МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Государственное образовательное учреждение
высшего профессионального образования

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «МАМИ»

В.И. Антомони, В.Н Архипов, А.Н. Любин, В.Н. Тихомиров

Программирование на VBA в Microsoft Office

Сборник лабораторных работ

по дисциплине «Информатика» для студентов всех специальностей

Москва

2011

УДК 681.3.06

Разработано в соответствии с Государственным образовательным стандартом 2008 г. Для всех специальностей на основе примерной программы дисциплины «Информатика»

Рецензенты: к.т.н., доцент кафедры «Автоматизация процессов управления»
Ю.А. Савостьянок;
к.т.н., с.н.с.,ФГУП «ИНИСТИ им. акад.А. И. Берга»
В.Ф. Блохина.

Работа подготовлена на кафедре «Информационные системы и дистанционные технологии»

Антомони В.И., Архипов В.Н, Любин А.Н., Тихомиров В.Н. Программирование на VBA в Microsoft Office. Сборник лабораторных работ по дисциплине «Информатика» для студентов всех специальностей. М.: МАМИ 2011, 152 с.:ил.

Пособие ориентировано на изучение основ языка программирования Visual Basic for Applications, освоение программирования на этом языке и получение навыков решения задач на ПК.

Лабораторные работы № 1, 2 написаны В.Н. Архиповым, правила выполнения работ и лабораторные работы № 3, 4 написаны А.Н. Любиным, лабораторные работы № 5, 6 написаны В.И Антомони, введение и лабораторная работы №7, написаны В.Н. Тихомировым.

ВВЕДЕНИЕ

В настоящее время Visual Basic for Applications (VBA) – это современный язык программирования, позволяющий использовать все возможности операционной системы WINDOWS и пакета прикладных программ Microsoft Office для решения широкого круга прикладных задач, отвечающий современным требованиям структурного и объектно-ориентированного программирования (ООП).

Программа (приложение), создаваемая в VBA, входит в файл проекта. Проект – совокупность файлов, связанных с программой. Основными компонентами проекта являются файлы форм, файлы модулей, и др.

Язык программирования VBA – это язык, основанный на манипулировании ***объектами*** и их атрибутами. В VBA ***объект*** – это комбинация программного кода и данных, которая воспринимается как единое целое и которой можно каким-либо образом манипулировать. Объектами являются также команды меню, базы данных, аппаратные устройства вычислительной системы – принтеры, мониторы, диски, которыми можно манипулировать из программного кода. Особым видом объектов являются формы и элементы управления (ЭУ). Элементы управления позволяют инициировать определенные события, реагируя на которые можно управлять программой. Excel позволяет управлять более чем ста классами объектов, включая рабочую книгу, рабочий лист, диапазон ячеек рабочего листа, диаграмму и нарисованный прямоугольник.

Класс — разновидность абстрактного типа данных в объектно-ориентированном программировании, характеризуемый способом своего построения. Суть отличия классов от других абстрактных типов данных состоит в том, что при задании типа данных класс

определяет одновременно и интерфейс, и реализацию для всех своих экземпляров. Объект — экземпляр класса. В ООП при использовании классов весь исполняемый код программы будет оформляться в виде так называемых методов, функций или процедур, что соответствует обычному структурному программированию, однако теперь они могут принадлежать тому или иному классу. Классы объектов организованы в иерархическую структуру. Объекты могут выступать контейнерами других объектов.

Объекты обладают свойствами и методами. Каждый объект имеет собственный набор свойств и методов. Однако некоторые объекты характеризуются общими свойствами (например, *Name*) и методами (например, *Delete*).

События связаны с определенными действиями пользователя и могут вызывать код VBA. Методы – это рабочие операторы объекта, программные процедуры. Свойства отвечают за внешний вид и поведение объекта. Основное различие между ними заключается в том, что со свойствами можно работать как во время разработки проекта, так и во время его выполнения, тогда как методы доступны только при выполнении. Свойства и методы называются интерфейсом объекта. Объекты объединяются в классы. К одному классу принадлежат объекты с одинаковым набором свойств, методов и событий.

Манипулировать объектами можно двумя способами:

1) изменяя свойства объекта,
2) заставляя объект выполнять специфические задания активизацией методов, ассоциированных с этим объектом.

Оба эти способа часто ассоциируются с наступлением некоторого пользовательского (программного) или системного события, т. е. действия или ситуации, связанной с объектом.

Если объект должен выполнить действие, не входящее в круг его "обязанностей", он должен иметь доступ к объекту, который способен выполнить требуемое действие: 1-й объект передает 2-му запрос на выполнение действия с использованием модифицированных версий функций и процедур (аналогично структурному программированию) - объект-клиент передает сообщение объекту-серверу. При этом один объект никогда не должен манипулировать внутренними данными другого объекта. Вся связь должна осуществляться только через сообщения, т. е. объекты могут управлять только изменением свойств или вызовом методов. В программной реализации внутри создаваемых объектов-элементов управления не должно быть никаких общих переменных типа **public.**

Программные коды содержатся в процедурах и функциях, объединяемых в модули. Формы, классы и модули являются контейнерами для других элементов управления и объектов.

Функции и процедуры в VBA соответствуют методам и свойствам объекта.

Модуль любого типа может содержать не более 65534(64K-2) строк кода. Строка кода может содержать до 1023(1K-1) символа. Символ продолжения строки (перенос строки) содержит символ пробел, сопровождаемый символом подчеркивания (одна логическая строка кода не может содержать более 25 символов переноса).

Можно добавить код в модуль VBA двумя способами:

-ввести код традиционным способом с помощью клавиатуры, для этого надо войти в меню **Вид** и выбрать пиктограмму **Макросы** (Alt+F8), далее ввести имя макроса и выбрать пункт **Создать.** (Рис.1, Рис.2);

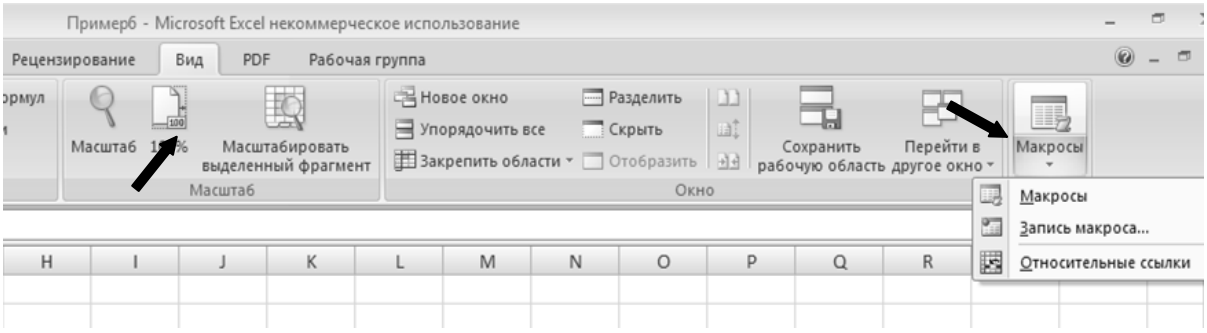


Рис.1

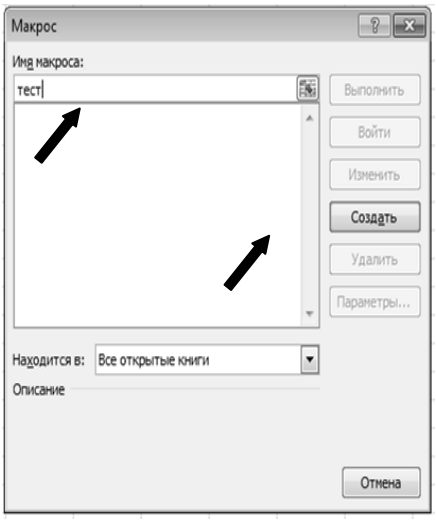


Рис.2

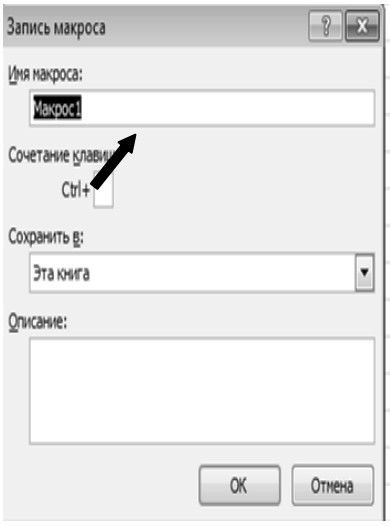


Рис.3

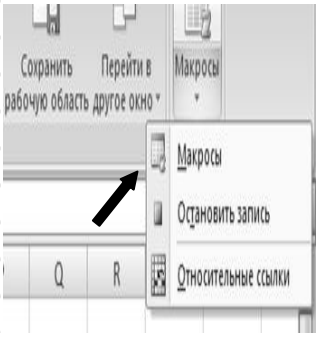


Рис.4

-использовать функцию записи макросов в Excel, чтобы записать действия и преобразовать их в код VBA(Рис.1, Рис.3).

Чтобы остановить запись надо вызвать пиктограмму **Макрос** и выбрать соответствующий пункт меню (Рис.1, Рис.4). Для просмотра и редактирования записи нажать Alt+F11. Программа Excel запустит интегрированную среду разработки VBA (Рис.5).

Под макросом здесь понимается записанная компьютером последовательность действий пользователя, которая может быть затем многократно повторена.

Описание среды VBA

В верхней части находится строка ***главного меню***, а под ней – ***панель инструментов***. Здесь собраны все команды, необходимые для создания и проверки разрабатываемого приложения.

Под ними, в левой верхней части экрана находится ***окно обозревателя*** (1) ***Project Explorer***, в котором представлена структура файлов форм и модулей текущего проекта, которые могут отображаться в окне редактора справа.

Ниже располагается ***окно свойств*** (3) ***properties***. В окне свойств перечисляются установки свойств для текущей формы или элемента управления.

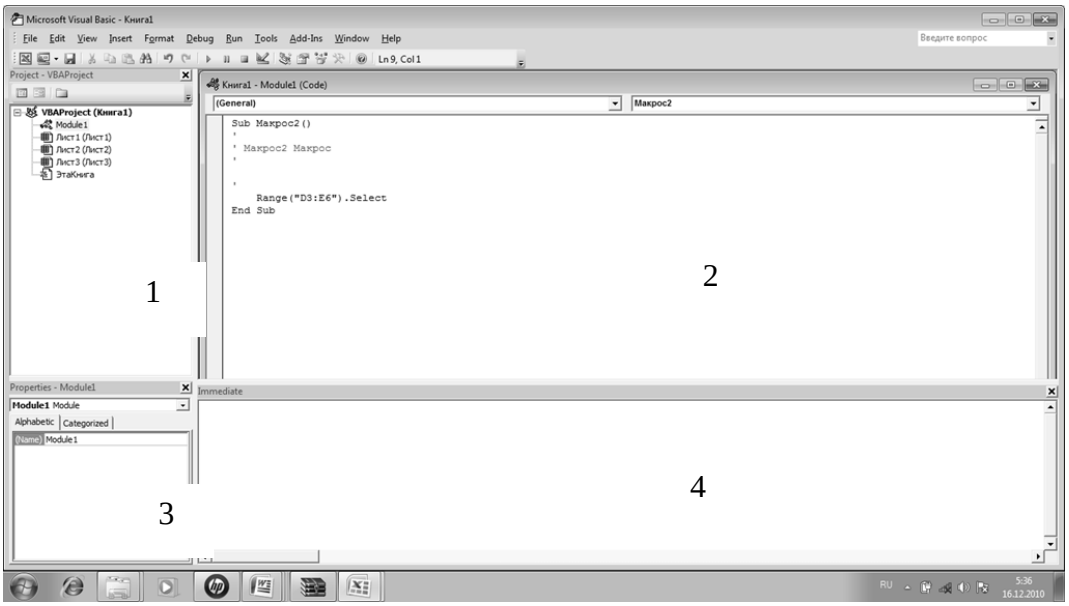


Рис.5

Правое верхнее окно – ***окно редактора кода*** (2). В нём отображается текст макроса, его можно редактировать. Под окном редактора обычно находится ***окно отладки*** (4) ***immediate*** (окно немедленного исполнения). В этом окне можно исполнить любую команду VBA или процедуру, или вывести отладочную информацию. Например: набрав ?2*2 получим ответ 4.

Чтобы запустить программу из окна редактора следует нажать F5, или воспользоваться меню Run. Возможно исполнение программы в режиме отладки меню Debug (нажать клавишу F8). Значение переменной во время отладки можно узнать, подведя указатель мыши к имени переменой в окне редактора.

ПРАВИЛА ВЫПОЛНЕНИЯ РАБОТ

При выполнении лабораторных работ студент обязан.

1. Заранее (дома) подготовиться к лабораторной работе. Для этого необходимо:

- изучить теоретическую часть к лабораторной работе, изложенную в методических указаниях (целесообразно использовать лекции и указанную в них литературу);
- выполнить задание своего варианта, изложенное в методических указаниях;
- оформить отчет по лабораторной работе по следующим правилам.

МГТУ « МАМИ»

Кафедра: «Информационные системы и дистанционные технологии»

Лабораторная работа № ?
Название (из методических указаний)
Вариант (выдает преподаватель)

Студент ФИО, группа
Преподаватель ФИО

Правила оформления отчёта.

Отчёт оформляется на листах формата А4, **строго рукописно** и включает в себя:

-титульный лист, с указанием учебного заведения (МГТУ «МАМИ»), кафедры (кафедра: «Информационные системы и дистанционные технологии»), номера лабораторной работы, её названия, варианта, ФИО и группы студента, ФИО преподавателя и наст. года;

-теоретическую часть (краткий конспект, минимум две страницы);

-текст задания (строго по тексту методических указаний);

-выполненное задание. Это может быть текст, таблицы, блок-схема, программа, прочее;

-исходные данные, данные для тестирования программы;

-место, оставленное для записи результатов, полученных на занятии с помощью ПК.

2.Представить подготовленный отчёт преподавателю.

3.Получить у преподавателя доступ к ПК и выполнить на нём своё задание.

4.Полученные на ПК результаты показать преподавателю, после чего записать их в отчет (можно в печатном виде).

5.Защитить лабораторную работу.

Защита лабораторной работы включает в себя выполнение контрольного задания преподавателя и ответы на его вопросы по теме лабораторной работы в объёме методических указаний и лекций.

Задание и вопросы преподавателя, решения и ответы студента письменно фиксируются на последнем листе отчёта по лабораторной работе. При защите лабораторной работы студенту разрешается пользоваться конспектом теоретической части его отчёта. После защиты лабораторной работы преподаватель ставит на титульном листе отчёта свою подпись и дату. Только после этого лабораторная работа считается полностью выполненной, и студент может приступать к выполнению следующей.

6. Студент обязан после выполнения всех лабораторных работ сброшюровать все лабораторные работы, сделав для них общий титульный лист, аналогично представленному выше, с названием: Отчёт по лабораторным работам. Дисциплина «Информатика». На титульном листе преподаватель должен сделать запись о допуске студента к экзамену. В таком виде студент должен представить отчёт лектору на экзамене. В противном случае студент к экзамену не допускается.

ЛАБОРАТОРНАЯ РАБОТА № 1
Типы данных, переменные и константы. Выражения в Visual Basic for Application(VBA). Функции ввода и вывода данных.

1.Краткие теоретические сведения

Обзор типов данных VBA

Обрабатываемые данные делят на числа, даты, текст и другие типы. ***Тип данных*** (data type) – это термин, относящийся к определенным видам данных, которые VBA сохраняет и которыми может манипулировать. В табл. 1.1 обобщены типы данных и показано, какой объем памяти занимает каждый тип, кратко описаны типы данных и дан диапазон значений, которые данный тип может сохранять.

Таблица 1.1 – Типы данных VBA

Название типа	**Размер в байтах**	**Описание и диапазон значения**
Byte	1	Целые положительные числа от 0 до 255
Integer	2	Целые числа от -32768 до 32767
Long	4	Длинные целые числа от -2147483648 до 2147483647
Single	4	Вещественные числа обычной точности с плавающей точкой. Отрицательные числа: от $-3.402823\cdot10^{38}$ до $-1.401298\cdot10^{-45}$. Положительные числа: от $1.401298\cdot10^{-45}$ до $3.402823\cdot10^{38}$
Double	8	Вещественные числа двойной точности с плавающей точкой. Отрицательные числа: от $-1.79769313486232\cdot10^{308}$ до $-4.94065645841247\cdot10^{-324}$. Положительные числа: от $4.94065645841247\cdot10^{-324}$ до $1.79769313486232\cdot10^{308}$

Название типа	Размер в байтах	Описание и диапазон значения
Currency	8	Числа, имеющие до 15 цифр до десятич–ной точки и 4 цифры после нее (денежные единицы). От -922337203685477.5808 до 922337203685477.5807
Boolean	2	Для хранения логических значений; может содержать только значения True (Истина) или False (Ложно)
Date	8	Для хранения комбинации информации о дате и времени. Диапазон дат может быть от 1 января 100 года до 31 декабря 9999 года. Диапазон времени от 00:00:00 до 23:59:59
String (строка пе-ременной длины)	10 байт длина строки	Используется для хранения текста. Может содержать от 0 символов до (при-близительно) 2 миллиардов символов
String (строка фиксиро–ванной длины)	Длина строки (один байт на один символ)	Используется для хранения текста. Может (содержать от одного до(приблизительно) 65400 символов
Variant	16 байт + 1 байт/символ	Тип Variant может хранить любой другой тип данных. Диапазон для данных типа Variant зависит от фактически сохраняе-мых данных. В случае текста диапазон со-ответствует строковому типу; в случае чи-сел диапазон такой, как у типа Double
Object	4	Используется для доступа к любому объ-екту, распознаваемому VBA. Сохраняет адрес объекта в памяти

VBA имеет шесть различных ***численных типов данных***: **Byte, Integer, Long, Single, Double** и **Currency**. ***Численные типы данных*** используются для хранения (и манипулирования) чисел в различных форматах, в зависимости от конкретного типа.

