{если } x > 0 \text{ и } y > 1 \\ x / y, \text{если } x \le 1 \text{ и } y <> 0 \end{cases}$$

Предусмотреть сбойную ситуацию (при $y = 0$) и в этом случае выдать сообщение об ошибке.

Закончить запись программного кода, дать визуальное представление ввода данных и вывода результата.

Фрагмент программного кода:

```
…
If y = 0 Then
MsgBox "Ошибка! На ноль делить нельзя!"
ElseIf x > 0 And y > 1 Then
z = x + y
ElseIf x <= 1 And y <> 0 Then
z = x / y
End If
…
```

Пример 3. Составить программу для вычисления выражения:

$$Y = \begin{cases} \sin^3 x, \text{ если } -4 \le x \le 0 \text{ или } 2 \le x \le 3 \\ \dfrac{4.1\,tgx}{x^2 + 4}, \text{если } -6 \le x \le -4 \\ 1, \text{в противном случае} \end{cases}$$

Создать форму, в которой ввод исходного значения ***X*** и вывод результата вычислений ***Y*** осуществляется в поля ввода (TextBox – рис. 24). Пример формы приведен на рисунке 25.

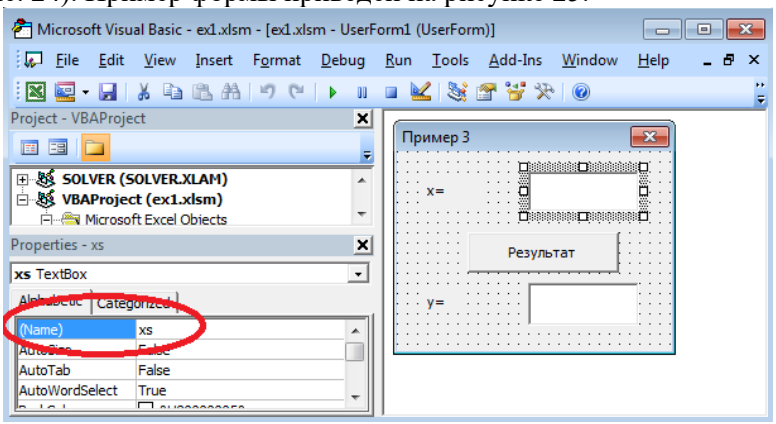


Рис. 24. Свойство *Name* для TextBox ввода данных (*x*)

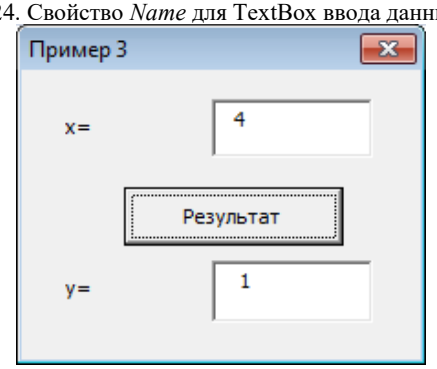


Рис. 25. Пример оформления формы для выполнения примера 3

Фрагмент текста программы:

```
...
X = Val (xs.Text)
If (-4 <= X) and (X <= 0) Or (2 <= X) And (X <= 3) Then
        Y = Sin(X) ^ 3
Else
        If (-6 <= X) and (X <= -4) Then
                Y = 4.1 * Tan(X) / (X ^ 2 + 4)
        Else
                Y = 1
        End If
End If
ys.Text = Str(Y)
...
```

Пример 4. Проектирование интерфейса в программе, решения квадратного уравнения $a*x*x+b*x+c=0$

Шаг 1. Продумывание программы.

Чтобы решить такое уравнение, нужно найти его дискриминант, а затем корни. Дискриминант ищется по формуле: *D = b*b - 4*a*c*. Если дискриминант $D > 0$, то *X1 = (b + (корень из D)) / 2*a*, *X2 = (b - (корень из D)) / 2*a*. Если дискриминант $D = 0$, то *X1 = X2 = b / 2*a*. Если дискриминант $D < 0$, то корней не существует.

Входные данные в программу – коэффициенты *a, b, c*. Данные будем вводить на форме в текстовые поля для ввода (*TextBox*).

Выходные данные – корни (если они есть) и дискриминант. Данные будем выводить на форму в элемент метку (*Label*). Запускать решение будем нажатием на кнопку (*CommandButton*).

Шаг 2. Проектирование интерфейса.

Запустите Visual Basic. Поместите на форму элементы управления.

Элемент метка (*Label*) нужен для отображения текста, т.е. служит подсказкой. В нём мы будем выводить наш дискриминант и корни.

Элемент текстовое поле (*TextBox*) предназначен для ввода пользователем информации. Введённую информацию можно легко получить в коде программы. Для этого у *TextBox* есть свойство *Text*.

Элемент кнопка (*CommandButton*), нужен для запуска некоторого действия. В нашем случае действием будет решение квадратного уравнения, а запускаться оно будет при нажатии на кнопку.

Разместить элементы управления лучше следующим образом: поместите сначала на форму элемент *Label*. В окне *Properties* измените свойство *Caption* этого элемента на "A:". Затем перетащите на форму элемент *TextBox* так, чтобы он оказался рядом с элементом *Label*. Удалите текст из элемента *TextBox* (свойство Text). Затем выделите оба этих элемента (используя Ctrl) и скопируйте их в буфер обмена клавишами Ctrl+C. Далее нажмите Ctrl+V. Visual Basic спросит вас, нужно ли создать массив из вставляемых элементов управления? Ответьте «Нет». Теперь на форме появилась копия элементов *Label* и *TextBox* (для параметра *B*). Скопируйте таким же образом элементы для параметра *C*. У кнопки измените свойство *Caption* на "Решить!". Изменяя свойство *Caption* кнопки, вы, тем самым изменяете текст на самой кнопке. (Если этого не сделать, то пользователю нашей программы трудно было бы догадаться, что же произойдёт при нажатии на кнопку *CommandButton*!?).

Сделаем текст "Результаты решения" подчёркнутым. Для этого выделите *Label* и найдите в окне *Properties* свойство *Font*. Если выделить это свойство, то справа появится маленькая кнопочка, при нажатии на которую откроется окно выбора шрифта, в окне выбора шрифта поставьте галочку – Подчёркнутый.

Измените свойство *Name* каждого элемента управления согласно таблице 2.

Таблица 2.

Свойство Name каждого элемента управления

Назначение элемента управления	Имя (свойство Name)
A: - параметр A	TxtParamA
B: - параметр B	TxtParamB
C: - параметр C	TxtParamC
Кнопка для запуска решения	CmdCalculate
Label, с вычисленным дискриминантом	LblD
Label, с корнем X1	LblX1
Label, с корнем X2	LblX2
Форма, содержащая все эти элементы	FrmMain

Шаг 3. Написание программного кода.

Алгоритм работы программы:

1. Вводим исходные данные в текстовые поля (*a, b, N*). Напомню, что код для этого писать не нужно. За нас всё сделает Visual Basic и Windows. В этом то и заключается прелесть графического интерфейса пользователя (GUI). Мы только считаем введенные значения и всё.

2. После нажатия на кнопку, производим вычисление дискриминанта и корней.

3. Выводим полученные значения в метки (*Label*).

Нам необходимо написать обработчик события клик (*Click*) нашей кнопки – cmdCalculate.