Логический тип данных может принимать значения True и False называют булевыми (Boolean) значениями. Их название связано с именем математика, разработавшего систему матема–тической логики. Логический тип данных называют также типом **Boolean**. Если вы отображаете тип **Boolean** на экране, VBA автоматически преобразует его в строку, содержащую либо слово True, либо False.

VBA использует тип **Date**-тип, для хранения дат и времени. VBA тип **Date** является типом последовательных дат (***serial Dates***). Последовательные даты сохраняют дату как число дней от заданной стартовой даты. Базовой датой для VBA-типа **Date** является 30 декабря 1899. VBA использует отрицательные числа, для представления дат ранее 30.12.1899 и положительные – для дат после 30.12.1899. Число 0 представляет саму дату 30.12.1899. По этой схеме 1 января 1900 записывается числом 2 (1.1.1990 – это 2 дня после 30.12.1899), однако число –2 является датой 28.12.1899 (два дня до 30.12.1899). Чтобы убедиться в этом, напишите простую процедуру.

Пример 1. Процедура, выводящая сообщение даты.

Код процедуры DateTest

```
Sub DateTest()
Dim d As Date:  d = 1: MsgBox d
End Sub
```

Можно записать несколько операторов в одной строке, отделяя каждый из них « **:** » (двоеточием). В результате работы этой процедуры на экран будет выдана базовая дата (Рис. 1.1).

Рис 1.1

В значении последовательной даты целая часть (цифры слева от десятичного знака) – это общее число дней от базовой даты. Последовательная дата может иметь цифры справа от десятичного знака; эти цифры обозначают время дня как часть дня. Один час – это 1/24 дня (приблизительно 0,0416). Аналогично, одна минута – это 1/1440 дня, а секунда – 1/86400 дня. Вы можете вычитать одну дату из другой, добавлять к дате или вычитать числа для изменения ее значения.

Текстовые данные, называются ***строками*** (***strings***). ***Строки*** в VBA сохраняются с использованием типа данных **String**. Строка может содержать текстовые символы любых типов: буквы алфавита, цифры, знаки пунктуации или различные символы. Существуют две категории строк: ***строки переменной длины***, размер которых растет или уменьшается, и ***строки фиксированной длины***, размер которых всегда остается одним и тем же. Все строки в VBA являются ***строками переменной длины***, если только вы не задаете фиксированную длину. Большинство данных ввода пользователей (в диалоговых окнах, ячейках рабочих листов) – это строковые данные. Кроме того, поскольку вы можете отображать на экране только текст, все другие типы данных должны быть преобразованы в строковые данные. Многие встроенные процедуры VBA (подобные Msgbox) используют строковые данные во всех или в некоторых своих аргументах.

Тип данных **Variant** – это особый тип данных, который может сохранять любые типы, приведенные в табл. 1.1, за исключением типа **Object**. VBA использует тип **Variant** для всех переменных, если только, вы не объявляете явно тип переменной.

Константы

Константа (constant) – это значение в программе VBA, которое не меняется. Константы, "Здравствуй, Петров!" и "Моя первая программа" называют литеральными константами (literal constants), потому что литеральное значение записывается непосредственно в код.

В коде VBA можно также писать ***литеральные численные константы*** и ***даты***; примеры численных литеральных констант включают числа 25, 3.14. Примеры литеральных констант-дат включают даты: #12/31/96#, или #октябрь 28, 1997#.

VBA позволяет создавать ***именованные константы*** (***named constants***). ***Именованная константа***, подобно переменной, имеет заданное ей имя; это имя представляет конкретное неизменяемое значение. Синтаксис для объявления именованных констант имеет вид:

Const namel = valuel [operator name2…] _

[,name3 = value3 [operator name4] …]

nameN представляет любой допустимый идентификатор, valueN – любое значение данных: численное, строковое или дата, а operator – арифметическая или операция сравнения между двумя именами ранее описанных констант. Следующие строки показывают несколько объявлений именованных констант:

```
Const Pi = 3.14, text = "Здравствуй, Петров!"
Const Pi2 = 2*Pi        ' Задание константного выражения
```

Область действия констант: можно объявлять именованные константы в процедурах или в области объявлений в начале модуля. Константа, объявляемая в процедуре, имеет область действия процедурного уровня, а константа, объявляемая в области объявлений модуля, – область действия модульного уровня. Для именованных констант выполняются те же правила области действия, что и для переменных.

Написание литеральных констант

При записи ***литеральных строковых констант*** в коде VBA соблюдайте следующие правила:

– строковые константы должны быть заключены в двойные кавычки (“…”);

– пустая строковая константа (называемая нулевой строкой – null string или empty string) обозначается двумя двойными кавычками, между которыми ничего нет ("");

– строковая константа должна вся находиться на одной и той же строке.

При записи литеральных численных констант в коде VBA следует соблюдать следующие правила:

– численные константы должны состоять только из числовых символов от 0 до 9;

– численная константа может начинаться со знака « – » и может содержать десятичную точку « **.** »;

– можно использовать экспоненциальное представление для численных констант например 3,0E+07.

VBA распознает константы даты в любом из нескольких различных форматов; необходимо помещать все константы даты между знаками фунта (#).

Следующие строки показывают некоторые из форматов констант дат, которые распознает VBA:
#2-5-97 21:17:34# #February 5, 1997 9:17:34pm# #Mar-31-97#
#15 April 1997#.

Для типа Boolean существует только две константы: True и False.

Задание типа константы

Когда вы объявляете ***именованную константу*** или используете ***литеральную константу***, VBA «считает», что значение, представленное этой константой, имеет тип данных, наиболее согласующийся с выражением, присвоенным константе.

В VBA можно задать тип константы. Общий синтаксис для объявления типизированной константы следующий:

Const name **As type** = value [, name **As type** = value]

name – это любое допустимое имя константы, **type** – имя любого из типов данных VBA и value – значение, которое вы присваиваете константе.

Следующая строка иллюстрирует правильное объявление константы с определенным типом:

Const Pi **As Double** = 3.14

Внутренние константы

VBA представляет несколько ***внутренних констант*** (***intrinsic constants***), называемых также ***предопределенными константами*** (***predefined constants***). ***Внутренняя константа*** – это ***именованная константа***, которая была определена разработчиками VBA.

Внутренние константы, определяемые VBA, все начинаются с букв vb для указания того, что они определяются языком Visual

Basic for Applications (или Visual Basic). Например, константы, vbOKCancel (константа отображает кнопки OK и Cancel для функции MsgBox), vbLf (символ смещения на одну строку) определяются VBA. Внутренние константы Excel начинаются с букв xl, чтобы было понятно, что они определяются Excel. Благодаря внутренним константам, легче использовать некоторые встроенные процедуры VBA, такие как функции MsgBox и InputBox.

Переменные

Переменная (***variable***) – это имя, которое программист дает области компьютерной памяти, используемой для хранения данных какого-либо типа. Переменные VBA могут хранить любые типы данных, перечисленные в таблице 1.1.

Идентификатор (***identifier***) – это имя, которое вы даете элементам в создаваемых вами процедурах и модулях, таким как переменные. ***Идентификатор*** (***identifier***) – это имя, которое дается элементам в создаваемых процедурах и модулях, таким как переменные. Термин ***идентификатор*** основывается на том факте, что имена, которые задаются, определяют конкретные участки памяти для имени переменной, имени макроса или процедуры или других элементов программы.

Имена переменных не «чувствительны» к состоянию регистра (not case-sensitive), то есть написание имени переменной прописными или заглавными буквами не имеет значения.

Сохранение значения данных в переменной называется присваиванием переменной (assigning the variable или making an assignment). Присваивание выполняется с помощью ***оператора присваивания***, представляемого знаком равенства (=). Следующая строка является примером присваивания значения переменной:

```
MyVar = 14
```

Этот оператор сохраняет численное значение 14 в ячейке памяти, заданной именем переменной MyVar.

Создание переменной путем ее использования в операторе называется ***неявным объявлением переменной*** (***implicit variable declaration***). Используя имя переменной в операторе, можно неявно указать (объявить) VBA создать эту переменную. Все переменные, которые VBA создает неявным объявлением переменной, имеют тип данных **Variant**. Неявное объявление переменных известно также как объявление переменных «на лету» (on-the-fly).

Неявное объявление переменных удобно, но имеет потенциальные проблемы. Например, когда вы имеете переменную с именем MyVar и сделаете позже ошибку в имени, набирая MVar. В зависимости от того, где появится в вашем коде неправильное имя переменной, VBA может выдать ошибку во время исполнения макроса или просто создать новую переменную. Если VBA создаст новую переменную, могут возникнуть ошибки, причины появления которых очень трудно обнаружить.

По этим и другим причинам VBA предоставляет вам возможность выполнять явное (explicit) объявление переменных.

Для явного объявления переменных используйте VBA оператор Dim со следующим синтаксисом:

Dim name1 [, name2]

nameN – это любой допустимый идентификатор переменной. Все переменные, которые вы создаете с этой формой ключевого слова **Dim**, являются переменными типа **Variant.**

Переменную можно объявлять только один раз в отдельной процедуре или модуле. Так как оператор **Dim** находится перед любыми операторами, фактически использующими переменную, вы

можете помещать его в любом месте в процедуре. В практике программирования хорошим правилом является собирать все явные объявления переменных в одном месте в начале процедуры.

Пример 2. Процедура HelloStudent, использующая явное определение переменной.

Код процедуры HelloStudent

```
Sub HelloStudent()                                          ' 1
Dim HelloMsg  ' переменная для MsgBox                       ‘2
HelloMsg = "Здравствуй, Петров!"                            ‘3
MsgBox HelloMsg, , "Моя Вторая программа"                   ‘4
End Sub                                                     ‘5
```

Оператор **Dim** (в строке 2) объявляет переменную HelloMsg и резервирует для нее определенную область памяти (в данной подпрограмме HelloMsg – это переменная типа **Variant**). Строка 2 включает конечный комментарий, указывающий назначение этой переменной. В строке 3 выполняется присваивание переменной HelloMsg строки "Здравствуй, Петров!". Далее (в строке 4) переменная HelloMsg используется как один из аргументов для MsgBox. Функция MsgBox отображает то же окно сообщения, что и раньше. Хотя MsgBox получает теперь свой первый аргумент из переменной, эта переменная содержит ту же строковую информацию, что была ранее (в программах предыдущей части) записана непосредственно в оператор MsgBox.

Все переменные в VBA, независимо от того, объявляются – ли они неявно или явно, являются переменными типа **Variant**, если только вы не задаете тип переменной в объявляющем ее операторе. Для объявления типизированной переменной и ее типа с помощью оператора **Dim** добавьте ключевое слово **As** после переменной, а

затем введите имя типа данных для этой переменной. Вот общий синтаксис для использования оператора **Dim** при объявлении типизированных переменных:

Dim varnamel [**As typel**] [, varname2 [**As type2**]]

где varnameN представляет любое допустимое имя переменной VBA, а **typeN** – любое из имен типов данных VBA.

После объявления типизированной переменной, независимо от того, объявляется ли эта переменная явно или неявно, и как задается тип, эта переменная сохраняет тот же самый тип столько времени, сколько она существует. Нельзя повторно объявить переменную или переопределить ее тип.

При неявном объявлении можно также задавать тип переменной, добавляя специальный символ, называемый символом определения типа (type-definition character), в конец имени переменной. В табл. 1.2 приводятся символы определения типа VBA и типы, которые они обозначают.

Таблица 1.2 – Символы определения типа

Символ определения	**Тип**	**Символ определения**	**Тип**
!	Single	&	Long
@	Currency	%	Integer
#	Double	$	String

Символы определения типа могут находиться только в конце имени переменной.

Несмотря на то, что следует знать, что такое символы определения типа и как они используются, необходимость в их применении бывает редкой — использование оператора **Dim** с ключевым

словом **As** намного легче и понятнее. Большинство программистов, работающих на VBA, не используют символы определения типа.

Пример 3. Процедура использует определение типизированной переменной.

Код процедуры HelloSudent

```
Sub HelloStudent()                          '1
Dim HelloMsg As String                      '2
HelloMsg = "Здравствуй, Петров!"            '3
Title$ = "Моя третья программа"             '4
MsgBox HelloMsg, , Title$                   '5
End Sub                                     '6
```

Строка 1 содержит объявление процедуры. В строке 2 оператор **Dim** явно объявляет переменную HelloMsg. Поскольку оператор **Dim** включает ключевое слово **As** и имя типа **String**, переменная HelloMsg имеет тип **String**. Строка 3 присваивает текст сообщения строковой переменной HelloMsg. Строка 4 неявно объявляет переменную Title$ и в то же время присваивает переменной текст заголовка окна сообщения. Поскольку имя переменной Title$ имеет на конце символ определения типа для строки, эта переменная также имеет тип **String**. Наконец, строка 5 использует функцию MsgBox для отображения окна сообщения; в этом операторе и текст сообщения, и строка заголовка окна являются переменными: HelloMsg и Title$, соответственно.

После добавления символа определения типа к переменной необходимо включать символ определения типа каждый раз, когда вы используете имя переменной.

Независимо от того объявляются ли переменные типа **String** с помощью оператора **Dim** или добавлением символа определения

типа $, создаваемые вами строковые переменные по умолчанию являются строками переменной длины. Строковые переменные переменной длины изменяют длину, в зависимости от длины строки, сохраняемой переменной. Иногда может понадобиться использовать строку фиксированной длины (fixed-length). Строки фиксированной длины всегда имеют одну и ту же длину. Следующая строка показывает общий синтаксис для создания строки фиксированной длины:

Dim varname **As String** * N

varname – это любое допустимое имя переменной, а N – это любое число от 1 до 65400 символов.

Требование явного объявления переменных

Хотя неявное объявление переменных (объявление переменных просто их использованием) удобно, с ним связаны некоторые проблемы. Когда переменные объявляются неявно, существует риск нечаянно создать новую переменную, когда на самом деле необходимо использовать уже существующую, или использовать существующую переменную, когда пользователь намеревается создать новую. Обе эти ситуации приводят к ошибкам в коде, которые очень трудно отслеживать.

Чтобы легче было в любое время обнаруживать ошибки, связанные с неявным объявлением переменных, VBA предоставляет команду **Option Explicit**. При использовании **Option Explicit** VBA требует объявления всех переменных (с помощью оператора **Dim**) перед их использованием в модуле.

Чтобы установить режим, при котором VBA требует явного объявления для всех переменных в модуле, необходимо добавить команду **Option Explicit** в область объявлений модуля, то есть в начало модуля перед любыми объявлениями переменных или проце-

дур. Такие команды, как **Option Explicit**, называются ***директивами компилятора*** (***compiler directives***).

Команда **Option Explicit** действует только на модуль, в котором она появляется. Если проект, содержащий этот модуль, содержит также другие модули, внутри них не действует команда **Option Explicit**. Необходимо включать команду **Option Explicit** в каждый модуль, для которого требуются явные объявления переменных.

Отображение окон сообщений. Ввод данных пользователя

Получение данных от пользователя, сохранение их в переменной и отображение результатов действий, выполненных над ними, являются основными элементами, необходимыми для написания интерактивных процедур. Интерактивная (interactive) процедура – это процедура, обменивающаяся информацией с пользователем, то есть процедура, которая взаимодействует с пользователем отображая сообщения и получая ввод.

Функция MsgBox отображающая окно с сообщением имеет следующий синтаксис:

MsgBox (Prompt [, Buttons] [, Title] [, HelpFile, Context])

Аргумент Prompt представляет любое строковое значение (литерал, константу или переменную). MsgBox отображает эту строку в диалоговом окне; необходимо всегда предоставлять аргумент Prompt, поскольку это – обязательный аргумент (required argument). Аргумент Buttons (необязательный аргумент), является численным выражением, определяет отображаемые в диалоговом окне кнопки и сообщений. Аргумент Title представляет любое строковое значение (литерал, константу или переменную). MsgBox отображает текст этой строки в строке заголовка диалогового окна. Если опустить аргумент Title, VBA отображает в строке заголовка диалогово-

го окна MsgBox слово "Microsoft Excel". Аргумент HelpFile – файл справки, Context – раздел в справочном файле. Текст сообщения можно заключать в скобки, но скобки необязательны, когда функция MsgBox используется как оператор.

Данные, вводимые пользователем, называются входными данными (input). Чтобы получить входные данные от пользователя процедуры, используйте функцию InputBox. Функция (**Function**) – это особый тип процедуры VBA, возвращающей значение.

Функция InputBox отображает диалоговое окно, содержащее текст, который запрашивает пользователя ввести некоторое значение, и текстовое окно для ввода этого значения. Диалоговое окно, отображаемое InputBox, содержит также командные кнопки ОК и Cancel.

Функция InputBox имеет следующий синтаксис:

stringvar = InputBox (Prompt [, Title] [, Default] [, XPos] [, YPos] [, HelpFile, Context])

Здесь stringvar представляет любую переменную, которая может сохранять строку (либо переменную типа **String**, либо – **Variant**). Аргумент Prompt представляет любое строковое значение (литерал, константу или переменную). InputBox отображает эту строку как запрос в диалоговом окне; необходимо всегда предоставлять аргумент Prompt, поскольку это – обязательный аргумент; все остальные – необязательные. Аргумент Title является вторым аргументом для InputBox. Title представляет любое строковое значение (литерал, константу или переменную). InputBox отображает текст этой строки в строке заголовка диалогового окна. Если опустить аргумент Title, VBA отображает в строке заголовка диалогового окна InputBox слово "Microsoft Excel".

Аргумент Default – строковое выражение, отображаемое в поле ввода как используемое по умолчанию, если пользователь не введет другую строку; если это аргумент опущен, поле ввода изображается пустым. Аргументы XPos и YPos могут быть любыми численными выражениями. Эти аргументы позволяют указать, где в активном окне появляется окно ввода, и являются координатами верхнего левого угла диалогового окна: XPos – горизонтальное расстояние от левого края окна; YPos – это вертикальное расстояние от верхнего края окна. Оба расстояния измеряются в твипах (twips); один твип равен 1/20 точки (точка – это измерение шрифта печати). Поскольку точка составляет 1/72 часть дюйма, то один твип приблизительно равен 0,0007 дюйма. Последние два необязательных аргумента для функции InputBox – это HelpFile и Context. Они имеют то же назначение, что и подобные аргументы функции MsgBox.

Использование именованных аргументов функций

Пропуск или перестановка аргументов в списке аргументов функции могут привести к ошибкам несовпадения типов. Ошибка (что еще хуже) может быть не обнаружена. Чтобы предотвратить ошибки программирования и сделать более легким использование функций, имеющих необязательные аргументы, VBA предоставляет альтернативу перечислению значений в списке аргументов в определенном порядке. Можно также передавать значения аргументов функции, используя ***именованные аргументы*** (***named arguments***) функций. Следующие строки показывают два оператора MsgBox, которые имеют один и тот же результат; первый оператор использует обычный метод перечисления аргументов, а второй – метод именованных аргументов:

```
MsgBox AnyMsg, , AnyTitle
MsgBox Prompt:=AnyMsg, Title:=AnyTitle
```

Символ, который присваивает значение именованному аргументу «:=», не является в точности тем же обычным оператором присваивания «=». Если опустить двоеточие «:» во время присваивания значения именованному аргументу, VBA не обязательно обнаружит ошибку синтаксиса, но не может интерпретировать этот оператор правильно. Когда VBA выполняет этот оператор, он отображает одну из нескольких возможных ошибок во время исполнения, наиболее часто – ошибку несовпадения типов.

Нельзя смешивать именованные аргументы с обычным списком аргументов в одном и том же вызове функции. Необходимо использовать либо именованные аргументы, либо список обычных аргументов для каждого отдельного вызова функции.

Пример 4. Процедура, вычисляющая площадь круга.

Код процедуры Prog4 (ввод исходных данных при помощи функции InputBox отображается в диалоговом окне (Рис.1.2))

```
Const Pi As Single =3.14  ' число π                                    '1
Dim CircleArea As Single ' площадь круга                               '2
Sub Prog4 ()                                                           '3
Const BoxTitle = "Площадь круга"                                       '4
Dim Radius As Single, Temp As String                                   '5
Temp = InputBox("Введите радиус " & Chr(13) & "круга", _               '6
BoxTitle)                                                              '7
Radius = CSng(Temp)                                                    ‘8
CircleArea = Pi * Radius * Radius                                      '9
MsgBox "Результат " & CircleArea, vbInformation +  _                   '10
vbOKCancel, BoxTitle                                                   '11
End Sub                                                                '12
```

Строки 1 и 2 программы 4 объявляют константу Pi и переменную CircleArea модульного уровня (после знака апостроф расположены пояснения – комментарии). Строка 3 содержит объявление процедуры. Строка 4 объявляет константу процедурного уровня BoxTitle; эта константа имеет только локальный доступ в процедуре. В строках 6, 7 оператор вызывает функцию InputBox. Она отображает свой первый аргумент как текст в диалоговом окне (Рис.1.2), запрашивая у пользователя ввести значения какого-либо типа. В этом операторе InputBox отображает текст "Введите радиус круга" (функция Chr(13) – символ перехода на новую строку), чтобы сообщить пользователям процедуры, какое значение они должны ввести. InputBox использует второй аргумент, представленный константой BoxTitle, в качестве заголовка диалогового окна.

Рис. 1.2

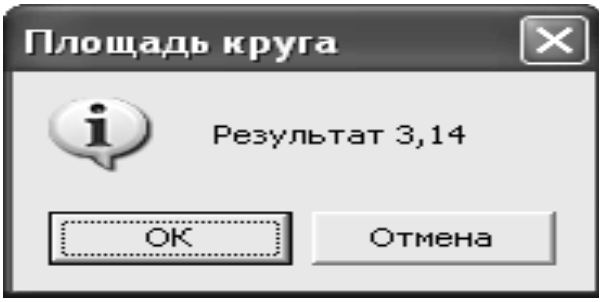


Рис. 1.3

При выполнении оператора InputBox пользователь вводит число в текстовое окно и выбирает командную кнопку OK или Cancel, чтобы закрыть диалоговое окно, как и любое другое окно Windows. Всякий раз, когда вы вызываете какую-либо функцию, необходимо как-то использовать значение, возвращаемое функциией. Значение, возвращаемое функцией, называется результатом функции (function result). Как показано в строках 6, 7 (строка 7 продолжение строки 6,

в конце строки 6 на это указывает пробел с подчеркиванием) функция InputBox вводит значение, которое присваивается переменной Temp. Результат функции InputBox всегда является строкой (вот почему переменная Temp была объявлена как **String**). Поскольку переменная Temp была объявлена явно как **String**, значение строки должно быть преобразовано в численное значение перед тем, как можно будет его использовать в математических вычислениях. Строка 8 программы 4 выполняет именно это, используя, встроенную функцию CSng для преобразования пользовательских данных ввода в число с типом **Single.** В строках 10,11 используя функцию MsgBox, в диалоговое окно (Рис.1.2) выводится площадь круга, переменная CircleArea, вычисленная операторе 9.

Выражения в VBA

Выражение (expression) – это комбинация знаков операций и операндов. Каждое выражение вычисляется до конкретного значения. Выражения могут состоять из операндов: констант (литеральных или именованных), переменных (любого типа), массивов, элементов массива, функций, знаков операций, а также скобок.

Результат вычисления выражения имеют результатом одно значение конкретного типа данных. Выражения могут также иметь результатом одно из специальных значений Empty (неинициализированную переменную типа **Variant** или результат выражения, содержащий неинициализированную переменную типа **Variant**) или Null (Null представляет выражение, содержащее неверные данные). Когда используется какой-либо знак в выражении, элементы данных (переменные или константы), над которыми производится действие, называются ***операндами*** (***operands***); большинству операций требуется два операнда.

Совместимость типов данных. Автоматическое преобразование данных

Не все типы данных совместимы друг с другом, и нельзя использовать несовместимые типы данных в одном и том же выражении. Например, не имеет смысла арифметическое сложение строки с числом, так как такое выражение не является значащим и VBA не может его оценить.

Многие типы данных совместимы друг с другом. Например, вы можете объединять различные численные типы данных в одном и том же выражении; VBA автоматически выполняет необходимые преобразования типа различных численных типов. Очень важно контролировать и знать тип выражения, потому что если выражения содержат несовместимые типы, VBA выдает ошибку времени исполнения – ошибку несовпадения типов (type-mismatch). При обработке выражения, содержащего различные типы данных VBA сначала «пытается» устранить любое различие типов, преобразуя значения в выражении в совместимые типы данных. Если устранить какие-либо различия преобразованием типов не удается, отображается ошибка времени исполнения и процедура прекращает выполняться. Например, в выражении 25 & "Информатика" VBA всегда выполняет строковую конкатенацию (соединяет две строки), независимо от типов переменных; результатом является тип String; это выражение никогда не вызывает ошибки несовпадения типов.

VBA обычно преобразует все численные типы данных в выражении в тип наибольшей точности, а затем дает этот тип результату выражения. Например, если выражение содержит численные значения с типами **Integer** и **Single**, результат выражения является типом **Single** – тип наибольшей точности в этом выражении. Если вы присваиваете результат численного выражения переменной с наи-

меньшей точностью, чем фактический тип результата выражения, VBA округляет результат выражения до тех пор, пока его точность не совпадет с ожидаемым типом. Например, если вы присваиваете численное выражение, имеющее результатом число типа **Double**, переменной типа **Integer**, VBA округляет число двойной точности до типа **Integer**.

При преобразовании числа в строку VBA создает строку, содержащую все цифры этого числа и десятичный знак (если число имеет его). Число 3413.72 (точка используется для записи числа в коде), например, преобразуется в строку "3413,72". Если число очень большое или очень маленькое, VBA может создать строковое представление числа в экспоненциальной записи; например, число 0.0000000004927 преобразуется в строку "4,927E–11".

VBA может преобразовывать строку в число, если только эта строка содержит символьное представление числа в десятичном формате или экспоненциальном. Строки "988,6", "812", "-186,7", "1,3E10" представляют числа, и VBA может преобразовать их в числа. Строки "1.045", "$74.550" и "С добрым утром!" не могут быть преобразованы в числа.

Когда VBA преобразует значения типа **Boolean** в числа, значение True преобразуется в 1, а значение False – в 0. Когда VBA преобразует число в тип **Boolean**, нуль преобразуется в False, а любое другое значение преобразуется в True. Когда VBA преобразует значения типа **Boolean** в строки, VBA использует строку "True" для True и "False" – для False.

Оператор присваивания (=)

Этот оператор используется для присваивания результата выражения переменной. Синтаксис ***оператора присваивания*** следующий***:***

varname = expression

переменная varname - любая переменная, а expression - любое выражение.

При выполнении оператора присваивания VBA сначала вычисляет выражение справа от ***оператора присваивания*** « = », а затем сохраняет результат выражения в переменной, имя которой находится слева от оператора присваивания.

Когда присваивается результат выражения переменной с определенным типом данных, этот результат может иметь тип данных, совместимый с типом переменной, получающей новое значение. Во многих случаях, VBA может преобразовывать тип данных результата выражения в тип, совместимый с типом переменной, принимающей новое значение, если результат выражения и переменная еще не имеют совместимых типов. Переменным типа **Variant** может быть присвоен любой тип данных.

Арифметические операции

VBA может выполнять все обычные арифметические операции (реализуемые посредством арифметических выражений): сложение, вычитание, умножение и деление, а также возведение чисел в указанную степень и предоставляет дополнительные особые математические операции для целочисленного деления и деления по модулю (табл. 1.3).

Таблица 1.3. - Обозначения, используемые в арифметических выражениях (Ni - это допустимое численное выражение VBA)

Знак	Синтаксис	Имя/Описание
+	N1 + N2	*Сложение*. Прибавляет N1 к N2
–	N1 – N2	*Вычитание.* Вычитает N2 из N1
*	N1 * N2	*Умножение.* Умножает N1 на N2
/	N1 / N2	*Деление.* Делит N1 на N2.
\	Nl \ N2	*Целочисленное деление.* Делит N1 на N2, отбрасывая любую дробную часть так, чтобы резуль-
	N1 Mod N2	*Деление по модулю.* Делит N1 на N2, возвращая только остаток операции деления.
^	N1 ^ N2	*Возведение в степень*. Возводит N1 в степень N2.

Оба операнда должны быть численными выражениями или строками, которые VBA может преобразовать в число.

Операции сравнения

Операции сравнения также называют операциями отношения (***relational operators***). Результатом любой ***операции сравнения*** является значение типа **Boolean**. Операции сравнения используются для сравнения значений любого сходного типа (табл. 1.4).

Таблица 1.4 –операции сравнения (E в этой таблице - любое действительное выражение VBA)

Операция (Оператор)	Синтаксис	Наименование/описание
=	El = E2	*Равенство.* **True**, если El равно E2, иначе – **False**
<	El < E2	*Меньше, чем.* **True**, если El меньше, чем E2, иначе – **False**
<=	El < E2	*Меньше, чем или равно.* **True**, если E1 меньше или равно E2, иначе – **False**
<>	<>	*Не равно.* **True**, если E1 не равно E2, иначе – **False**
>	>	*Больше, чем.* **True**, если E1 больше, чем E2, иначе – **False**
>=	>=	*Больше, чем или равно.* **True**, если E1 больше или равно E2, иначе – **False**

Конкатенация строк

Присоединение одной строки к другой называется ***конкатенацией*** (***concatenation***) строк. Знак « & » можно использовать только для конкатенации строк. Общий синтаксис знака & такой:

Operand1 & Operand2 [& Operand3…]

Operand1 и Operand2 – любые допустимые строковые или численные выражения. VBA преобразует числа в строки перед выполнением операции конкатенации. Тип данных результата конкатенации строк – это всегда тип **String**.

Логические операторы

Чаще всего ***логические операторы*** VBA используются для объединения результатов отдельных выражений сравнения, чтобы создать сложные критерии для принятия решений в процедуре, или для создания условий, при которых группа операторов должна повторяться (табл**.** 1.5).

Таблица 1.5 – Логические операторы (E в этой таблице представляет собой любое допустимое выражение с результатом типа Boolean, такое как операция сравнения)

Оператор	**Синтаксис**	**Имя/Описание**
And	El And E2	*Конъюнкция*. **True**, если оба E1 и E2 имеют значение **True**, иначе – **False**
Or	El Or E2	*Дизъюнкция*. **True**, если одно выражение или оба (E1 и E2) являются равными **True**; иначе – **False**
Not	Not El	*Отрицание*. **True**, если E1 имеет значение **False**; **False**, если El является равным **True**
Xor	El Xor E2	*Исключение*. **True**, если E1 и E2 имеют разные значения; иначе – **False**
Eqv	El Eqv E2	*Эквивалентность*. **True**,если E1 имеет то же самое значение, что и E2; иначе – **False**
Imp	El Imp E2	*Импликация*. **False**, когда E1 является равным **True** и E2 равно **False**; иначе **True**.

Приоритеты выполнения операций при вычислении сложных выражений

Сложное (составное) выражение (complex expression) – это любое выражение, образованное из двух или более выражений. Многие из записываемых вами выражений – это сложные выражения, особенно, если они определяют управление последовательностью выполнения кода в процедурах или представляют различные математические формулы (табл. 1.6).

Таблица 1.6 – Иерархия операторов/операций от наивысшего до самого низкого приоритета.

Оператор знак	Комментарии
^	Возведение в степень, наивысший
–	Унарный —
*, /	Умножение и деление имеют равные приоритеты; они вычисляются по мере появления в выражении
\	Деление нацело
Mod	Остаток от деления нацело
+, –	Сложение и вычитание имеют равный приоритет; они вычисляются по мере появления в выражении
&	Всякая конкатенация строк выполняется после любых арифметических операций в выражении и перед любыми операциями сравнения или логическими
<, <=, >, >=, Like, =, <>, Is	Все операторы сравнения имеют равные приоритеты и вычисляются по мере появления в выражении слева направо. Используйте круглые скобки для груп-
Логические операторы	Not And Or Xor Eqv Imp

Использование функций Visual Basic

Функция (**Function**) – это встроенная формула, выполняющая действия над выражениями и генерирующая значение. Функция всегда возвращает значение, которое VBA вставляет в программу в том месте, где появляется имя функции. Функции VBA делятся на несколько групп в зависимости от типа операции или вычисления,

которое они выполняют. Вы уже пользовались функциями: InputBox и MsgBox - для ввода вывода данных. Не путайте термины функция и процедура. Обычно процедура выполняет определенную задачу (или группу задач) подобно тому, как определенная команда меню в Excel, Word или другом приложении выполняет конкретную задачу. Функция оперирует одним или более значениями и возвращает некоторое результирующее значение (как формула в ячейке рабочего листа Excel). Чтобы использовать функцию, просто вводите имя функции в оператор VBA вместе с любыми аргументами, которые требуются для этой функции, в том месте в операторе, где вам необходимо использовать результат функции. Размещение имени функции в операторе называют вызовом (calling) функции.

Встроенные функции VBA делятся на несколько категорий (математические, преобразования данных, даты и времени, строковые и прочие).

Математические функции

VBA предоставляет стандартный набор математических функций.

Таблица 1.7 – Математические функции (N означает любое численное выражение)

Функции (аргументы)	**Возвращает/действие**
Abs(N)	Возвращает абсолютное значение N
Cos(N)	Косинус угла N, где N – это угол, измеренный в радианах
Sin(N)	Возвращает синус угла; N – это угол, измеренный в радианах
Tan(N)	Возвращает тангенс угла; N – угол в радианах
Atn(N)	Возвращает арктангенс N как угол в радианах

Функции (аргументы)	Возвращает/действие
Exp(N)	Возвращает константу *e*, возведенную в степень N *e* – это основание натуральных логарифмов и она (приблизительно) равна 2,718282
Fix(N)	Возвращает целую часть N. Fix не округляет число, а отбрасывает любую дробную часть. Если N является отрицательным, Fix возвращает ближайшее отрицательное целое большее, чем или равное N
Int(N)	Возвращает целую часть N. Int не округляет число, а отбрасывает любую дробную часть. Если N является отрицательным, Int возвращает ближайшее отрицательное целое меньшее, чем или равное
Log(N)	N Возвращает натуральный логарифм N
Rnd(N)	Возвращает случайное число; аргумент является необязательным. Используйте функцию Rnd только после инициализации VBA-генератора случайных чисел оператором Randomize
Sgn(N)	Возвращает знак числа: –1, если N – отрицательное; 1, если N –положительное; 0, если N равно 0
Sqr(N)	Возвращает корень квадратный из N. VBA отображает ошибку исполнения, если N – отрицательное

Функции преобразования данных

VBA содержит функции для преобразования одного типа данных в другой (табл. 1.8). Эти функции используются для устранения ошибок несовпадения типов, и обеспечения явного контроля за типами данных в выражениях.