Для начала объявим переменные:

```
Private Sub cmdCalculate_Click()
        'объявляем переменные
        Dim paramA As Double
        Dim paramB As Double
        Dim paramC As Double
        Dim x1 As Double
        Dim x2 As Double
        Dim D As Double
End Sub
```

Теперь считаем введённые параметры *a*, *b* и *c*. Для этого присвоим переменным *paramA*, *paramB* и *paramC* значения свойства *Text* всех 3-х полей для ввода (*TextBox'ов*). Мы можем это сделать потому, что Visual Basic сам преобразует число в виде строки в обычное число с плавающей точкой.

```
paramA = txtParamA. Text
paramB = txtParamB.Text
paramC = txtParamC.Text
```

Доступ к свойству любого элемента управления осуществляется через точку, которая разделяет имя свойства и имя элемента. Обратите внимание на технологию *Intellisence*. Visual Basic выдает список доступных свойств этого элемента управления. Это очень удобно. Вам не придётся выучивать наизусть длинные и сложные названия свойств. Достаточно выбрать нужное свойство из списка и всё.

Теперь нам нужно рассчитать дискриминант. Для этого присвоим переменной D, которая будет хранить значение дискриминанта, следующее выражение:

*D = (paramB * paramB) - (4 * paramA * paramC)*. Скобки для наглядности. Ставить их в данном случае не обязательно, т.к. умножение всё равно выполниться до вычитания.

Теперь, зная значение дискриминанта, нужно сравнить его с нулём. Если он больше нуля, то вычислить оба корня, если равен нулю, то вычислить один корень, ну, а если меньше, то ничего не вычислять и выдать сообщение о том, что корней нет. Такое ветвление мы организуем с помощью оператора *If*:

```
If D > 0 Then
        x1 = (paramB + Sqr(D)) / (2 * paramA)
        x2 = (paramB - Sqr(D)) / (2 * paramA)
        lblD. Caption = "Дискриминант: " & D
        lblX1.Caption = "Корень №1: " & x1
        lblX2.Caption = "Корень №2: " & x2
ElseIf D = 0 Then
        x1 = paramB / (2 * paramA)
        x2 = x1
        lblD. Caption = "Дискриминант: " & D
        lblX1.Caption = "Корень №1: " & x1
        lblX2.Caption = "Корень №2 = Корню №1"
ElseIf D < 0 Then
        lblD. Caption = "Дискриминант: " & D
        lblX1.Caption = "Корней нет!"
        lblX2.Caption = ""
        MsgBox "Дискриминант меньше нуля! Корней нет!",
vbCritical
End If
```

С отступами код намного нагляднее.

Корень вычисляемой встроенной функцией – *Sqr* (от Square). В выражениях, которые мы присваиваем свойству *Caption* у меток, используем оператор конкатенации (склеивание строк), склеиваем строку слева от & со строкой справа. Перед конкатенацией, числа *Double* сначала преобразуются в строку.

Давайте проверим работоспособность программы. Нажмите кнопку *Start*. Появится наша форма. Введите значение в поля: a = 3, b = -6, c = 2. Нажмите на кнопку "Решить!". Дело в том, что строка для вывода в *Label* не вписывается в его размеры. Поэтому происходит перенос на следующую строчку. Чтобы этого избежать, закройте программу и установите свойство меток *AutoSize* в *True*. (Совет: Чтобы не устанавливать это свойство 3 раза для каждой метки, выдели их сразу все 3 и установите свойство.). Свойство *AutoSize* - подгоняет размер метки так, чтобы текст в свойстве *Caption* полностью уместился в одну строчку и не переносился на другую. Теперь снова запустите программу, введите те же значения и нажмите на кнопку: "Решить!".

Технология выполнения работы:

1. Выполнить все примеры.
2. Разработать алгоритм для написания программ в виде блок-схемы (2 задачи).
3. Написать код программ (2 задачи).
4. Протестировать с разными исходными данными.
5. Написать программы для ЛР1 задание 2 (3 задачи).
6. Протестировать с разными исходными данными.

Итого: 4 примера, 2 задачи (+ блок-схема для каждой) и 3 задачи из задания №2 ЛР№1. Минимум 2 задачи решить, используя форму. Всего 9 задач.

7. Создать отчет.

При выполнении работы необходимо учесть следующие моменты:

1. Все типы переменных и констант, которые используются в программе, должны быть объявлены и отражать смысловое значение переменных или констант.
2. В программе предусмотреть сбойные ситуации (деление на ноль, извлечение квадратного корня из отрицательного числа и т. д.).
3. Результат работы программы должен иметь наглядный вид.
4. Для запуска макроса *создать кнопку* на листе Excel с соответствующей надписью.

Варианты заданий:

Задание 1. Составить блок-схему и написать программу вычисления функции пользователя:

1. $z = \begin{cases} (x-y)^2, \text{если } x > 0.1 \text{ и } y > 0.1 \\ \sin(x), \text{если } x < 0.1 \text{ и } y < 0.1 \\ x^2/y, \text{если } x = 0.1 \text{ и } y = 0.1 \end{cases}$

2. $y = \begin{cases} |-x+1|^2, \text{если } x \le 0.3 \\ \sin^3(x) - cos^2(x-3), \text{если } x \in [0.3;\ 0.9] \\ x^4 / 2e^x, \text{если } x \ge 0.9 \end{cases}$

3. $t = \begin{cases} |v| - 2, \text{если } v \le 0 \\ 1 / \sin(3v), \text{если } v \in [0;\ 1] \\ 3v^3 + e^{2v+1}, \text{если } v \ge 1 \end{cases}$

4. $g = \begin{cases} |x+1| + e^{3x}, \textit{если } x \le -1 \\ -2x / \sin(x+3), \textit{если } x \in [-1;\ 1] \\ x^3 + 2|\cos(3x)|, \textit{если } x \ge 1 \end{cases}$

5. $y = \begin{cases} z * x^2, \textit{если } x < 0 \textit{ и } z < 0 \\ \sqrt{z^2 + x^2}, \textit{если } x > 0 \textit{ и } z > 0 \\ 100 \textit{ в остальных случаях} \end{cases}$

6. $q = \begin{cases} a^2 + c^2 * x, \textit{если } x \ge 0 \textit{ и } a \ge 0 \textit{ и } c \ge 0 \\ \frac{c^2+x^2}{a}, \textit{если } x < 0 \textit{ и } a < 0 \textit{ и } c < 0 \end{cases}$

7. $g = \begin{cases} z^3, \textit{если } z \le 1 \\ 1 / z, \textit{если } z \in [-1; 2] \\ \sqrt{z+1} - e^{5z}, \textit{если } z \ge 2 \end{cases}$

8. $p = \begin{cases} \frac{1}{|a+\sin(a)|}, если\ a < 0 \\ 0, если\ a = 0 \\ -a + \sqrt{3\sin(a)}, если\ a > 0 \end{cases}$

9. $p = \begin{cases} e^b + 7b^3, если\ b \leq 4 \\ 1.8 / \sin^3(b^2), если\ 4 < b < 6 \\ |6 - (b-2)^3|, если\ b \geq 6 \end{cases}$

10. $y = \begin{cases} z + 4|t^3|\sin(t), если\ t < 0\ и\ z < 0 \\ \sqrt{z^2 + 3\cos(t^2)}, если\ t > 0\ и\ z > 0 \\ 1\ в\ остальных\ случаях \end{cases}$

11. $f = \begin{cases} |-x| - 3\sin(x^3), если\ x < 1.3 \\ 1.3x / (\sin(x+1) + \cos^2(x)), если\ x = 1.3 \\ (x - 1.1) / e^{-2x+1}, если\ x > 1.3 \end{cases}$

12. $s = \begin{cases} |cos^2(2b) - 3b^3|, если\ b \leq -2 \\ 8\sqrt{b+1} / \sin(3 + b^2), если -2 < b < 2 \\ (b+2)^4 + 6b, если\ b \geq 2 \end{cases}$

13. $w = \begin{cases} |x - 2| - 5x^3, если\ x < 1.4 \\ 1.3\sin(3x), если\ x = 1.4 \\ cos^2(x) + e^{-2x+1}, если\ x > 1.4 \end{cases}$

14. $z = \begin{cases} x + ye^x, если\ x < 0\ и\ y < 0 \\ \sqrt{\sin(x)}\ cos(y), если\ x \geq 0.1\ и\ y \geq 0.1 \\ y^3 - x\ в\ остальных\ случаях \end{cases}$

15. $s = \begin{cases} z^2 + \cos(2z), если\ z \leq 0.8 \\ -2z|\sin(z)|, если\ z \in [0.8; 2] \\ \sqrt{z+7} / e^{4-z}, если\ z \geq 2 \end{cases}$

Задание 2. Составить блок-схему и программный код согласно условию задачи.