Таблица 1.8 – Функции преобразования данных (N – это любое численное, S – любое строковое, а E – выражение любого типа)

Функция (аргументы)	**Возвращает/действие**
Asc(S)	Возвращает число кода символа, соответствующее первой букве строки S. Буква "A", например, имеет код символа 65
Chr(N)	Возвращает строку из одного символа, соответствующего коду символа N, который должен быть числом между 0 и 255, включительно. Код символа 65, например, возвращает букву "A" (Chr(13) – символ возврата каретки, Chr(10) – символ смещения на одну строку)
Format(E, S)	Возвращает строку, содержащую значение, представленное выражением E, в формате в соответствии с инструкциями, содержащимися в S
Hex(N)	Возвращает строку, содержащую шестнадцатиричное представление N
Oct(N)	Возвращает строку, содержащую восьмиричное представление N
Str(N)	Возвращает строку, эквивалентную численному выражению N
Val(S)	Возвращает численное значение, соответствующее числу, представленному строкой S, которая должна содержать только цифры и одну десятичную точку, иначе VBA не может преобразовать ее в число. Если VBA не может преобразовать строку в S, то функция
CBool(N)	Возвращает Boolean-эквивалент численного выражения N
CCur(E)	Возвращает численное значение типа Currency; E – любое допустимое численное или строковое выражение, которое может быть преобразовано в число
CStr(E)	Возвращает значение типа String; E – любое допустимое численное или строковое выражение
CVar(E)	Возвращает значение типа Variant; E – любое допустимое численное или строковое выражение

2. Практическая часть.

Задание 1. Напишите процедуру, выводящую сообщение даты вашего рождения (пример 1).

Задание 2. Модифицируйте процедуру HelloStudent (пример 2), используя явное определение переменной и свою Фамилию.

Задание 3. Составить процедуру для выполнения расчетов по формулам своего варианта из таблицы 1.9 (**пример 4**):

- для первой функции значения задавать в программе с помощью оператора присваивания;
- для второй функции значения задавать в диалоге с использованием функции InputBox.

Таблица 1.9

Вариант	Функция	Значения аргумента
1	$Y = 2^x + 5x - 3$ $F=(x+1)^2 + 3(x+1)$	-2,1; 1.5; 3.25 -2; -1; 0
2	$Y = tg\ (0{,}58x + 0{,}1) - x^2$ $F = (x+1)^3 + 2(x+1)$	-2,1; 1,5; 3,25 -2; -1; 0
3	$Z = (\log_2 (x+2))(x-1) - 1$ $F=2(x+3)^3 + 3(x+3)$	-1,1; 0; 2,32 -2 ;-1 ;0 2
4	$Q=(x-2)^2 2^x-1$ $F=x^2(x^2+1)$	-3,2; 2,1; 3,45 -2 ;-1 ; 0
5	$R=x \log_3(x+1)-2$ $F=4x^2 + 2(x^3+1)^2$	-0,5; 0; 2,34 -2; -1; 0
6	$L=(x-3)^2 \log_{0,5}(x-2) + 1$ $F=3(x+1)^2 + 2(x+1)^3+2$	2,5; 3; 4,15 -2; -1; 0
7	$A=arctg\ x - 1/3x^2$ $F=x^2 (x^2 - 1)$	-1; -0,3; 3,12 1; 0; 4
8	$B=2arctgx-x+3$ $F=x/2+(x/2)^2$	-1; -0,5; 2,25 -2; 0; 2

Ва-ри-ант	Функция	Значения аргумента
9	$C=1+x+x^2/2!+x^3/3!$ $F=(x+1)/5+(x+1)^2$	-1,1; 0; 2,5 -4; -1; 4
10	$D=8{,}36\times10^8+(1/(1-x)+1)\cos^2 x$ $F=x/3+(x/3)^2+1$	-4,2; 0; 4,15 -6; 0; 6
11	$G=\lg x^2 - 7/(2x+6)^2$ $F=2x^3 - 3x^2 + 3$	-4,3; -1; 2,4 -1; 0; 2
12	$K=\operatorname{ctg}x - x^2/2$ $F=x^3 + 3x^2 - 10$	-2,2; 0,1; 2,8 -1; 0; 1,5
13	$M=\pi(1+e^x)^{1/x}$ $F=2x^3 + 9x^2 - 21$	-2,1; 1,5; 2,3 -2; 0; 3
14	$N=\sqrt[5]{1+\cos^2 x}$ $F=x^3 + 3x^2 - 2$	0,23; 1,2; 2,4 -3; 0; 3
15	$Q=\frac{e^{-x}+\sqrt{x}}{1+\sqrt[4]{x^3}}$ $F=(x+1)/3+2(x+1)^2$	0,3; 0,5;2,8 -4; -1; 1
16	$U=\operatorname{tg}^2(0{,}4x+0{,}4) - x^3$ $F=x^3 + 4x - 6$	-1,5; 0; 1,5 -3; 0; 3
17	$W=\operatorname{ctg}x - 10e^{-x}$ $F=x^3 + 2(x-1) + 4$	-2,1; 0,1; 2,7 -1; 0; 2
18	$Y=x^2 2^x - 4/3$ $F=(x^2+1)/2+27/x^2$	-1,1; 0; 1,2 -3; -1; 3
19	$A=\sin(x-0{,}5) - x^2 + 8/10$ $F=(x-2)^3 + (x+1)^2 + 5$	-2,3; 0; 0;5 -1; 0; 2
20	$B=\operatorname{tg}^3 x - x + 1/2$ $F=(x+1)^3 + x^2 - 3$	-1; 0; 2,19 -1; 0; 2
21	$C=\sqrt{2\pi}\,2^{1/x}$ $F=(x+1)^2 + (x-2)^2 + 5$	-2,35; 1,2; 2,73 -1; 2; 3
22	$D=\sqrt[5]{x^3+x^2+4/3}$ $F=(x+4)^2 + x/2$	-2,3; 1,2; 3,75 -5; -4; 1
23	$E=5^x + 6\ln(x+1) - 3$ $F=x^4 - x^3 + 3x$	-0,75; 1,2; 2,4 -2 ; 0; 2
Ва-	Функция	Значения аргумента

ри-ант		
24	$G=2e^x + 3x + 1/3$ $F=x^4 - x^2 - 4/3$	-0,83; 0; 1,23 -2; 0; 4
25	$I=2x^2 + 0,5^x - 3/5$ $F=3x^4 + 8x^3 - 10/8$	-0,12; 0; 1,15 -1; 0; 1
26	$Y=3^x + 5x^2 - 2/3$ $F=x^4 - 18x^2 + 6$	-2,3; 0; 1,15 -2; 0; 2
27	$Q=2\lg\lvert x\rvert - (x/2)^2 + 1/2$ $F=x^3 -(x-2)^2 + 3$	-2,1; 0,85; 3,12 0; 1; 3
28	$R=0,5^x + 1/3 - (x+1)^2$ $F=x^2 + (x-2)^2/2 + 5$	-2,2; 0; 1,23 -1; 0; 3
29	$U=\sqrt[3]{x \lg^2 \lvert x\rvert}$ $F=x^3 - 3x^2 + 6x - 5/3$	-1,5; -0,5; 2,3 -3; 0; 3 2
30	$W=9,27\cdot10^5 - 5,2\cdot10^3 \cos^3 x$ $F=x^4 - 12x^2 - 2x -3/2$	-2,3; 0; 1,8 -2; 0; 2

ЛАБОРАТОРНАЯ РАБОТА №2
Ввод-вывод информации, с использованием файлов. Форматирование значений данных.

1.Краткие теоретические сведения

Работа с файлами

Файл – это поименованная область памяти на внешнем носителе (например, магнитный диск), содержащая некоторые данные или программу.

При открытии файлу ставится в соответствие канал с определенным номером. Каждый открытый файл имеет свой канал. Функция FreeFile [(Range_Number)] возвращает номер свободного канала, который можно использовать для очередного открываемого файла. Параметр Range_Number необязательный, если задан равным 0, то возвращаемый файловый номер находится в диапазоне от 1до 255.

Реализуются три типа доступа к файлам: 1) последовательный (Sequential) – для чтения и записи текстовых файлов, 2) произвольный (Random) – для чтения и записи текста или структурированных двоичных файлов с записями фиксированной длины, 3) двоичный (Binary) – для чтения и записи произвольно структурированных файлов.

Файл последовательного доступа

Для открытия файла с последовательным доступом используется команда

Open *FILE_NAME* **For** *WORK_TYPE* **As** #*FILE_NUMBER*,

где – *FILE_NAME* – имя с расширением и маршрутом (полным путем);

WORK_TYPE:

1) Append: – файл открывается для помещения в него записей, если он уже содержит какие то записи, то новые помещаются в конец файла;

2) Input – файл открывается для чтения из него записей;

3) Output: – файл открывается для помещения в него записей;

FILE_NUMBER – целое число между 1 и 255, предварительно определяемое с помощью функции FreeFile; обращение к файлу из процедуры выполняется под этим номером.

После того, как файл обработан, его закрывают командой

Close #FILE_NUMBER.

Доступ к файлу в процедере возможен между командами **Open** и **Close**.

Для помещения записи в файл используется команда

Print #FILE_NUMBER, VARNAME1 [, VARNAME2]...

Для чтения строк из файла используются следующие команды:

1) чтение одной строки:

Line Input #FILE_NUMBER, str_Varname

2) чтение всего файла в строковую переменную (strText):

strText = Input$ (LOF(FILE_NUMBER), FILE_NUMBER),

где LOF() – функция определения длины файла в байтах;

3) чтение последовательности определенного количества символов:

Input #FILE_NUMBER, ПОЛЕ_ДАННЫХ_1 [, ПОЛЕ_ДАННЫХ_2]...

Функция EOF(#FILE_NUMBER) возвращает логическое значение ИСТИНА (True), если достигнуто окончание файла, и значение ЛОЖЬ (False) – в противном случае.

Файл произвольного доступа

Открытие файла для произвольного доступа осуществляется командой

Open FILE_NAME **For Random** [Acces TYPE_ACCESS] **As** #FILE_NUMBER [Len = ДЛИНА_ЗАПИСИ],

где параметр Acces задает режим доступа к файлу: Read - чтение, Write - запись, Read Write - чтение и запись (без указания параметра также чтение и запись).

Для записи используется команда

Put #FILE_NUMBER, NUMBER_RECORD, VARNAME

для считывания команда

Get #FILE_NUMBER, NUMBER_RECORD, VARNAME

Файл двоичного доступа

Открытие файла для двоичного доступа осуществляется командой

Open FILE_NAME **For Binary** [Acces TYPE_ACCESS] **As** #FILE_NUMBER,

длина записи не указывается, т. к. обмен происходит побайтно; для ввода и вывода используются те же операторы Get и Put, но вместо номера записи указывается номер байта.

Хотя VBA может автоматически преобразовывать любой тип данных в строку для отображения с помощью функции MsgBox или для вставки в рабочий лист Excel, формат данных, который выбирает VBA, может не совпадать с желаемым. При преобразовании числа в строку VBA не добавляет в строку разделитель тысяч, символы доллара или другое числовое форматирование. Кроме того, если число очень большое или очень малое, VBA создает строку,

представляющую это число в экспоненциальном формате. При преобразовании дат VBA всегда использует короткий формат даты и времени, используемый операционной системой компьютера, и всегда отображает и дату, и время.

Для получения почти любого формата дат при преобразовании чисел или дат в строки можно использовать функцию Format; можно даже использовать функцию Format для форматирования строковых данных в соответствии с определенным шаблоном. Можно также создавать пользовательские экранные форматы, если вам необходимо, чтобы данные появлялись в каком-либо особом формате. Синтаксис оператора **Format**:

Format(Expression [, Format[, Firstdayofweek [, Firstweekofyear]]])

Expression – любое допустимое выражение (обязательный); Format – допустимое выражение именованного или определенного пользователем формата (необязательный); Firstdayofweek – константа, которая определяет первый день недели (необязательный); Firstweekofyear – константа, которая определяет первую неделю года (необязательный).

Для аргументов Firstdayofweek и Firstweekofyear в VBA имеются именованные константы, о которых можно узнать из справочной системы VBA в разделе Date Constants.

Чтобы использовать функцию **Format**, можно либо задать предопределенный формат (называемый *именованным форматом* (named format), либо создать образ определенного формата, используя комбинации особой группы символов, называемых *символами заполнителями* (placeholders). Если вам необходимо создавать пользовательские форматы для чисел, дат или времени, нужно создать строку, содержащую символы-заполнители, для задания образа форматирования, который должна будет использовать функ-

ция Format при преобразовании значений в строку (табл. 2.1). Кроме того, в табл. 2.1 используется как пример численное значение 1234,5.

Таблица 2.1 – Символы-заполнители пользовательских форматов

Символ-заполнитель	**Действие**
0	Цифровой символ, отображает цифру, если таковая находится в этой позиции, или 0, если – нет. Можно использовать символ 0 для отображения начальных нулей для целых чисел и конечных нулей в десятичных дробях; **00000.000** отображает 01234,500
#	Цифровой символ, отображает цифру, если таковая находится в этой позиции, иначе — не отображает ничего. Символ-заполнитель # эквивалентен 0, кроме того, что начальные и конечные нули не отображаются; #####.### отображает 1234,5
$	Отображает знак доллара; $###,###.00 отображает $1 234,50
. (точка)	Десятичный символ-заполнитель, отображает десятичную точку в обозначенной позиции в строке символов-заполнителей 0; #.##.## отображает 1234,5
%	Символ процента, умножает значение на 100 и добавляет знак процента в позицию, указанную символами-заполнителями 0; **#0.00%** отображает число 0.12345 как 12,35% (12,345 округляется до 12,35)
, (запятая)	Разделитель тысяч, добавляет запятые как разделители тысяч в строках символов-заполнителей 0 и #; ###,###,###**.00** отображает 1 234,50
E-, e-	Отображает значения в экспоненциальном формате со знаком порядка только для отрицательных значений; #.####**E–00** отображает 1,2345E03; 0,12345 отображается как 1,2345E-01
E+, e+	Отображает значения в экспоненциальном формате со знаком порядка для положительных и отрицательных значений; #.####**E+00** отображает 1,2345E+03

Пример 1. Процедура, использующая для ввода-вывода файлы на диске и форматирование данных.

Код процедуры lab2

```
Sub lab2()
Dim x, y As Single
Open "z:\PETROV\LAB2\dat2.txt" For Input As #1
Open "z:\ PETROV\LAB2\res2.txt" For Output As #2
Print #2, Tab(10); "результаты расчетов"
10 Input #1, x
If EOF(1) Then GoTo 99
y = Sin(x)
Print #2, Tab(10); " x= "; Spc(3); Format(x, "##.0#"); _
Tab(30);  " y= "; Spc(3); Format(y, "#.##0E+")
GoTo 10
99 Print #2, Tab(15); "Студент Петров ИВАН"
Close #1
Close #2
End Sub
```

Пояснения к программе.

Функции Tab(n) – перемещает позицию вывода на n позиций относительно начала. Функция Spc(n) – заполняет пробелами n позиций, относительно текущей позиции.

Чтобы позиционировать позицию вывода сразу после последнего выведенного символа, используется разделитель точка с запятой (;).

Функция EOF (file_number) возвращает значение истина при достижении конца файла, целым допустимым номером file_number.

Файл исходных данных создается в папке лабораторной работы до выполнения программы. Файл результата получается в папке лабораторной работы после выполнения программы.

Файл исходных данных dat2.txt

```
1
2
3
4
```

Файл результатов res2.txt

```
      результаты расчетов
    x=   1,0        y=   8,415E-1
    x=   2,0        y=   9,093E-1
    x=   3,0        y=   1,411E-1
    x=   4,0        y=  -7,568E-1
       Студент Петров ИВАН
```

2. Практическая часть.

Задание 1. Составить процедуру для вычисления и печати значений функции из таблицы 2.2. Вычислить 8 значений функции на заданном интервале. Исходные данные задать в файле Dat2.txt. Ре-

зультат поместить в файл вывода с именем Res2.txt в заданной форме (таблица 2.3).

Таблица 2.2

№	Функция	Контр. Значение		Интервал x		Вариант формы вывода
		x_*	y_*	x_{min}	x_{max}	
1	$y = \pi\cdot(x^3 - 6x^2)^{1/3}$	3,0	-9,4	-3	8	1
2	$y = \ln10\cdot(2 - \lvert x^5-1\rvert)$	1,0	4,6	-2	2	2
3	$y = (2/\pi)\cdot\mathrm{arctg}\, x^2$	-1,5	0,73	-3	3	3
4	$y = \pi x^5 - 5x^4 + 4$	1,6	4,2	-1,5	2,5	4
5	$y = \ln\pi\cdot\lvert e^x -1\rvert$	0,5	0,74	-2	2	1
6	$y = \ln10\cdot x^{1/2}\cdot e^{-x} + 1$	1,4	1,7	1	5	2
7	$y = (4x^3 - x^4)\cdot e^{-1/2}$	1,5	5,1	-1	4	3
8	$y = x^3 - \pi x^2 - 9x+35$	-0,8	39,7	-4	4	4
9	$y = x\cdot\sin x / \pi$	$\pi/2$	0,5	-5	5	1
10	$y = 2\pi / (x^2 + \pi)$	0,5	1,9	-3	3	2
11	$y = \ln\pi\cdot(x^3 + x^2)^{1/2}$	-0,6	0,43	-1	1,5	3
12	$y = \ln8 - 5x^4 + \pi x^5$	0,7	1,4	-1,5	3,5	4
13	$y = (2\pi)^{-1/2}\cdot e^{-x}$	0,5	1,5	0	3	1
14	$y = (x^2 - 5x + 6)/(x^2 + 1)$	-2	4,0	-6	6	2
15	$y = e^{-x}\cdot\sin(2\pi x)$	0,2	0,78	0	2	3
16	$y = 2x - \pi x^{2/3}$	1,2	-1,1	-0,5	4	4
17	$y = \pi(1,41 - x^2)^3$	0,5	4,9	-1,5	1,5	1
18	$y = e^x\cdot x/(x^2 + 1)$	1/e	6,5	-3	3	2
19	$y = 1,41(x^3 + x^4/4)$	-2,5	-8,3	-4	2	3
20	$y = 1/e + \pi\cdot\lvert x\rvert^{1/2}$	0,8	3,2	-2	2	4
21	$y = -\pi^2 x / (x^2 + 1)$	1,2	-4,9	-3	3	1
22	$y = x2 - \pi\cdot\lvert x-1\rvert + 1$	-2,1	-4,3	-5	4	2
23	$y = e^{1/2}\cdot\lvert\lvert x^2 - 1\rvert - 2\rvert$	0,25	1,75	-2	2	3
24	$y = \pi^{1/2}\cdot x\cdot\mathrm{arctg}x$	1	1,4	-2	3	4
25	$y = x^{2/3} - (x^2 - 1)^{1/3}$	0,5	1,5	-2	2	1

№	Функция	Контр. Значение		Интервал x		Вариант формы вывода
		x_*	y_*	x_{min}	x_{max}	
26	$y = x\cdot(1-x^2)^{1/2}/\ln 2$	0,8	0,69	-1	1	2
27	$y = 2x^{5/3} - x^{2/3} + \pi$	1,1	4,4	1,5	2,5	3
28	$y = \pi^3(x^2 - x^4)^{1/2}$	0,7	15,5	-1	1	4
29	$y = (x+1)^{2/3}/e^{1/2}$	1,2	1,03	-2	2	1
30	$y = \pi^2x^2 - 100(1+x)^{1/2}$	1,5	-136	-1	3	2

Таблица 2.3

Вариант формы вывода	Форма вывода информации
1	Таблица значений I----------I I X I Функция I I----------I I X=… I Y=… I I X=… I Y=… I I----------I Составил: < Ф.И.О. >
2	Таблица ********** * X=… * Y= … * ********** * X=… * Y=… * ********** Составил: < Ф.И.О. >
4	Получено: для заданной функции Y(…)= … для заданной функции Y(…)= … . Составил: < Ф.И.О. >

ЛАБОРАТОРНАЯ РАБОТА №3

Управляющие операторы безусловного и условного переходов. Разветвляющие программы

1.Краткие теоретические сведения

Как известно, из предыдущих лабораторных работ, все программы (макросы, подпрограммы процедуры, функции) состоят из последовательности операторов, которые обычно выполняются в том порядке, в каком они записаны в программе. Однако часто возникает необходимость пропустить какую-то группу операторов или наоборот выполнить её в зависимости от некоторых заданных условий, а также - повторить группу операторов несколько раз, т. е. организовать цикл. Для выполнения этих задач служат управляющие операторы. Управляющие операторы подразделяются на операторы принятия решения, к ним относятся операторы без условного и условного переходов, и операторы для организации циклов, которые будут рассмотрены в следующей лабораторной работе.

Оператор безусловного перехода Go To

Оператор безусловного перехода **Go To** (перейти к) осуществляет переход, без проверки каких-либо условий, к оператору, обозначенному соответствующей меткой. Синтаксис этого оператора выглядит следующим образом:

Go To *метка*

...

метка : оператор

где *метка* - метка. Это любой допустимый идентификатор VBA, который помещается слева от *оператора*, которому надо передать управление выполнением программы и отделяется от него

двоеточием. Причём *метка* может ставиться у *оператора* расположенного как до оператора **Go To**, так и после него. В случае если оператор **Go To** используется самостоятельно, без каких либо конструкций, то первый оператор, следующий за оператором **Go To**, должен иметь свою метку, иначе он не будет выполнен в процессе работы программы. Обычно оператор **Go To** используется совместно с оператором условного перехода **If**, и используется в программах редко т. к. есть более эффективные операторы.

Оператор условного перехода If и его конструкции

Оператор условного перехода **If** (если) служит для управления последовательностью выполнения операторов программы в зависимости от некоторого заданного условия. Он предназначен для выбора одного из возможных вариантов исполнения операторов программы в зависимости от выполнения или не выполнения заданного условия. Поэтому его называют оператором принятия решения. Существует несколько разновидностей конструкций этого оператора. Рассмотрим их последовательно по мере усложнения.

Если при выполнении некоторого условия необходимо выполнить лишь один оператор, то целесообразно использовать конструкцию следующего вида:

If *условие* **Then** *оператор (или группа операторов которая может следовать до конца строки)*

В качестве *условия* выбора используется значение логического выражения. При выполнении этой конструкции вначале вычисляется значение логического выражения, записанного в *условии,* после служебного слова **If.** Результат вычисления имеет тип **Boolean.** Если вычисленное значение выражения True (Истина), то выполняется *оператор* указанный после служебного слова **Then** (тогда) *или группа операторов, которая может следовать* после **Then** *до кон-*

ца строки. Если же после проверки *условия* было получено значение False (ложь), то выполняется первый оператор, следующий за этой конструкцией.

Пример 1.

```
Max = b
If a > b Then Max = a
MsgBox "Max=" & Max
```

В данном фрагменте программы определяется наибольшее значение среди переменных a и b и присваивается переменной Max. Фрагмент программы данного примера, записанный с использованием оператора безусловного перехода, будет выглядеть следующим образом.

Пример 2.

```
If a > b Then Max = a: GoTo L1
Max = b
L1: MsgBox "Max=" & Max
```

Видно, что этот вариант написания фрагмента программы менее рационален, чем предыдущий. Замечание: в качестве меток, возможно, использование целочисленных констант, в этом случае ставить двоеточие после метки необязательно.

Если при выполнении *условия* оператора **If** требуется выполнить не один, а несколько операторов, записанных построчно, т. е. *блок операторов*, то следует использовать следующую блочную конструкцию:

If *условие* **Then**

блок операторов

End If

В случае истинности проверяемого *условия* будет выполняться *блок операторов* расположенный после служебного слова **Then**. Если *условие* является ложным, то выполняется следующий оператор после данной конструкции, т.е. после служебного словосочетания **End If**. В том случае, когда *блок операторов* состоит из одного оператора, данная конструкция всё равно должна заканчиваться служебным словосочетанием **End If**.

Пример 3.

```
If a>b Then
Max= a
Min =b
End If
```

Если a>b, то переменной Max присваивается переменная a, переменной Min присваивается переменная b. В противном случае, т.е. если, a меньше или равно b**,** то будет выполняться следующий оператор после **End If**.

Если необходимо задать выполнение одного из двух блоков операторов, в зависимости от результата проверки *условия* выбора, то может использоваться один из следующих вариантов конструкции условного оператора **If**:

If *условие* **Then**
операторы блока 1
Else
операторы блока 2
End If
	или	**If** *условие* **Then**
		операторы блока 1
		Else: *операторы блока 2*
		End If

Если результатом проверки *условия* является значение True (Истина**)**, то будут выполнены *операторы блока 1*. Далее будет вы-

полняться первый оператор, следующий за оператором **End If**. *Операторы блока 2* выполнены не будут. Если проверка *условия* даст результат False (ложь), то *операторы блока 1* будут пропущены, а будут выполнены *операторы блока 2*, расположенные после служебного слова **Else (**иначе**)**. Далее будет выполняться первый оператор, следующий за оператором **End If.** Каждый из указанных блоков операторов может состоять из одного оператора. Оба варианта конструкции оператора **If** работают одинаково хорошо и выбор между ними - вопрос вкуса и/или привычки. Заметим, что конструкцию **If…Then…Else** допускается записывать в одну строку. Однако, однострочный вариант записи не всегда удобен при написании программ, поэтому лучше использовать блочную форму записи.

Пример 4.

If a>b **Then**		**If** a>b **Then**
Max= a		Max= a
Else	или	**Else**: Max=b
Max=b		**End If**
End If		

Если a>b, то переменной Max будет присвоена переменная a, в противном случае переменной Max будет присвоена переменная b. Во втором варианте этой конструкции в качестве операторов *блока 2* может использоваться как один оператор (тогда он записывается после символа двоеточие в той же строке, что и служебное слово **Else**), так и несколько. Тогда каждый оператор, начиная со второго, записывается в отдельной строке, либо все операторы записываются в одной строке, но разделённые символом двоеточие, как в следующем примере.

Пример 5.

```
If a>b Then                          If a>b Then
  Max= a                               Max= a
  Min = b                              Min = b
Else: Max=b          или             Else: Max=b: Min=a
  Min =a                             End If
End If
```

Если a>b, то переменной Max будет присвоено значении a, переменной Min - значение b, в противном случае переменной Max присваивается значение b, а переменной Min –значение a.

Вложенные конструкции оператора If

В том случае, когда определённый *блок операторов* (или один оператор) нужно выполнить после проверки ни одного, а нескольких условия, то используется один из следующих вариантов вложенной конструкции оператора **If**:

```
If условие 1 Then                    If условие 1 Then
операторы блока 1                    операторы блока 1
Else                                   ElseIf условие 2 Then
  If условие 2 Then        или        операторы блока 2
  операторы блока 2                    Else
  Else                                 операторы блока 3
  операторы блока 3                  End If
  End If
End If
```

Если *условие 1* истинно, то выполняются *операторы блока 1.* Далее в первом варианте конструкции управление будет передано оператору**,** стоящему за вторым оператором **End If**, а во втором

варианте конструкции за оператором **End If** , который в данной конструкции записывается один раз. Если же оно ложно, то происходит проверка *условия 2*, находящегося во вложенном условном операторе **If** после служебного слова **Else**, для первого варианта конструкции, или после служебного слова **ElseIf** для второго варианта. Если *условие 2* истинно, то выполняются *операторы блока 2*, если же оно ложно, то выполняются *операторы блока 3*. После чего и в том и в другом случае выполняется первый оператор после оператора **End If**. Оба варианта вложенной конструкции оператора **If** работают одинаково хорошо и выбор между ними – вопрос вкуса и/или привычки.

Пример 6.

```
If x< -1 Then                          If x< -1 Then
N=1                                    N=1
   Else                                   ElseIf x>1 Then
   If x>1 Then          или               N=2
   N=2                                    Else 'или Else: N=0
   Else                                   N=0
   N=0                                 End If
   End If
End If
```

Если x<-1, то переменной N будет присвоено значение 1, и в первом варианте конструкции управление будет передано оператору**,** стоящему за вторым оператором **End If**, а во втором варианте конструкции за оператором **End If** , который в данной конструкции записывается один раз. Если же условие x<-1 не выполняется, то в первом варианте конструкции работает вложенный оператор **If**, а во втором варианте - **ElseIf**. Если вложенное условие x>1 является

истинным, то N будет присвоено значение 2, в противном случае N будет присвоено значение 0.

Следует заметить, что конструкция **If… Then…ElseIf** имеет свои особенности. С её помощью можно в одном условном операторе проверять несколько условий. Синтаксис этой конструкции может выглядеть следующим образом:

If *условие 1* **Then**
*блок операторов 1 'или один опера*тор
ElseIf *условие 2* **Then**
    блок операторов 2
ElseIf *условие 3* **Then**
    блок операторов 3
. . .
[Else] *'необязательный оператор*
    блок операторов N
End if

Конструкция работает следующим образом. Проверяется *условие 1,*если оно равно True, то выполняется *блок операторов 1*.Затем все остальные операторы пропускаются, и выполняется первый из операторов, следующих за оператором **End If**. Если значение *условия 1* будет равно False, то проверяется значение *условие 2*.Если значение *условия 2* будет True, то выполняется *блок операторов 2*,а затем все остальные операторы пропускаются, и выполняется первый из операторов, следующих за оператором **End If**. Если значение *условия 2* будет равно False, то проверяется значение *условия 3* и так далее. Если в конструкцию включён оператор **Else** и все условные выражения (*условие 1, условие 2* и т.д.) окажутся равными False ,то будет выполнен *блок операторов N*, расположенный после

оператора **Else**, и только после этого будет выполнен первый из операторов, следующих за оператором **End if**.

Пример 7.

```
Sub Example7()
Dim n As String
x = InputBox("Введите число”)
If x < -1 Then
   n = "отрицательное"
ElseIf x = 0 Then
   n = "ноль"
ElseIf x > 1 Then
    n = "положительное"
End If
MsgBox "n=" & n
End Sub
```

Из приведённого примера 7 видно, что служебное слово **ElseIf** может повторяться в конструкции сколько угодно раз. При этом заданные условия проверяются строго последовательно, в том порядке в каком они записаны. И как только истинное условие (True) будет найдено, выполняется соответствующий ему оператор или блок операторов, после чего выполнение данного условного оператора прекращается, управление передается первому оператору, следующему за оператором **End If**

Функций IIf

В дополнение к рассмотренным конструкциям условного оператора **If** следует рассмотреть функцию IIF, которая возвращает

одно из двух значений в зависимости от проверяемого условия. Её синтаксис:

IIf (*условие, значение 1, значение2)*

Если результатом проверки *условия* является значение True, функция возвращает *значение 1*,а когда проверка даёт значение False, то возвращаемый результат – *значение 2.*

Пример 8.

N = IIf (Y<0, 1, 2)

Если Y<0, то переменной N будет присвоено значение 1, в противном случае - значение 2.

Во всех рассмотренных конструкциях оператора **If** при записи его условий выбора могут использоваться следующие операторы сравнения и логические операторы.

Операторы сравнения

Для проверки условия равенства двух выражений, как и в алгебре в программировании используется символ =. Аналогично: символ > используется для проверки условия «больше»; символ < для проверки условия «меньше»; >= - «больше или равно»; <= - «меньше или равно»; <> - «не равно»; is - сравнение двух операндов содержащих ссылки на объекты; Like - сравнение двух строковых выражений.

Логические операторы

AND - конъюнкция (логическое И) используется для логического объединения двух выражений;

EQV - логическая эквивалентность используется для проверки эквивалентности двух выражений;

OR - дизъюнкция (логическое ИЛИ) используется чтобы убедиться в том, что хотя бы одно из выражений истинно;

XOR - исключение (логическое исключающее ИЛИ), используется чтобы убедиться в том, что истинно одно из двух выражений;

NOT - отрицание (логическое НЕ), возвращает обратное логическое выражение.

Пример 9.

If X>-1 AND X<1 **Then**

В условии оператора **If** записано обычное алгебраическое неравенство -1<X<1.

Отметим, что в операторе **If** допускается записывать условия в круглых скобках так, как в следующем примере.

Пример 10.

If (X-<1 OR X>1) **Then**

В условии оператора **If** записано алгебраическое неравенство X<-1 либо X>1.

Конструкция Select Case

В случае, когда необходимо сделать выбор при наличии нескольких условий, запись рассмотренных конструкции оператора **If** будет достаточно громоздкой. В подобных случаях следует использовать конструкцию **Select Case**(выбор случая). Эта конструкция позволяет проверить одно и то же выражение, сравнивая его с различными условиями, к тому же она улучшает читаемость программы. Её синтаксис выглядит следующим образом.

Select Case *ключ выбора*

Case *значение 1 ключа выбора*
операторы блока 1 *'или один оператор*

...
Case *значение N ключа выбора*
операторы блока N
...
[**Case Else**
операторы блока K]
End Select

где *ключ выбора* - переменная или выражение;
значение 1 ключа выбора… значение N ключа выбора может быть численным, логическим или символьным.

Выполнение оператора. Если *ключ выбора* имеет *значение 1*,то будут выполняться *операторы блока 1*, если – *значение N*, то - *операторы блока N*. Если *ключ выбора* не равен ни одному из приведенных, возможных его *значений,* то будут выполняться *операторы блока K*

Кроме того в операторе **Case** можно указывать диапазон значений:

от значения 1 ключа выбора **To** *к значению N ключа выбора*

В операторе **Case** можно записывать операции сравнения:

Case is *оператор сравнения* («<» , «> «, «=» и т.п.) *значение для сравнения*

В этом случае значение ключа выбора сравнивается со значением для сравнения.

В одном операторе **Case** можно указывать несколько значений, а также условия, которые разделяются запятыми, например:

Case 2 **To** 10, 14, **is** >20

Пример 11.