1. Задача определения премии торговому агенту от суммы совершенной им сделки. Если объем сделки до 3000 рублей, то премия 1.5 %; если объем до 10000, то – 3 %; если выше 10000, то – 5 %. Предусмотреть проверку корректности данного «объем сделки».

2. Задача вычисления годовой процентной ставки контракта по кредиту, взятого на определенный срок при известных сумме долга и сумме, подлежащей возврату. Используйте формулу:

$$I = \frac{S-P}{PT},$$

где P – сумма кредита, S – сумма, подлежащая возврату, T – срок кредита.

Предусмотреть проверку корректности данных. Для проверки и отладки программы используйте следующие исходные данные: T = 4 месяца, P = 200 000, S = 210 000.

3. Задача вычисления надбавки к стипендии по результатам сессии. Размер стипендии составляет 2 МРОТ (минимальный размер оплаты труда). Если три экзамена сданы на "5", то надбавка составляет 50%, если есть одна "4" (при остальных "5"), то надбавка составит 25 %, если есть хотя бы одна "2", то стипендия не назначается, т.е. равна "0".

4. Задача расчета суммы вклада с начисленным процентом в зависимости от вида вклада: до востребования – 2 %, праздничный – 5 %, срочный – 3 %. Для проверки и отладки программы используйте данные для всех видов вкладов.

5. Задача расчета реальной доходности с учетом налога на прибыль, которая вычисляется по формуле:

$$Y = \frac{i(1-g)-h}{1+h}.$$

где i – годовая процентная ставка, g - налог на прибыль, выраженный в процентах, h - годовой темп инфляции, вычисляемый по формуле:$h=(1+h_{1/12})^{12}- 1$, где h1/12 – месячный темп инфляции.

Предусмотреть проверку корректности данных. Для проверки и отладки программы используйте следующие исходные данные: i = 60%, $h_{1/12}$ = 3%, g = 25%.

6. Задача расчета надбавки к окладу в зависимости от стажа работы сотрудников предприятия. Если стаж работы меньше 5 лет, то надбавка составляет 0%, если стаж от 5 до 10 лет, то надбавка – 5%, если стаж больше 10 лет, то надбавка – 10%.

7. Задача определения комиссионных менеджеру по продажам от объема проданного товара. Если объем продажи до 8000 рублей, то комиссионные 6%; если объем до 16000 рублей, то – 8%; если объем до 32000 рублей, то – 10%; если выше 32000, то – 12%. Для обозначения ограничений объема продажи используйте константы.

8. Задача расчета премиальных, выплачиваемых рабочему, размер выплаты которых определяется в зависимости от оклада и процента перевыполнения нормы выработки: ниже 100% – премия не назначается, т.е. равна нулю, 100% – премия 20% от оклада, 101–110% – премия 30% от оклада, выше 110% – премия 40% от оклада.

9. Задача, вычисляющая стоимость потребляемой энергии компанией в зависимости от установленной расценки и количества потребляемой энергии: первые 240 кВт/час: 1,62руб. за кВт/час, следующие 300 кВт/час: 2,10руб. за кВт/час, свыше 540 кВт/час: 2,76руб. за кВт/час. Для обозначения тарифов в процедуре использовать константы.

10. Задача начисления премии сотрудникам малого предприятия в зависимости от стажа работы и объема продажи товаров. Если стаж работы меньше 2 лет и объем продажи больше 80000руб., то премия составляет 1.5%. Если стаж от 2 до 5 лет, а объем продажи больше 100 000 руб., то премия – 5%, если стаж больше 5 лет, а объем продажи выше 100000 руб., то премия – 7%, в остальных случаях - премия – 1% .

11. Задача начисления процента удержания у работников завода от начисленной заработной платы и количества иждивенцев по следующему правилу: при количестве иждивенцев более трех – 0 %; при трех иждивенцев – 5 %; при двух иждивенцев – 10 %; при одном иждивенце – 12 %; если нет иждивенцев – 14 %.

12. Задача вычисления надбавки к стипендии по результатам сессии. Размер стипендии составляет 2 МРОТ (минимальный размер оплаты труда). Если три экзамена сданы на "5", то надбавка составляет 50%, если есть одна "4" (при остальных "5"), то надбавка составит 25%, если есть хотя бы одна "2", то стипендия не назначается, т.е. равна "0".

Отчет, подготовленный в MS Word, должен содержать:

1. Титульный лист
2. Цель работы

Для каждой задачи

3. Условие задачи
4. Алгоритм в виде блок-схемы. Блок-схему выполнить в специализированной программе в соответствии с ГОСТ 19.701-90.
5. Исходный текст программы.
6. Вводимые данные и результаты – фрагменты экрана с диалоговыми окнами Excel.

Контрольные вопросы:

1. Какая конструкция у оператора повтора?
2. Какие операторы разветвления вы знаете?
3. Чем заканчивается оператор IF, если в ветке «да» несколько операторов?
4. Как выглядит полная форма оператора IF?
5. Как выглядит оператор IF, если в ветке «нет» отсутствуют операторы?

ЛАБОРАТОРНАЯ РАБОТА №4. СОЗДАНИЕ, ТЕСТИРОВАНИЕ, ОТЛАДКА, ОЦЕНКА И АНАЛИЗ ПОЛУЧЕННОГО РЕЗУЛЬТАТА ПРОГРАММЫ С ПРИМЕНЕНИЕМ ОПЕРАТОРОВ ЦИКЛА И ОПЕРАТОРА ВЫБОРА ВАРИАНТА

Цель работы: научиться использовать язык программирования Visual Basic for Applications (VBA) для разработки модулей в Excel на основе циклических алгоритмов.

Теоретические сведения

VBA относится к языкам объектно-ориентированного программирования (ООП).

Все визуальные объекты, такие как *рабочий лист (WorkSheet), диапазон (Range), диаграмма (Chart), форма (userForm),* являются объектами. В VBA имеется более 100 встроенных объектов.

Семейство (объект collection) представляет собой объект, содержащий несколько других объектов, как правило, одного и того же типа. Например, объект workbooks (рабочие книги) содержит все открытые объекты Workbook (рабочая книга).

Вручную в рабочий лист OLE-объекты вставляются командой **Вставка, Объект** (Insert, Object) с выбором в появившемся диалоговом окне **Вставка объекта** (Object) из списка на вкладке **Создание** (Create New) внедряемого объекта. OLE-объект отличается от обычного тем, что при выборе внедренного объекта (перемещении на него указателя и щелчке кнопкой мыши) активизируется программа, связанная с этим объектом, и меню приложения заменяется меню программы, его создавшей.