Select Case X

Case 1

N=30

Case 2, 3

N=20

Case 4 To 10

N=40

Case is < 0

N=10

Case Else

N=0

End Select

Значения *ключа выбора X* сравнивается со значениями указанными в операторах **Case**. Если значение X равно 1, то переменной N присваивается значение 30. Если X равно 2 или 3, то переменной N присваивается значение 20. Если X принадлежит интервалу от 4 до 10, то переменной N присваивается значение 40. Если X меньше нуля, то N примет значение 10. Во всех остальных случаях N примет значение 0.

Допускается вложенность операторов **Select Case**. При этом каждому вложенному оператору **Select Case** должен соответствовать оператор **End Select**.

Разветвляющиеся программы

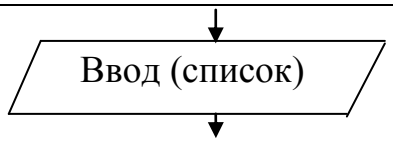
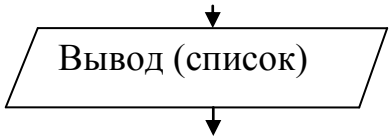
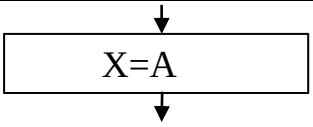
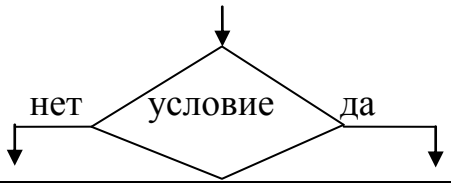
Разветвляющиеся программы это - такие программы, в которых на определённых этапах происходит анализ значений тех или иных

параметров и в зависимости от этого выбирается один из возможных вариантов дальнейшего хода программы. Практически все более или менее сложные программы являются разветвляющимися. Для их написания используются рассмотренные конструкции управляющих операторов принятия решения.

При написании разветвляющих программ предварительно составляется блок-схема алгоритма решения задачи. Блок-схема это – графическое изображение алгоритма или последовательности решения задачи программирования.

Для составления блок-схем используются стандартизованные графические изображения (блоки) определённых операторов алгоритмического языка. Некоторые из них представлены в таблице 3.1.

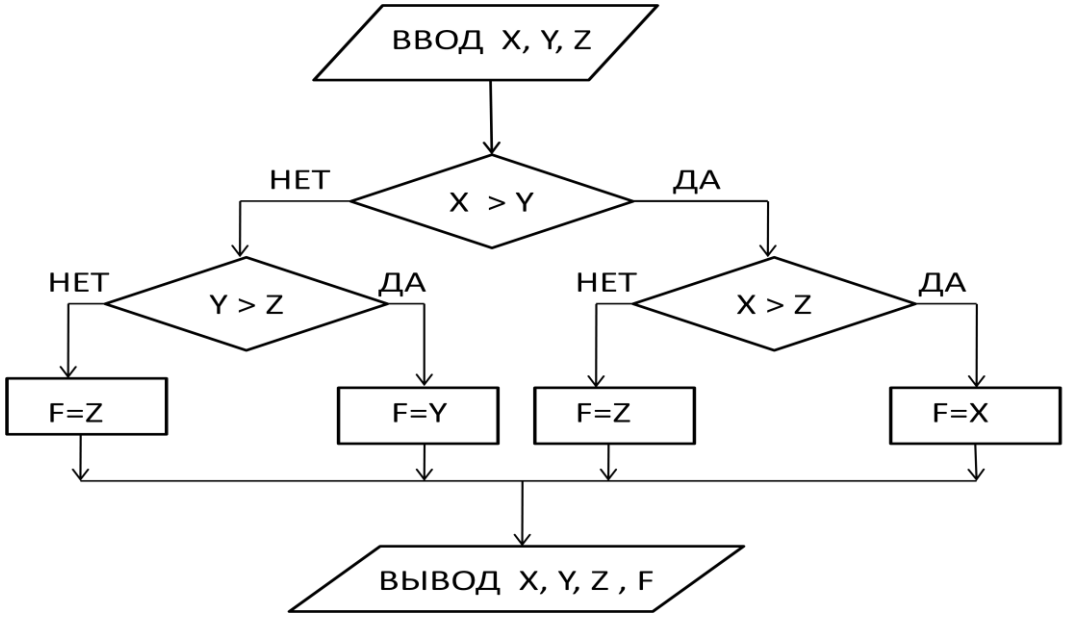
Таблица 3.1

№	Название блока	Графическое изображение блока	Операторы и функции эквивалентные блоку
1	Блок ввода		Операторы ввода, функция INPBOX и другие
2	Блок вывода		Операторы вывода, функция MSGBOX и другие
3	Блок присваивания		Оператор присваивания
4	Блоки сравнения		Условный оператор If

Далее рассмотрены примеры написания разветвляющихся программ, аналогичные тем, которые должен выполнить студент в данной лабораторной работе.

Пример 12. Составить блок-схему, написать для неё и отладить процедуру, которая производит выбор наибольшей из трёх заданных величин X , Y и Z и присваивает её значение переменной F, т. е. вычисляет F=max(X,Y,Z). Замечание: Данный пример является тренировочным, на практике подобные задачи решаются с помощью соответствующих встроенных функций.

Блок-схема



Пояснения к блок-схеме. После ввода численных значений для переменных X ,Y и Z производиться их последовательное сравнение друг с другом на предмет выявления наибольшего из них. Первоначально сравниваются значения переменных X и Y. Если условие X> Y выполняется (истинно), то далее переменная с наибольшим значением, а именно X сравнивается с Z. Если поставленное в блоке сравнения условие X>Z верно, то переменной F будет присвоено значение переменной X в противном случае – значение переменной Z. Аналогично поступаем в случае если условие X> Y, не выполняется (ложно).

После составления блок-схемы по ней пишется процедура, при этом каждый блок описывается соответствующим оператором алгоритмического язык.

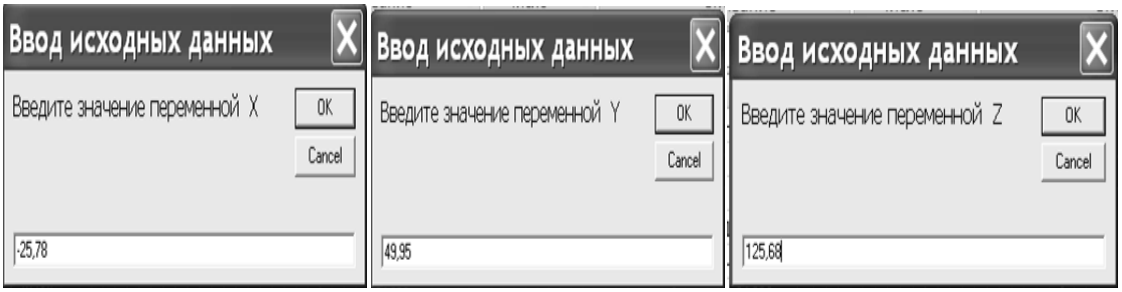
Процедура (листинг-копия рабочей процедуры)

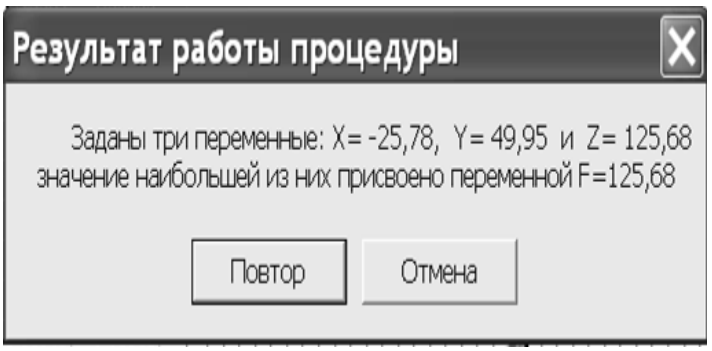
```
Sub Example12()
Title = "Ввод исходных данных" :
Vvod = "Введите значение переменной "
10    X = CSng(InputBox(Vvod & " X", Title))          'Ввод X
      Y = CSng(InputBox(Vvod & " Y", Title))          'Ввод Y
      Z = CSng(InputBox(Vvod  & "Z ", Title))         'Ввод Z
      If X > Y Then
         If X > Z Then  'начало первой вложенной конструкция If
           F = X         'Ветвь «да»
         Else: F = Z     'Ветвь «нет»
         End If          'конец первой вложенной конструкция If
ElseIf  Y > Z Then      ' начало второй вложенной конструкция If
F = Y                     ' Ветвь «да»
Else: F = Z               ' Ветвь «нет»
End If                    'конец второй вложенной конструкция If
ВЫВОД = MsgBox("          Заданы три переменные: X= " & X _
& ", Y= " &Y & " и Z= " & Z & vbLf _
& "значение наибольшей из них присвоено переменной F=" & F, _
vbRetryCancel, "Результат работы процедуры")
Select Case ВЫВОД        'Выбор
Case vbRetry  : GoTo 10   ' повторения
Case vbCancel: GoTo 20    ' либо завершения работы процедуры
End Select
20 End Sub
```

Пояснения к процедуре. После ввода значений переменных X,Y и Z согласно блок-схеме проверяется условие X>Y. Если оно ис-

тинно, то выполняется первая вложенная конструкция оператора **If**, в которой проверяется условие X>Z, если оно истинно, то переменной F присваивается значение X в противном случае -Y. Если же условие X>Y ложно, то выполняется вторая вложенная конструкция оператора **If**, в которой проверяется условие Y>Z, если оно истинно, то переменной F присваивается значение *Y* иначе - *Z*. В конце процедуры предусмотрен выбор либо повторения работы процедуры, либо её завершения. Происходит это в зависимости от того какое значение передаст функция MsgBox (см. лаб. работу №2) в переменную ВЫВОД. Если при работе процедуры пользователем будет нажата кнопка «повтор», то в переменную ВЫВОД будет передано значение vbRetry и конструкции **Select Case** передаст управление на метку 10 для повторного ввода данных, а если – «отмена», то значение vbCancel и конструкция **Select Case** передаст управление на метку 20 **End Sub** –завершение работы процедуры. Так как результатом вызова функции InputBox является строковое значение, то чтобы исключить возможность возникновения ошибки, связанной с несовпадением типов используемых переменных, её значение должно быть преобразовано при помощи функции изменения типа переменных **CSng** (см. лаб. работу №1) в численное значение.

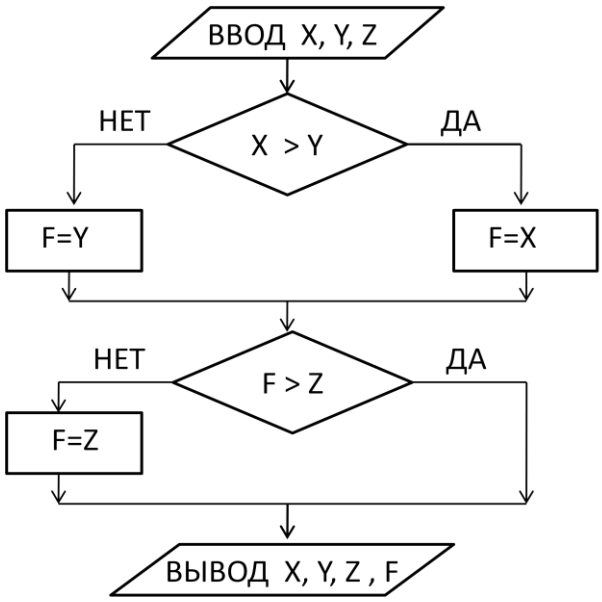
Работа процедуры





Рассмотренный алгоритм решения задачи примера 12 не является единственным. Ниже представлена блок-схема другого варианта алгоритма и основной фрагмент другого варианта процедуры, с использованием функции IIF.

Блок-схема Основной фрагмент процедуры




```
. . .
F = IIF (X>Y, X, Y)
F = IIF (F>Z, F, Z)
. . .
```

Пример 13. Составить блок-схему, написать для неё и отладить процедуру, которая определяет номер N области, в которой находиться точка M(x,y) с заданными координатами (см. рисунок 3.1). Границы области относить к области с наибольшим номером.

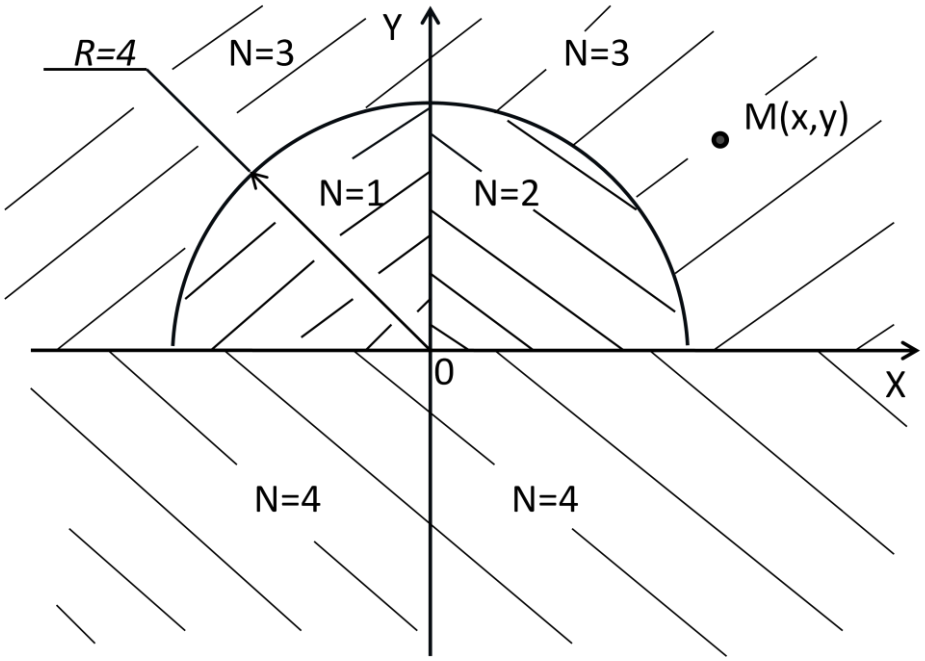
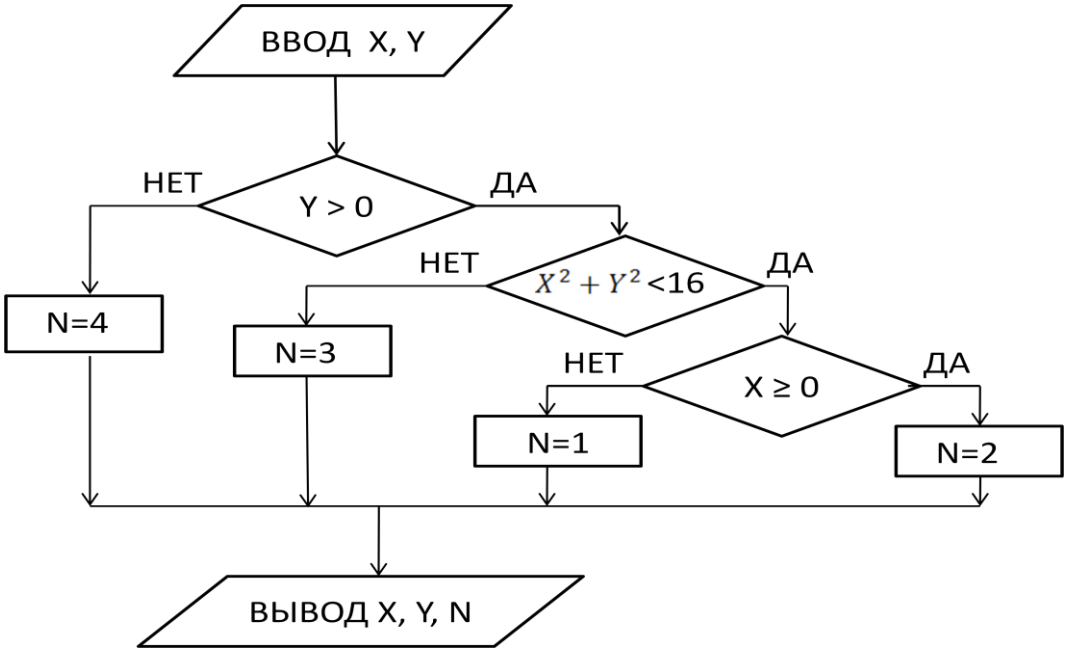


Рис. 3.1

Блок-схема



Пояснения к блок-схеме. В первом блоке производиться ввод численных значений для переменных X и Y, которые являются координатами точки M. Далее целесообразно сравнить переменную Y (координата по оси «y») с нулём. В блок-схеме это первый блок

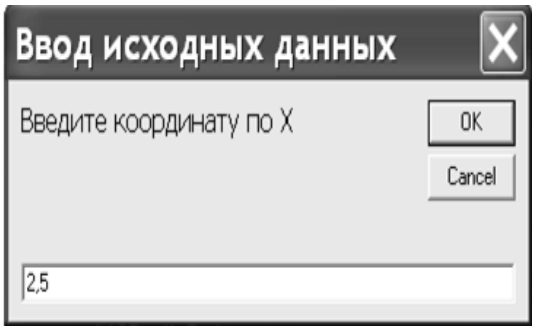
сравнения, если его условие Y>0 не выполняется (ложно), то координата по оси «Y» точки M отрицательна или равна нулю, а это значит, что она расположена ниже оси «X» или на ней, т.е. в области с номером N=4. Если условие Y>0 первого блока сравнения выполняется (истинно), то точка M расположена выше оси «X», а это значит, что она может находиться в одной из областей с номером N=1, N=2 или N=3. Далее для определения номера области целесообразно задать во втором блоке сравнения условие $X^2+Y^2<16$, которое следует из уравнения окружности $X^2+Y^2=R^2$,где R радиус окружности. Если заданное условие выполняется, то точка M расположена внутри окружности , а так как Y>0 то внутри полуокружности. Согласно условию задачи внутри полуокружности точка может находиться либо в области с номером N=1 либо в области с N=2. Если условие X ≥0 третьего блока сравнения выполняется (истинно), то точка расположена в области с N=2, в противном случае с N=1. После чего идёт печать результата. Если условие $X^2+Y^2<16$ второго блока сравнения не выполняется (ложно), то точка M находиться вне полуокружности и над осью «X» так как Y>0 т.е. в области N=3. Далее представлена процедура, составленная по рассмотренной блок-схеме.