Важнейшим понятием ООП является *класс.* Класс обычно описывается, как проект, на основе которого впоследствии будет создан конкретный объект. Таким образом, класс определяет имя объекта, его свойства и действия, выполняемые над объектом. В свою очередь каждый объект, в соответствии с описанным выше, является *экземпляром класса.* Иерархия определяет связь между объектами и показывает пути доступа к ним.

Например, полная ссылка на ячейку A1 рабочего листа лист1 рабочей книги с именем Архив имеет вид:

Application. Workbooks ("Архив")

.Worksheets ("Лист1").Range ("Al")

Объект сам по себе не представляет большого значения. Намного значительнее то, какие действия можно совершать над объектом, и какими свойствами он обладает. *Метод* как раз и представляет собой действие, выполняемое над объектом. Синтаксис применения метода: Объект.Метод

В данном примере при помощи метода Quit (закрыть) закрывается приложение (объект Application). Application.Quit

Свойство представляет собой атрибут объекта, определяющий его характеристики, такие как размер, цвет, положение на экране и состояние объекта, например, доступность или видимость. Чтобы изменить характеристики объекта, надо просто изменить значения его свойств. Синтаксис установки значения свойства:

Объект.Свойство = ЗначениеСвойства

В следующем примере изменяется заголовок окна Excel посредством задания свойства Caption объекту Application:

Application.Caption = "Пример"

Приведем наиболее часто употребляемые подобные свойства.

Событие представляет собой действие, распознаваемое объектом (например, щелчок мышью или нажатие клавиши), для которого можно запрограммировать отклик. События возникают в результате действий пользователя или программы, или же они могут быть вызваны системой (табл. 3).

Таблица 3

События

Событие	Действие
ActiveWindow	Возвращает активное окно Excel
ActiveWorkbook	Возвращает активную рабочую книгу активного окна Excel
ActiveSheet	Возвращает активный лист активной рабочей книги
ActiveDialog	Возвращает активное диалоговое окно активного рабочего листа
ActiveChart	Возвращает активную диаграмму активного рабочего листа
ActiveCell	Возвращает активную ячейку активного рабочего листа

Суть программирования на VBA как раз и заключается в этих двух понятиях: *событие* и *отклик* на него. Если пользователь производит какое-то воздействие на систему, скажем, нажимает кнопку, тогда в качестве отклика выполняется код созданной пользователем процедуры. Если такой отклик не создан, т. е. не написана соответствующая процедура, то система никак не реагирует на данное событие, и оно остается безответным. Как говорится, на нет и суда нет. Таким образом, действия, происходящие в системе, являются событиями, а отклики на них – процедурами.

Этот специальный вид процедур, генерирующих отклик на события, называется *процедурами обработки событий.* В целом программирование на VBA состоит в создании кода программ, которые генерируют прямо или косвенно отклики на события.

Пример 1. Цикл For. Next

Синтаксис:

For *counter* = *Start* To *End* [Step *StepSize*]

Statements

Next [*counter*]

где Counter – любая численная переменная VBA;

Start – любое численное выражение, определяет начальное значение для переменной Counter;

End – численное выражение, определяет конечное значение для переменной Counter;

Statements – один, несколько или ни одного оператора VBA (тело цикла).

По умолчанию VBA увеличивает переменную *Counter* на 1 каждый раз при выполнении операторов в цикле. Можно задать другое значение (*StepSize* – любое численное выражение), на которое будет изменяться *Counter*.

Ключевое слово *Next* сообщает VBA о том, что достигнут конец цикла. Необязательная переменная *Counter* после ключевого слова *Next* должна быть той же самой переменной *Counter*, которая была задана после ключевого слова *For* в начале структуры цикла.

На рис. 26. представлен листинг простейшего цикла *For. Next*, который считает сумму цифр от 1 до 10:

```
Sub example_01()

    Dim i As Integer, SUM As Integer

    SUM = 0

    For i = 1 To 10
        SUM = SUM + i
    Next i

    MsgBox "Сумма чисел от 1 до 10 равна " & SUM

End Sub
```

Рис.26. Программный код вычисления суммы

Замечание: При уменьшении счетчика цикла *For. Next* цикл выполняется, пока переменная счетчика больше или равна конечному значению, а когда счетчик цикла увеличивается, цикл выполняется, пока переменная счетчика меньше или равна конечному значению.

Пример 2 (рис. 27)

Пример. Вычислить значение функции F=x*cos(x) для всех x, изменяющихся от 0 до 1 с шагом dx=0,1.

Текст программы.

```
Sub Пример_7()
For x=0 To 1.01 Step 0.2
F=x*cos(x)
Debug.Print x, F
Next
End Sub
```

Рис.27. Пример вычисления значения функции

Результат выполнения задания приведен на рис. 28 (View-Immediate Window или Ctrl+G). Задачи построения таблиц, содержащих аргументы функций и соответствующие им значения функций называются задачами табулирования функции.

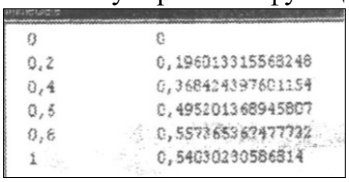


Рис.28. Результат вычисления значений функции для заданных аргументов

Цикл For Each. Next. Цикл *For Each. Next* не использует счетчик цикла. Циклы *For Each. Next* выполняются столько раз, сколько имеется элементов в определенной группе. Проще говоря, цикл *For Each. Next* выполняется один раз для каждого элемента в группе.

Синтаксис:

For Each *Element* In *Group*

Statements

Next [*Element*]

где Element – переменная, используемая для итерации по всем элементам в определенной группе;

Group – это объект коллекции или массив;

Statements – один, несколько или ни одного оператора VBA (тело цикла).

Пример 3. Вычислить факториал

Программный код:

```
Sub P1()
    'С помощью Dim создаём в оперативной памяти компьютера переменную i.
    'Long – означает, что в переменной будут только целые числа.
    'Double – означает, что в переменной будут вещественные числа (это целые и дробные).
    'В данном случае Double используется, потому что Double содержит большие числа, чем Long, а не потому что нам нужны дробные числа.
```

```
    Dim i As Long
    Dim N As Long
    Dim Факториал As Double
    'InputBox используется для помещения в переменные данных.
    N = InputBox("Введите число, для определения его факториала")
    Факториал = 1
    'For ... To ... Step ... Next - называется циклом и используется
    'для выполнения одного и того же действия заданное количество раз.
    For i = 1 To N Step 1
        Факториал = Факториал * i
    Next i  'Next увеличивает переменную i на число, указанное в Step.
    'MsgBox используется для вывода на монитор сообщений.
    MsgBox Факториал
End Sub
```

Пример 4. Вычислить произведение ряда:

$$\prod_{i=1}^{n} \frac{i \cdot x}{(2i-1)}$$

Программный код:

```
Sub Main()
    Dim x, n, res As Double
    x = 3
    n = 4
    res = 1
    For i = 2 To n
        res = res * (i * x / (2 * i - 1))
    Next i
    MsgBox "res=" & CStr(res)
End Sub
```

Пример 5. Визуальное представление ввода данных и вывода результата в активный лист активной рабочей книги (рис. 29).

Программный код:

```
Private Sub CommandButton4_Click()
        Dim x, n, res As Double
        x = Cells(36, 1)
        n = Cells(36, 2)
        res = 1
        For i = 2 To n
                res = res * (i * x / (2 * i - 1))
        Next i
        Cells(38, 2) = res
End Sub
```

Рис. 29. Результат работы программы

Конструкция Do...Loop

Цикл Do применяется для выполнения блока операторов неограниченное число раз. Существует несколько разновидностей конструкции Do . . . Loop, но каждая из них вычисляет выражение-условие, чтобы определить момент выхода из цикла. Как и в случае конструкции If . . . Then условие должно быть величиной или выражением, принимающими значение False (нуль) или True (не нуль).

В следующей конструкции Do . . . Loop операторы выполняются до тех пор, пока значением условия является True (Истина):

```
Do While условие
   операторы
Loop
```

Выполняя этот цикл, VBA сначала проверяет условие. Если условие ложно (False), он пропускает все операторы цикла. Если оно истинно (True), VBA выполняет операторы цикла, снова возвращается к оператору Do While и снова проверяет условие.

Следовательно, цикл, представленный данной конструкцией, может выполняться любое число раз, пока значением условия является не нуль или True (Истина). Отметим, что операторы тела цикла не выполняются ни разу, если при первой проверке условия оно оказывается ложным (False).

Пример 6. Вычислить сумму ряда с заданной точностью.

$$S = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots (-1)^{i-1} \frac{x^i}{i} + \ldots$$

Программный код:

```
Option Explicit
Sub Вложенные_циклы()
    Dim e As Single, x As Single, s As Single
    Dim m As Single, p As Single, i As Single
    x = InputBox("Введите x")
    e = InputBox("Введите e")
    s = 0: i = 1: m = 1: p = -1
    Debug.Print ("Шаг     Величина члена    Значение суммы")
    Do While Abs(m) >= e
            p = -p * x
            m = p / i
            s = s + m
            Debug.Print ("  " & i & "   " & Abs(m) & "    " & s)
            i = i + 1
    Loop
End Sub
```

Замечание: При использовании в файле **Option Explicit On** или **Option Explicit** необходимо явно объявлять все переменные с помощью инструкций Dim или ReDim.

Разновидности конструкции цикла Do Loop Until аналогичны предыдущим, за исключением того, что в этой конструкции цикл

выполняется по крайней мере один раз и цикл выполняется, пока условие ложно (False).

Синтаксис:

Do

операторы

Loop Until условие

Пример 7. Вычислить сумму с заданной точностью.

Программный код:

```
…
Dim i As Long
Dim Сумма As Double
i = 1
'Do ... Loop – называется циклом и используется, когда
'неизвестно, сколько раз нужно сделать одно и то же действие.
'Цикл продолжается до тех пор, пока не выполнится заданное условие.
Do Until 1 / i < 0.001
    Сумма = Сумма + 1 / i
    i = i + 1
Loop
MsgBox Сумма
…
```

Инструкция выбора Select Case

Инструкция выбора Select Case () предназначена для программирования разветвлений, она позволяет выполнить одну из нескольких групп операторов в зависимости от значения некоторого выражения. Инструкция Select Case имеет следующий синтаксис:

Select Case *Проверочное выражение*
*[**Case** Значение1*
[Инструкции1]]
*[**Case** Значение2*
[Инструкции2]]
[Case Else
[Инструкции3]]
End Select

Пример 8. Пример анализа введенного числа приведен на рис. 30.

```
Sub pr2()
V = InputBox("Введите V")
Select Case V
Case 1
Debug.Print " Равно 1"
Case 2, 3
Debug.Print " Равно 2 или 3"
Case 4 To 6
Debug.Print " Больше или равно 4 и меньше или равно 6"
Case Is >= 9
Debug.Print " Больше или равно 9"
Case Else
Debug.Print "Ни одно из предшествующих"
End Select
End Sub
```

Рис. 30. Пример анализа введенного числа

Технология выполнения лабораторной работы

1. Выполнить все примеры (8 примеров)

2. Разработать алгоритм для написания программ в виде блок-схемы (6 задач по вариантам).

3. Написать программы (6 задач по вариантам).

4. Протестировать программы с разными исходными данными.

5. Написать программы для задания 3 лабораторной работы №1 (3 задачи).

6. Протестировать их с разными исходными данными.

Итого: 8 примеров, 6 задач (задача 1 – 4 варианта, задача 4 – 4 варианта) (+ блок-схема для каждой) и 3 задачи из задания №3 ЛР№1. Всего 17 задач.

7. Создать отчет.

Задание 1. Найти сумму (произведение ряда – рис. 31) четырьмя способами (табл. 4):

Таблица 4

Способы решения

1	2	3	4
Do While условие Операторы Loop	Do Операторы Loop While условие	Do Until условие Операторы Loop	Do Операторы Loop Until условие

Написать программу для вычисления функции ***f***. Полученное значение функции вывести на экран.

1. $f = 20\sum_{i=1}^{30} \frac{Ln(i^2)}{i}$	2. $f = 0{,}1\prod_{i=2}^{60} i^8 \cdot Ln\sqrt{i}$
3. $f = 35 \sum_{i=2,4,6,8} \left(i^2 - Ln(i)\right)$	4. $f = 3{,}14 \prod_{i=3,6,9,12,15} i\left(4 - e^i\right)$

Рис. 31. Варианты заданий

Задание 2. Построить таблицу значений функции *y=F(x)* с учетом проверки основных видов неопределенностей:

– не существует логарифм нуля и отрицательного числа;

– неопределен квадратный корень из отрицательного числа;

– недопустимо деление на ноль (знаменатель равен нулю), функции arcsin(x) и arccos(x) неопределены, если |X| >1.

Таблица 5

Варианты заданий

№ варианта	Функция *y=F(x)*	№ варианта	Функция *y=F(x)*
1	$\frac{Sin(x)}{x} e^{-x}$	6	$\frac{Ln(1+\pi x)}{x} e^{-x}$
2	$\frac{1-x}{x} Lg(1+x)$	7	$\frac{Tg(x)}{x} e^{-x}$
3	$\frac{ArcTg(x)}{x} e^{-x}$	8	$x^x e^{-x} Lg(1+x)$
4	$x^5 e^{-x/Sin(x)}$	9	$\frac{(1-x)(1-x^2)}{1-x^5}$
5	$10^{1-x} \frac{Ln(1+x)}{x}$	10	$\frac{Ln(1+x)}{x} - \frac{Sin^2(x}{x^2}$

Задание 3. Варианты заданий

1. Найти сумму пяти чисел, введенных с клавиатуры. Ввод осуществить в цикле. Найти среднее арифметическое этих чисел – сумму чисел, деленную на количество.

2. Найти произведение первых 10 членов натурального ряда.

3. Найти сумму корней 5 чисел, введенных с клавиатуры. Сделать проверку на не отрицательность чисел и в случае отрицательности попросить пользователя заново ввести положительное число.

4. Вычислить $y = \sqrt{x-1}$, если a<=x<=b с шагом h. Сделать проверку условия, что подкоренное выражение неотрицательно (*x*-1 >= 0).

5. Составить программу перевода температуры в градусах по Цельсию в температуру по Кельвину. Перевести температуру от 0^0 до 100^0C через каждые 20^0C в температуру по Кельвину. Формула перевода K=T-273. Результат вывести в два столбца (по Цельсию, по Кельвину).

6. Найти сумму чисел, введенных с клавиатуры. Вычисление прекратить, когда сумма чисел будет больше 15.