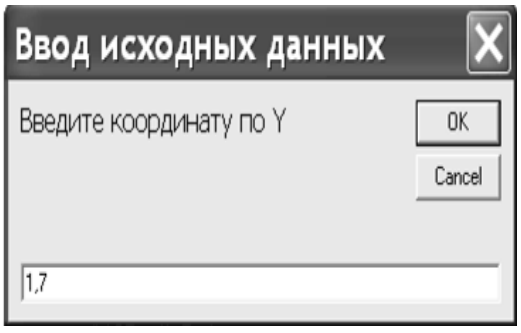
Процедура (листинг)

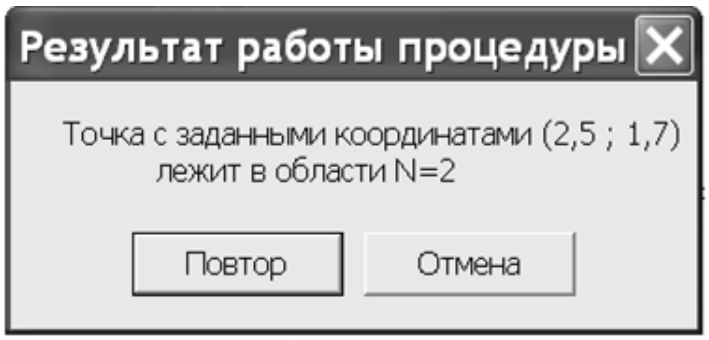
```
Sub Example13()
Dim X, Y As Single:Dim N As Byte
Title = "Ввод исходных данных": Vvod = "Введите координату по"
10 X = InputBox(Vvod & " X", Title)          'Ввод координаты по X
 Y = InputBox(Vvod & " Y", Title)            'Ввод координаты поY
If Y > 0 Then
    If X * X + Y * Y < 16 Then 'начало первой вложенной
                               ‘конструкции If
```

```
If X >= 0 Then          'начало второй вложенной конструкция If
N = 2
Else: N = 1
End If                  'конец второй вложенной конструкция If
Else: N = 3
End If                  'конец первой вложенной конструкция If
Else: N = 4
End If
                        'ВЫВОД РЕЗУЛЬТАТА
    Вывод = MsgBox(" Точка с заданными координатами (" _
    & X & " ; " & Y & ")" & vbLf _
    & "             лежит в области N=" & N, _
    vbRetryCancel, "Результат работы процедуры")
    Select Case Вывод           'Выбор
    Case vbRetry:   GoTo 10     ' повторения
    Case vbCancel: GoTo 20      ' либо завершения работы
                                'процедуры
    End Select
  20 End Sub
```

Работа процедуры







2.Практическая часть

Задания к лабораторной работе

Составить блок-схему, написать для неё и отладить процедуру для выполнения следующих заданий. При этом руководствоваться выше приведёнными примерами выполнения заданий (см. примеры 12 и 13)

Задание 1. Вычислить для своего варианта значение функции F по соответствующим её выражениям. При получении в знаменателе нуля дать соответствующее сообщение.

Варианты заданий

1) $F=\frac{\min(x,y)+0{,}5}{(\max(x,y))^2-\sin z}$

2) $F=\frac{\min(x,y,z)+x}{(\max(x,z))^2+y}$

3) $F=\frac{\max(x,y)+y}{(\min(x,y,z))^2+yx}$

4) $F=\frac{\min(z,\max(x,y))}{x^2+z}$

5) $F=\frac{\max(x^2,y^2,xz)+x}{(\min(x,y))^2-y}$

6) $F=\frac{\min(x,y+z)}{\max(x^2,y)+z^3}$

7) $F=\frac{\max(x^2,y^2,x-y)+x}{(\min(x,y))^2+y^4}$

8) $F=\frac{\min(x,(x+y)^2)}{x^2+\max(y^3,x)}$

9) $F=\frac{\max(x+z,\min(x,y))}{x^2}$

10) $F=\frac{\min(x,\max(x+y,z))^2}{x^2+z^2}$

11) $F=\frac{\min(x,y+z)}{\max(x,y)+\sin z}$

12) $F=\frac{\min(x,y-x)}{\max(yz,x^2)+\cos 2z^3}$

13) $F=\frac{\max(x^2,z^2)+\cos 2x^2}{(\min(x,y))^2-y}$

14) $F=\frac{\min(x^2,y+z)}{x^2+\max(z^3,xy)}$

15) $F=\dfrac{\max(x+y,\max(x,zy))}{xe^2}$

16) $F=\dfrac{\max\left(x^3,y^2,xy\right)+x}{(\min(x,yz))^2-y}$

17) $F=\dfrac{\mathrm{x}(\max(x+z,zy))}{\min(x,y)+x^2}1$

18) $F=\dfrac{\min\ (\mathrm{x},\max(\mathrm{x}+\mathrm{z},\mathrm{y}))^2}{x^3+z^2}$

19) $F=\dfrac{x^3+\max(z^2,y)}{(\max(x,z))^2-y}$

20) $F=\dfrac{\max(\mathrm{x},\mathrm{y}+\mathrm{z})+e^{xz}}{\min(x^2,y)+z^3}$

21) $F=\dfrac{\min\left(x^2,z^4,xy\right)+x}{(\max(x,y))^2-y}$

22) $F=\dfrac{\min(\mathrm{x},\mathrm{y}+\mathrm{z})+e^x}{\max(x^2,y)+z^3}$

23) $F=\dfrac{\max\left(x^2,y^2\right)+\cos 4z^2}{\min(x,y)+x^2}$

24) $F=\dfrac{(\min(x,y))^2-y}{x^2+\max(z^3,x)}$

25) $F=\dfrac{\min\ (\mathrm{x},\mathrm{y}+2\mathrm{x})}{\max(y,z)+\sqrt[3]{x}}$

26) $F=\dfrac{\min\ (\mathrm{x},\mathrm{y}-\mathrm{z})}{\max\ (yz,x^2)+\cos 2x^2}$

27) $F=\dfrac{\max\left(x^3,z^2\right)+\cos 4y^2}{\min(y,yz)+\sqrt{x}}$

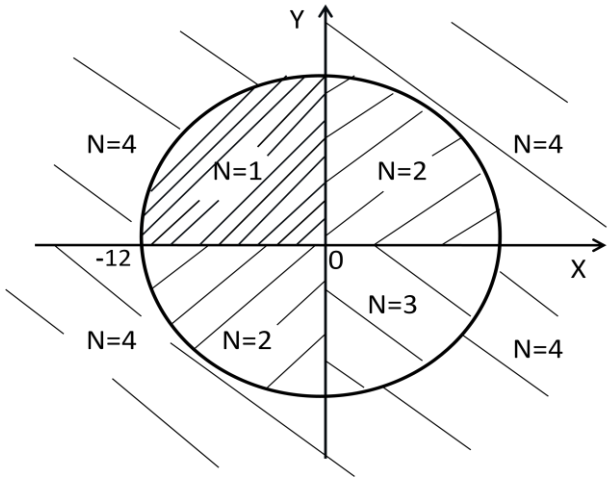
28) $F=\dfrac{(\min(x,y))^4+2e^x}{x^3+\max(z^2,x)}$

29) $F=\dfrac{\max(xz,\min\ (y,z))}{x^2+\sin zy}$

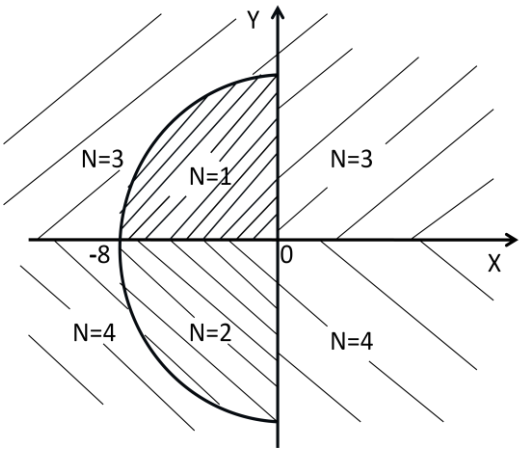
30) $F=\dfrac{\max\ (\mathrm{x},\max(\mathrm{y},\mathrm{z}))^4}{\sin 2y+xe^2}$

Задание 2. Определить для своего варианта номер N области, в которой находиться точка M(x,y) с заданными координатами. Границы области относить к области с наибольшим номером.