Задание 4. Выполните пошагово программу *Выбор* (рис. 32) для разных нечетных значений ***x∈[1, 5]*** и модифицируйте ее для четных значений ***x.***

```
Sub Выбор()
Dim x As Byte, y As Integer
  x = 1
  Select Case 2 * x - 1
       Case 1
            y = (x + 2) ^ 5
       Case 3
            y = (x + 1) ^ 4
       Case 5, 7, 9
            y = x ^ 3
       Case Else
            y = (x -1) ^ 2
  End Select
End Sub
```

Рис. 32. Программа Выбор

Задание 5. Ввести номер месяца и по номеру месяца выдать сообщение о времени года – зима, весна, лето, осень. Использовать инструкцию выбора Select Case.

Задание 6. Варианты заданий (4 варианта решения)

1. Найти сумму ряда чисел от 1 до 10, используя циклы: с предусловием (пока ложь), с постусловием (до тех пор, пока истина) и с параметром. При решении задания применить оператор выбора варианта Select Case.

2. Найти сумму ряда чисел от 1 до 10, используя циклы: с предусловием (пока истина), с постусловием (до тех пор, пока истина) и с параметром. При решении задания применить оператор выбора варианта Select Case.

3. Найти сумму ряда чисел от 1 до 10, используя циклы: с предусловием (пока истина), с постусловием (до тех пор, пока ложь) и с параметром. При решении задания применить оператор выбора варианта Select Case.

4. Найти сумму чисел применив функцию Sin(x), 1<=x<=10, используя циклы: с предусловием (пока ложь), с постусловием (до тех пор, пока истина) и с параметром. При решении задания применить оператор выбора варианта Select Case.

5. Найти сумму чисел применив функцию Cos(x), 1<=x<=10, используя циклы: с предусловием (пока истина), с постусловием (до тех пор, пока истина) и с параметром. При решении задания применить оператор выбора варианта Select Case.

6. Найти сумму чисел применив функцию Sin(x)+Cos(x), 1<=x<=10, используя циклы: с предусловием (пока истина), с постусловием (до тех пор, пока ложь) и с параметром. При решении задания применить оператор выбора варианта Select Case.

7. Найти произведение чисел, применив функцию Sin(x), 1<=x<=10, используя циклы: с предусловием (пока ложь), с постусловием (до тех пор, пока истина) и с параметром. При решении задания применить оператор выбора варианта Select Case.

8. Найти произведение чисел, применив функцию Cos(x), 1<=x<=10, используя циклы: с предусловием (пока истина), с постусловием (до тех пор, пока истина) и с параметром. При решении задания применить оператор выбора варианта Select Case.

9. Найти произведение чисел, применив функцию Sin(x)+Cos(x), 1<=x<=10, используя циклы: с предусловием (пока истина), с постусловием (до тех пор, пока ложь) и с параметром. При решении задания применить оператор выбора варианта Select Case.

10. Найти разность чисел, применив функцию Sin(x)+Cos(x), 1<=x<=10, используя циклы: с предусловием (пока истина), с постусловием (до тех пор, пока ложь) и с параметром. При решении задания применить оператор выбора варианта Select Case.

11. Найти сумму целых чисел применив функцию Int(Cos(x)*25), 1<=x<=10, используя циклы: с предусловием (пока истина), с постусловием (до тех пор, пока истина) и с параметром. При решении задания применить оператор выбора варианта Select Case.

12. Найти сумму целых чисел применив функцию Fix((Sin(x)+Cos(x))*40), 1<=x<=10, используя циклы: с предусловием (пока истина), с постусловием (до тех пор, пока ложь) и с параметром. При решении задания применить оператор выбора варианта Select Case.

Отчет, подготовленный в MS Word, должен содержать:

1. Титульный лист
2. Цель работы

Для каждой задачи

3. Условие задачи
4. Алгоритм в виде блок-схемы. Блок-схему выполнить в специализированной программе в соответствии с ГОСТ 19.701-90.
5. Текст программного кода.
6. Вводимые данные и результаты, фрагменты экрана с диалоговыми окнами Excel.
7. Результат работы программы должен иметь наглядный вид.

Контрольные вопросы:

1. Какая конструкция у оператора цикла с предусловием?
2. Какая конструкция у оператора цикла с постусловием?
3. Какие операторы циклов Вы знаете?
4. Приведите пример вложенного (сложного) цикла?

5. Какие требования предъявляются при составлении сложных циклов?

6. Как выглядит полная форма оператора с параметром (по счетчику)?

7. Какие выражения применяются при использовании операторов цикла с предусловием и с постусловием?

ЛАБОРАТОРНЫЕ РАБОТЫ № 5. СОЗДАНИЕ, ТЕСТИРОВАНИЕ, ОТЛАДКА, ОЦЕНКА И АНАЛИЗ ПОЛУЧЕННОГО РЕЗУЛЬТАТА ПРОГРАММЫ ОБРАБОТКИ МАССИВОВ ДАННЫХ

Массив

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.

Пример 1:

```
Option Base 1
    Sub Mas1()
        Dim a(5) As Integer
        Dim i As Integer, k As Integer
        WorkSheets("Лист1").Select 'Выбрать Лист1 из семейства
        листов
        Cells.Clear 'Очистить ячейки рабочего листа
        K=2
        For i=1 To 5
            a(i)=Int(Rnd(i)*100) 'Формирование массива a(i)
            Cells(k , i+1)=a(i) 'Вывод массива a(i) на рабочий лист.
        Next i
    End Sub
-------------------------------------------------------------------------------------
Sub Mas2()
        Dim a(1 to 5) As Integer, k As Integer
        Dim i As Integer
        WorkSheets("Лист1").Select
        Cells.Clear
        K=2
        For i = 1 To 5
          a(i)=Int(Rnd(i)*100) 'Формирование одномерного массива a
          Cells(k , i+1)=a(i) 'Вывод элементов массива a(i) на рабо-
        чий лист
        Next I
End Sub
-------------------------------------------------------------------------------------
```

```
Sub Dmas1()
        Dim i As Integer
        Dim j As Integer
        Dim a2(1 to 3, 1 to 5) As Integer
        WorkSheets("Лист1").Select
        Cells.Clear
        For i=1 To 3 'Формирование двумерного массива a2
            For j=1 To 5
               a2(i , j)=Int(Rnd(i*j)*100)
           Next j
        Next i
        For i = 1 To 3 'Вывод элементов массива a2(i , j) на рабочий
        лист
            For j = 1 To 5
               Cells(i+1 , j+1) = a2(i , j)
           Next j
        Next i
End Sub
-----------------------------------------------------
Sub Dmas2()
        Dim a(1 to 5) As Integer, k As byte, i As byte,
        prom As Variant
        WorkSheets("Лист1").Select
        Cells.Clear
        K=2
        For i=1 To 5 'Ввод элементов массива a(i)  с клавиатуры
          Do
           Prom=InputBox("Введите элемент a(" & CInt(i) & ")= ")
           If Not IsNumeric(prom)Then MsgBox("Повторите ввод!")
          Loop Until IsNumeric(prom)
          a(i)=prom
        Next i
        For i=1 To 5 'Вывод элементов массива a(i) на рабочий лист
           Cells(k, 2)= "a(" & CInt(i) & ") = "
           Cells(k ,3)=a(i)
           k=k+1
        Next i
End Sub
```

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>)*. Такую инструкцию необходимо применять, например, при создании нового массива из существующего.

Пример 2:

```
Sub Mas4()
    Dim a() As Integer, i, k, j, N As byte
    WorkSheets("Лист1").Select
    Cells.Clear
    k=2
    i=1
    Do
        N=InputBox("Введите количество элементов N=")
        If Not IsNumeric(N)Then MsgBox("Повторите ввод!")
    Loop Until IsNumeric(N)
    ReDim a(1 to N) As Integer 'Устанавливается фактическая размерность массива а
    Do 'Формирование массива
        a(i)=Int(Rnd(i)*100)
        i=i+1
    Loop Until  i>=N
    For j=1 To N 'Вывод массива а на рабочий лист
       Cells(k + j, 3)=a(j)
    Next j
End Sub
```

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.

Технология выполнения лабораторной работы

1. Выполнить все примеры.
2. Разработать алгоритм для написания программ в виде блок-схемы (4 задачи).
3. Составить код программы и ввести его в окно редактора VBA.
4. Протестировать программу и отладить ее при необходимости.
5. Результаты решения должны соответствовать поставленной задаче.

При выполнении работы необходимо учесть следующие моменты:

1. Необходимо учесть следующие моменты предусмотреть сбойные ситуации (деление на ноль, извлечение квадратного корня из отрицательного числа и т. д.).

2. Результат работы программы должен иметь наглядный вид.

Итого: 2 примера, 4 задачи (+ блок-схема для каждой). Всего 6 задач.

Отчет, подготовленный в MS Word, должен содержать:

1. Титульный лист
2. Цель работы
   Для каждой задачи
3. Условие задачи
4. Алгоритм в виде блок-схемы. Блок-схему выполнить в специализированной программе в соответствии с ГОСТ 19.701-90.
5. Текст программного кода.
6. Вводимые данные и результаты – фрагменты экрана с диалоговыми окнами Excel.

Варианты заданий:

Одномерные массивы (выбрать 2 задачи по варианту)

1. Сформировать массив **N** чисел, среди которых могут быть как положительные, так и отрицательные числа. Определить сумму и количество только отрицательных значений.
2. Сформировать массив **N** целых чисел. Определить сумму чисел, имеющих четные порядковые номера.
3. Сформировать массив **N** целых чисел. Подсчитать количество нулевых элементов и исключить их из массива.
4. Сформировать массив **N** целых чисел. Определить наличие среди них одинаковых. Нулевые значения не учитывать.
5. Сформировать массив **N** чисел, среди которых должны быть как положительные, так и отрицательные значения. Определить, сумма, каких чисел больше по абсолютной величине.
6. Сформировать массив **N** слов произвольной длины и найти самое длинное из них.
7. Сформировать массив **N** символов. Определить их коды и из них сформировать новый числовой массив
8. Сформировать массив **N** слов произвольной длины. Определить длину каждого из них и сформировать числовой массив.
9. Сформировать массив **N** целых чисел, удовлетворяющих условию: 32 =< a(i) <= 256. Считая эти числа кодами символов, определить эти символы и сформировать из них массив.

10. Сформировать массив **N** чисел, среди которых должны быть как положительные, так и отрицательные значения. Выбрать все положительные и все отрицательные числа и записать их в отдельные массивы.

11. Сформировать числовой массив заданной размерности. В полученных числах отделить целую часть, и записать в другой массив.

12. Из заданной строки текста сформировать массив символов, упорядочить его по возрастанию, преобразовать символы в коды и записать их в другой массив.

Двумерные массивы (выбрать 2 задачи по варианту)

1. Сформировать числовой массив из **M** строк и **N** столбцов. Подсчитать сумму значений элементов каждого столбца.
2. Создать строку из 25 букв русского алфавита. Используя функцию **Mid**, из букв этой строки сформировать массив, включающий 5 строк и 5 столбцов.
3. Создать строку из 25 букв русского алфавита. Определить код каждой буквы и сформировать числовой массив из 5 строк и 5 столбцов
4. Сформировать числовой массив из **M** строк и **N** столбцов. Подсчитать сумму значений элементов каждой строки.
5. Сформировать массив натуральных чисел из **M** строк и **N** столбцов. Подсчитать количество и сумму всех чётных чисел массива.
6. Сформировать массив натуральных чисел из **M** строк и **N** столбцов. Подсчитать количество и сумму всех нечётных чисел массива.
7. Сформировать массив натуральных чисел из **M** строк и **N** столбцов. Подсчитать сумму чисел в чётных строках массива.
8. Сформировать массив натуральных чисел из **M** строк и **N** столбцов. Подсчитать сумму чисел в нечётных строках массива.
9. Сформировать массив натуральных чисел из **M** строк и **N** столбцов. Подсчитать сумму чисел в чётных столбцах массива.
10. Сформировать массив натуральных чисел из **M** строк и **N** столбцов. Определить номер строки, сумма чисел которой наибольшая.

11. Сформировать массив натуральных чисел из **M** строк и **N** столбцов. Определить номер столбца, в котором число нечётных элементов больше числа чётных элементов.

12. Сформировать массив натуральных чисел из **M** строк и **N** столбцов. Выбрать из массива все числа, которые делятся на 3 без остатка.

Контрольные вопросы:

1. Что такое массив?
2. Как объявляются статические массивы (одномерный и двумерный)? Приведите примеры.
3. Как объявляется динамический массив? Приведите примеры.
4. В каких случаях необходимо применять динамический массив?
5. Какой оператор нужно использовать после определения фактической размерности массива?
6. В каком случае необходимо применять оператор установки фактической размерности массива с сохранением его элементов?

ЛАБОРАТОРНАЯ РАБОТА №6. ПОДПРОГРАММЫ
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
```

Другой пример:

```
Sub qwer(ByVal x As Single, y As Single,
       ByRef S As Single)
  Dim Z As Single
  Dim P As Single
  Z=Sin(2*x+3*y)
  P=Cos(x^2+y^3)
  S=Z+P
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.

Пример. Построение графиков функции

Задача: Построить график функции $y(x)=\begin{cases} x\cdot\sin(x), x<0; \\ x\cdot\sqrt{x}, x\geq 0, \end{cases}$ $x\in[-2;2]$, $h=0{,}1$.

Цикл с предусловием

```
Sub ex2()
Dim x, b, y, h As Single
Dim i As Byte
ActiveSheet.ChartObjects.Delete
ActiveSheet.Cells.Select
Selection.Clear
i = 1
x = -2
b = 2
h = 0.1
Cells(1, 1) = "x"
Cells(1, 2) = "y"
Do While x <= b
    If x < 0 Then
        y = x * Sin(x)
    Else
        y = x * Sqr(x)
    End If
    Cells(i + 1, 1) = x
    Cells(i + 1, 2) = y
    i = i + 1
    x = x + h
Loop
Range("B2:B41").Select
ActiveSheet.Shapes.AddChart.Select
ActiveChart.SetSourceData Source:=Range("'Лист1'!$B$2:$B$41")
ActiveChart.ChartType = xlLineMarkers
ActiveChart.SeriesCollection(1).XValues = "='Лист1'!$A$2:$A$41"
ActiveChart.Legend.Select
Selection.Delete
ActiveChart.SetElement (msoElementChartTitleAboveChart)
ActiveChart.ChartTitle.Text = "График функции y(x)"
End Sub
```

Цикл с постусловием

```
Sub ex3()
Dim x, b, y, h As Single
Dim i As Byte
ActiveSheet.ChartObjects.Delete
ActiveSheet.Cells.Select
Selection.Clear
i = 1
x = -2
b = 2
h = 0.1
Cells(1, 1) = "x"
Cells(1, 2) = "y"
Do
    If x < 0 Then
        y = x * Sin(x)
    Else
        y = x * Sqr(x)
    End If
    Cells(i + 1, 1) = x
    Cells(i + 1, 2) = y
    i = i + 1
    x = x + h
Loop Until x >= b
Range("B2:B41").Select
ActiveSheet.Shapes.AddChart.Select
ActiveChart.SetSourceData Source:=Range("'Лист1'!$B$2:$B$41")
ActiveChart.ChartType = xlLineMarkers
ActiveChart.SeriesCollection(1).XValues = "='Лист1'!$A$2:$A$41"
ActiveChart.Legend.Select
Selection.Delete
ActiveChart.SetElement (msoElementChartTitleAboveChart)
ActiveChart.ChartTitle.Text = "График функции y(x)"
End Sub
```

Цикл с параметром

```
Sub ex1()
Dim x, a, b, y, h As Single
Dim i As Byte
ActiveSheet.ChartObjects.Delete
ActiveSheet.Cells.Select
Selection.Clear
i = 1
a = -2
b = 2
h = 0.1
Cells(1, 1) = "x"
Cells(1, 2) = "y"
For x = a To b Step h
    If x < 0 Then
        y = x * Sin(x)
    Else
        y = x * Sqr(x)
    End If
    Cells(i + 1, 1) = Round(x, 3)
    Cells(i + 1, 2) = Round(y, 3)
    i = i + 1
Next
Range("B2:B41").Select
ActiveSheet.Shapes.AddChart.Select
ActiveChart.SetSourceData Source:=Range("'Лист1'!$B$2:$B$41")
ActiveChart.ChartType = xlLineMarkers
ActiveChart.SeriesCollection(1).XValues = "='Лист1'!$A$2:$A$41"
ActiveChart.Legend.Select
Selection.Delete
ActiveChart.SetElement (msoElementChartTitleAboveChart)
ActiveChart.ChartTitle.Text = "График функции y(x)"
End Sub
```

Использование функции и процедур

```
Sub ex5()
Dim b, y, h As Single
Dim x As Single
Dim i As Byte
ActiveSheet.ChartObjects.Delete
ActiveSheet.Cells.Select
Selection.Clear
i = 1
a = -2
b = 2
h = 0.1
Cells(1, 1) = "x"
Cells(1, 2) = "y"
For x = a To b Step h
    Cells(i + 1, 1) = Round(x, 3)
    Cells(i + 1, 2) = myfun(x) ' вызов функции
    i = i + 1
Next
Call graf ' вызов процедуры для построения графика
Call format ' вызов процедуры для оформления
End Sub
```

```
Function myfun(x As Single) As Single
If x < 0 Then
        myfun = x * Sin(x)
    Else
        myfun = x * Sqr(x)
End If
End Function
```

```
Sub graf()
Range("B2:B41").Select
ActiveSheet.Shapes.AddChart.Select
ActiveChart.SetSourceData Source:=Range("'Лист1'!$B$2:$B$41")
ActiveChart.ChartType = xlLineMarkers
ActiveChart.SeriesCollection(1).XValues = "='Лист1'!$A$2:$A$41"
ActiveChart.Legend.Select
Selection.Delete
ActiveChart.SetElement (msoElementChartTitleAboveChart)
ActiveChart.ChartTitle.Text = "График функции y(x)"
End Sub
```

```
Sub format()
Range("A1:B41").Select
    ActiveWindow.SmallScroll Down:=-15
    Selection.Borders(xlDiagonalDown).LineStyle = xl
    Selection.Borders(xlDiagonalUp).LineStyle = xlNo
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeRight)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlInsideHorizontal)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
```

График + форма

N21

	A	B
1	x	y
2	-2	1,819
3	-1,9	1,798
4	-1,8	1,753
5	-1,7	1,686
6	-1,6	1,599
7	-1,5	1,496
8	-1,4	1,38
9	-1,3	1,253
10	-1,2	1,118
11	-1,1	0,98
12	-1	0,841
13	-0,9	0,705
14	-0,8	0,574
15	-0,7	0,451
16	-0,6	0,339
17	-0,5	0,24
18	-0,4	0,156
19	-0,3	0,089
20	-0,2	0,04
21	-0,1	0,01
22	0	0
23	0,1	0,032

```
Private Sub CmbCancel_Click()
FrmGraf.Hide
End Sub
```

```
Private Sub CmBOK_Click()
Dim a, b, y, h As Single
Dim x As Single
Dim i As Byte
Sheets("График").Select
If ActiveSheet.ChartObjects.Count > 1 Then ActiveSheet.ChartObjects.Delete
ActiveSheet.Cells.Select
Selection.Clear
If IsNumeric(TxtA.Text) = False Then
    MsgBox ("Начальное значение введено некорректно")
    TxtA = ""
    TxtA.SetFocus
    Exit Sub
End If
If IsNumeric(TxtH.Text) = False Then
    MsgBox ("Значение шага введено некорректно")
    TxtH = ""
    TxtH.SetFocus
    Exit Sub
End If
If IsNumeric(TxtB.Text) = False Then
    MsgBox ("Конечное значение введено некорректно")
    TxtB = ""
    TxtB.SetFocus
    Exit Sub
End If
a = CSng(TxtA.Text)
b = CSng(TxtB.Text)
h = CSng(TxtH.Text)
If a >= b Then
    MsgBox ("Начальное значение больше конечного значения. Повторите ввод")
    TxtA.SetFocus
    Exit Sub
End If
    If a + h >= b Then
    MsgBox ("Шаг слишком большой. Повторите ввод.")
    TxtH.SetFocus
    Exit Sub
End If
i = 1
Cells(1, 1) = "x"
Cells(1, 2) = "y"
For x = a To b Step h
    Cells(i + 1, 1) = Round(x, 3)
    Cells(i + 1, 2) = myfun(x)
    i = i + 1
Next
Call graf
Call format
End Sub
```

```
Sub graf()
Range("B2:B41").Select
ActiveSheet.Shapes.AddChart.Select
ActiveChart.SetSourceData Source:=Range("'График'!$B$2:$B$41")
ActiveChart.ChartType = xlLineMarkers
ActiveChart.SeriesCollection(1).XValues = "='График'!$A$2:$A$41"
ActiveChart.Legend.Select
Selection.Delete
ActiveChart.SetElement (msoElementChartTitleAboveChart)
ActiveChart.ChartTitle.Text = "График функции y(x)"
ActiveChart.Export "graph.JPEG"
FrmGraf.Image1.Picture = LoadPicture("graph.JPEG")
ActiveSheet.Range("A1").Select
End Sub
```

```
Sub format()
Range("A1:B41").Select
    ActiveWindow.SmallScroll Down:=-15
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeTop)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeRight)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlInsideVertical)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    With Selection.Borders(xlInsideHorizontal)
        .LineStyle = xlContinuous
        .ColorIndex = 0
        .TintAndShade = 0
        .Weight = xlThin
    End With
    Range("A1:B1").Select
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Selection.Font.Bold = True
End Sub
```

Задание

Построить графики функций – «Численное решение задач в в пакетах Microsoft Excel, MathCAD и MatLab» с.44 (по вариантам). Сравнить полученные результаты с MS Excel, MathCad.

В работе необходимо использовать:

1. Разные виды циклов (1, 2, 3)
2. Обращение к подпрограммам (процедуры и функции) (4, 5)
3. Модули + формы (6, 7, 8)
4. Сделать проверки ввода данных (6, 7, 8)

Примеры см. выше. Для построения графика проще всего сначала записать макрос.

Отчет, подготовленный в MS Word, должен содержать:

1. Титульный лист
2. Цель работы

Для каждой задачи

3. Условие задачи
4. Алгоритм в виде блок-схемы. Блок-схему выполнить в специализированной программе в соответствии с ГОСТ 19.701-90.
5. Текст программного кода.
6. Вводимые данные и результаты – фрагменты экрана с диалоговыми окнами Excel.
7. Результат работы программы должен иметь наглядный вид.

СОДЕРЖАНИЕ