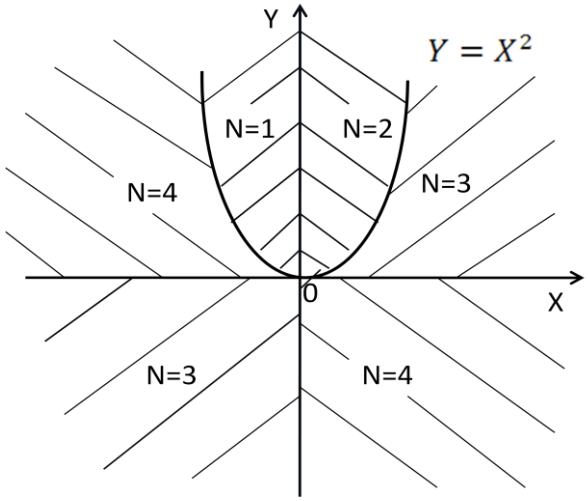
Вариант 1



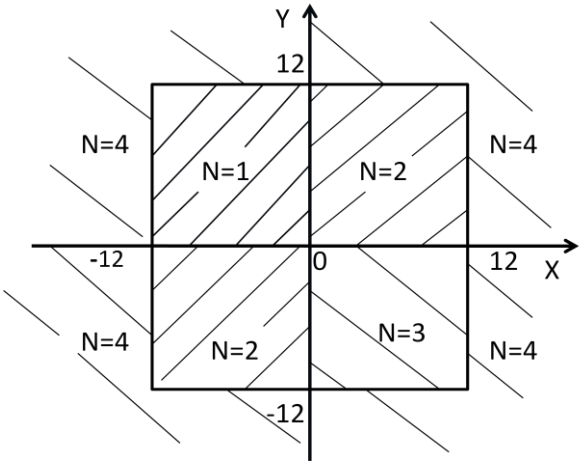
Вариант 2



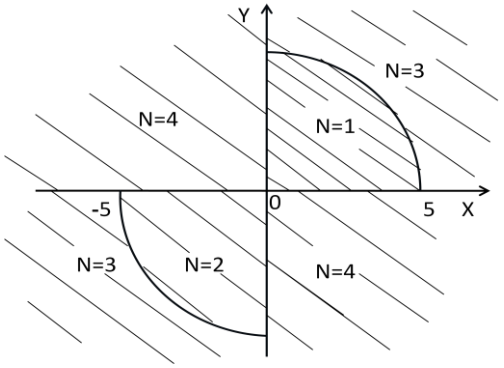
Вариант 3



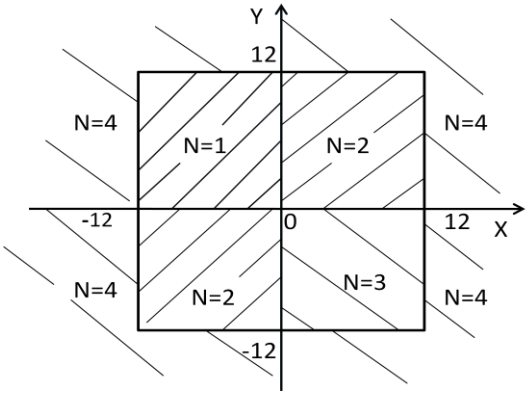
Вариант 4



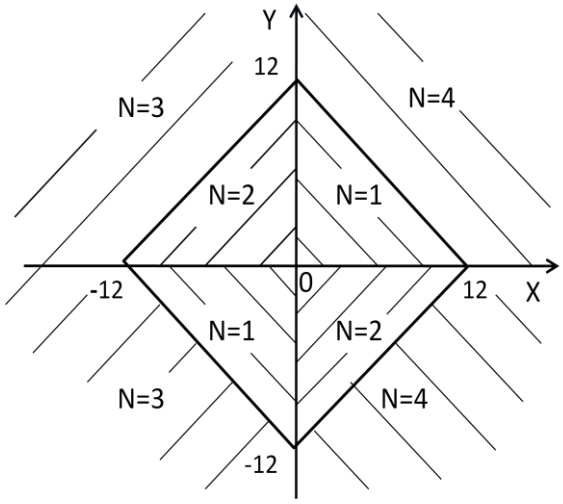
Вариант 5



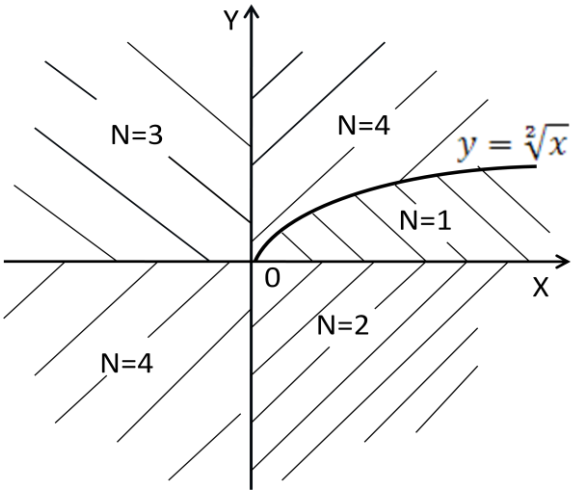
Вариант 6



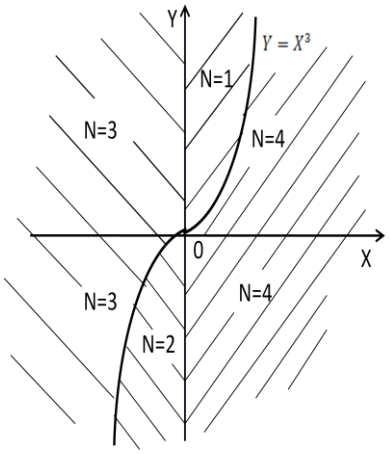
Вариант 7



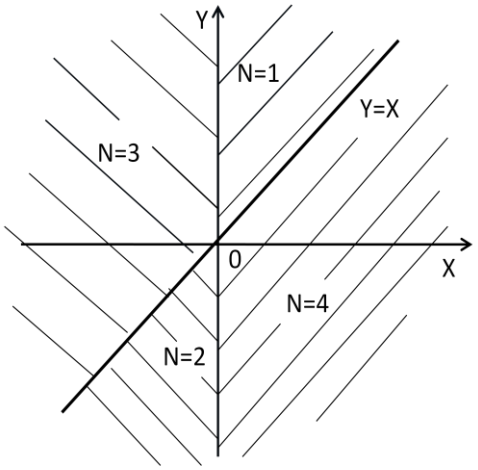
Вариант 8



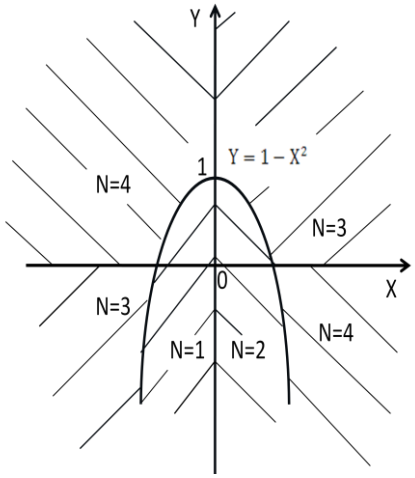
Вариант 9



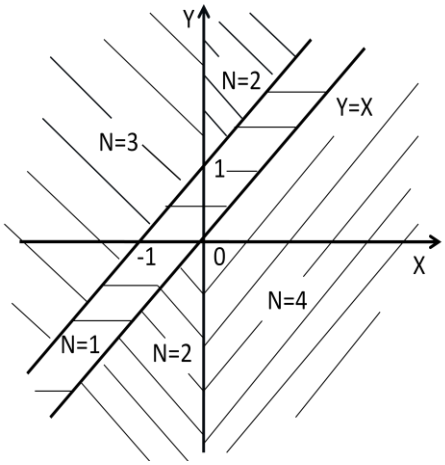
Вариант 10



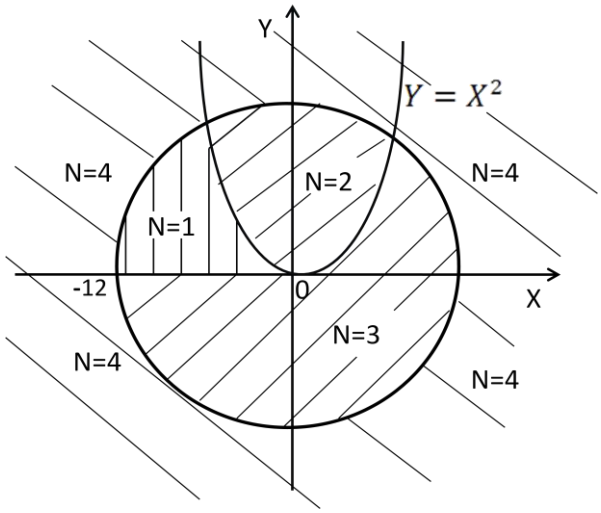
Вариант 11



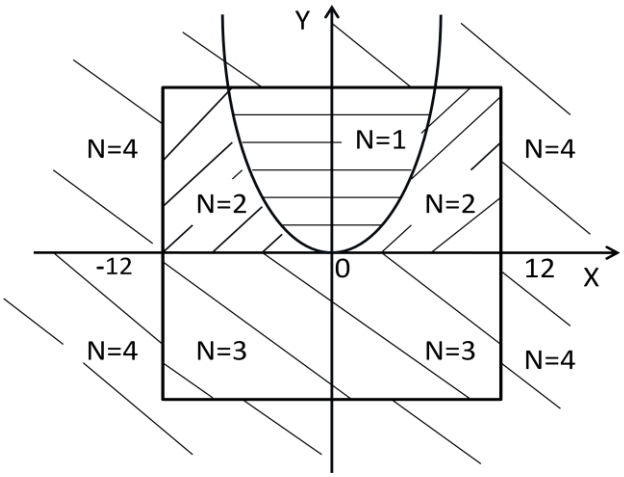
Вариант 12



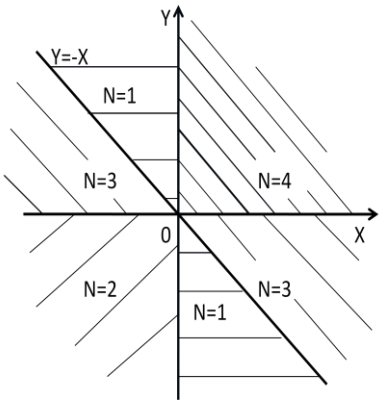
Вариант 13



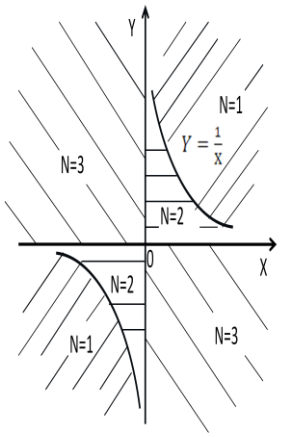
Вариант 14



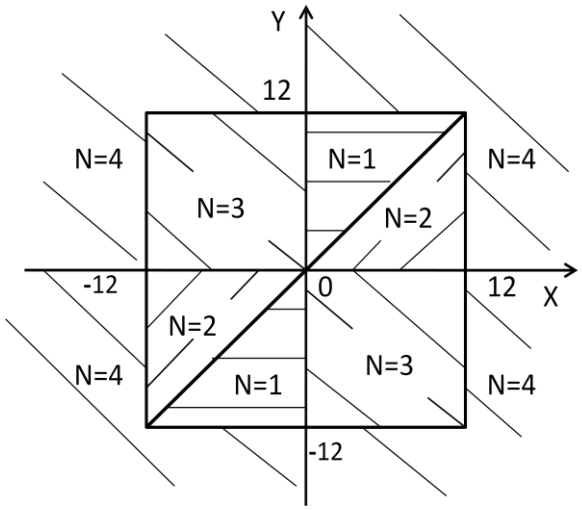
Вариант 15



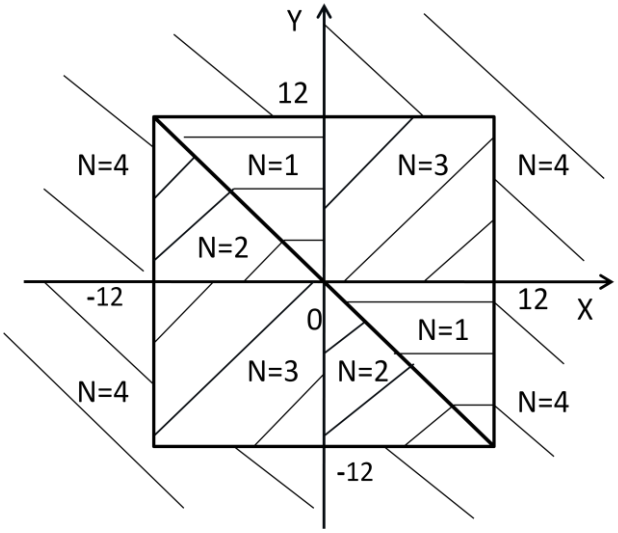
Вариант 16



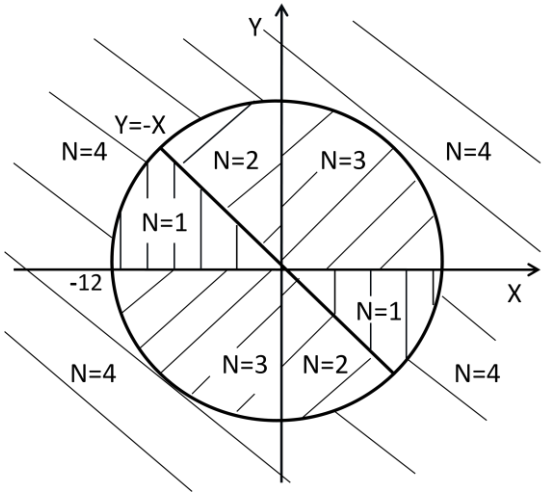
Вариант 17



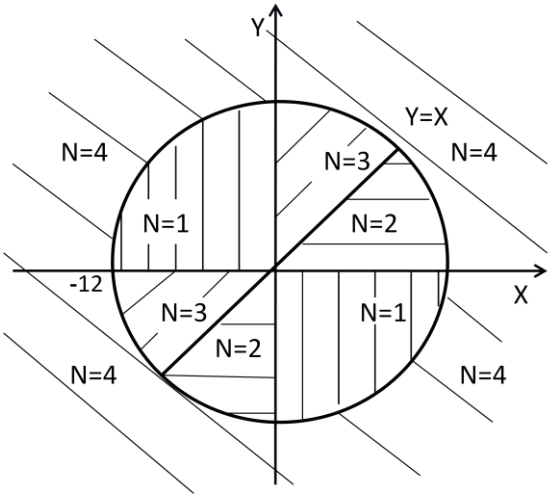
Вариант 18



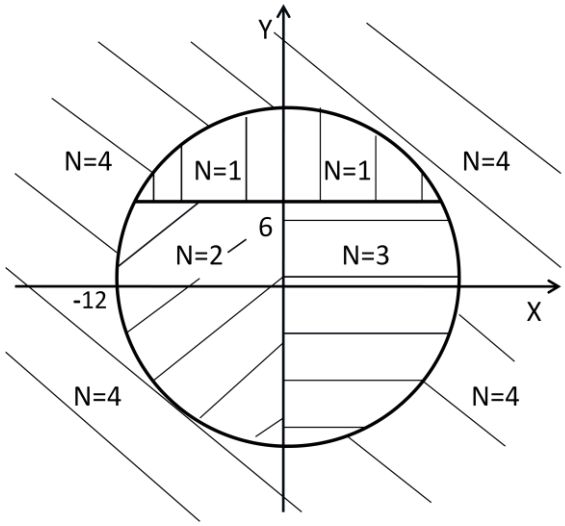
Вариант 19



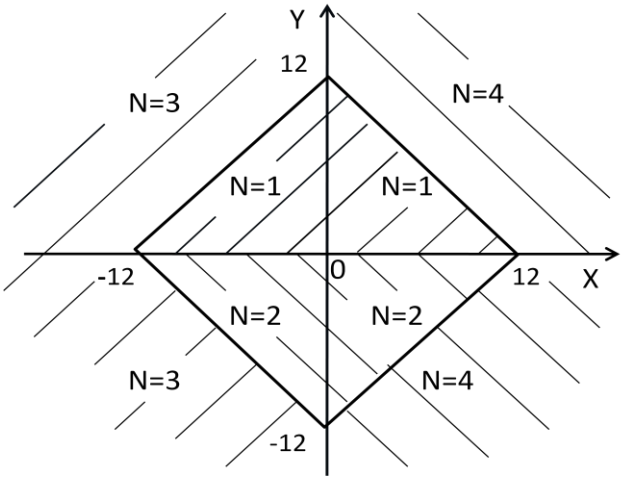
Вариант 20



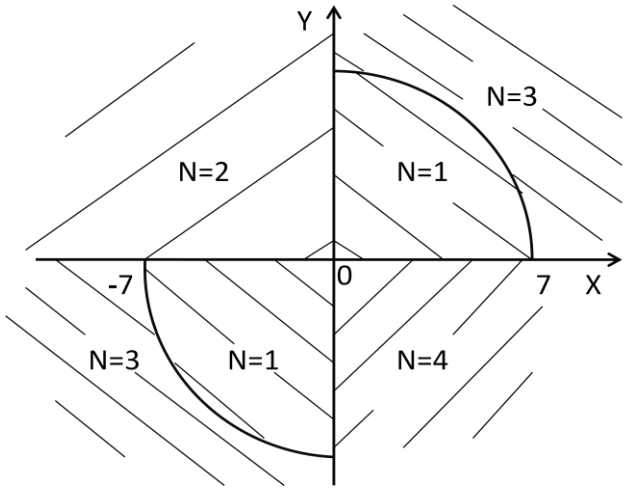
Вариант 21



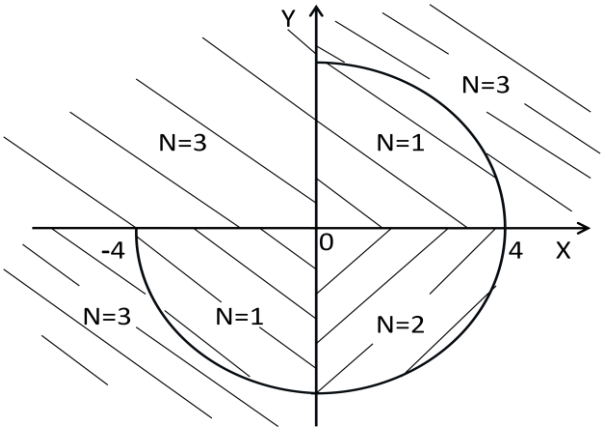
Вариант 22



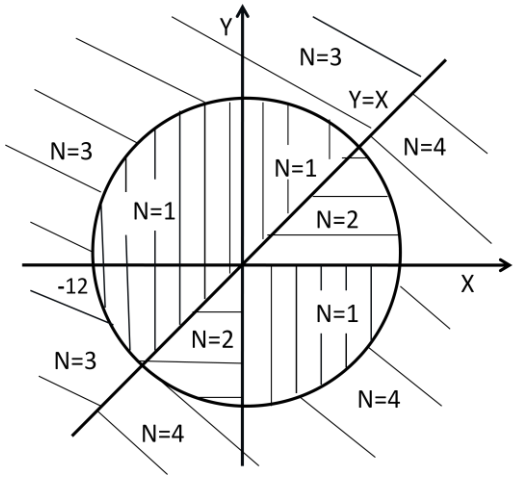
Вариант 23



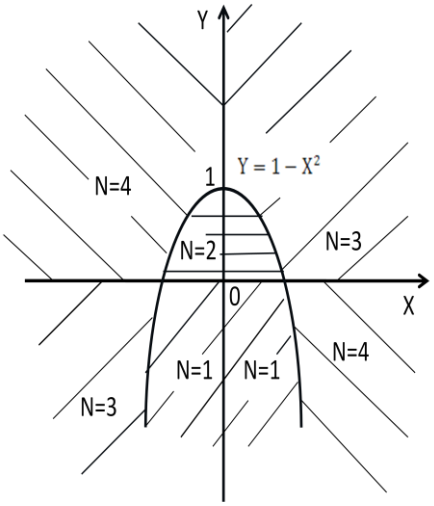
Вариант 24



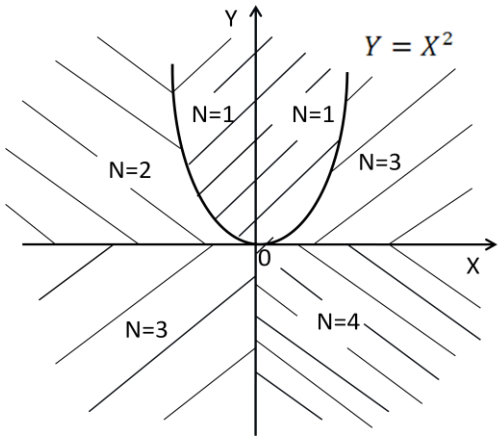
Вариант 25



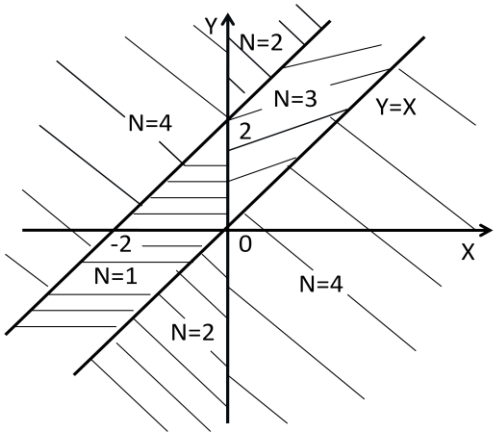
Вариант 26



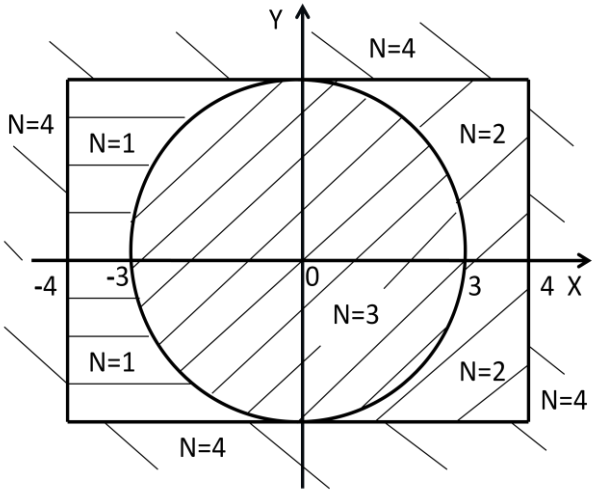
Вариант 27



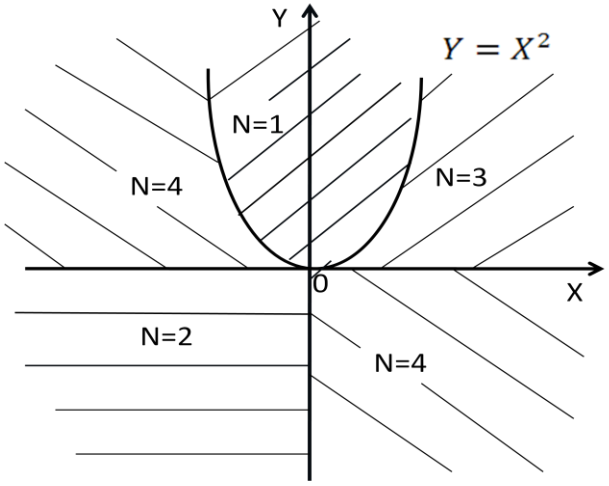
Вариант 28



Вариант 29



Вариант 30



ЛАБОРАТОРНАЯ РАБОТА №4
Управляющие операторы для организации циклов. Программы с циклами

1.Краткие теоретические сведения

Циклом называется процесс исполнения группы операторов программы заданное количество раз, либо до тех пор, пока выполняется или не выполняется некоторое поставленное условие. Процесс исполнения группы операторов цикла один раз называется итерацией цикла. Циклы, выполняющиеся заданное количество раз, называются циклами с фиксированным числом итераций, фиксированные циклы, ещё их называют безусловными. Циклы, выполняющиеся переменное количество раз, в зависимости от заданных условий, называются неопределёнными циклами. Группа операторов, расположенная между началом и концом цикла называется *телом цикла*. Самой простой конструкцией цикла является фиксированный цикл, который реализуется с помощью оператора цикла **For**.

Оператор цикла For

Оператор цикла **For** (для) служит для организации фиксированных, безусловных циклов с заданным количеством итераций - цикл по счётчику. Цикл выполняется от начального до конечного значения переменной цикла с заданным шагом. Синтаксис цикла:

For X = $X_{нач.}$ To $X_{кон.}$ [Step ΔX]

тело цикла

Next [X]

X - переменная (параметр, счётчик) цикла, представляет собой имя переменной численного типа, в ней сохраняется информация о количестве выполненных итераций;

$X_{нач.}$ - начальное значение переменой цикла, это может быть костанта, переменная или арифметическое выражение;

$X_{кон.}$ - конечное значение переменной цикла, это может быть константа, переменная или арифметическое выражение;

ΔX - шаг переменной цикла, это может быть константа, переменная или арифметическое выражение.

Оператор цикла работает следующим образом. Переменная X цикла принимает первоначальное значение $X_{нач.}$, при этом выполняются операторы, расположенные в *теле цикла* до служебного слова Next (следующий, указывает на окончание цикла, *тела цикла*). Далее у переменной X изменяется значение на величину шага ΔX и выполняется следующая итерация. Итерации цикла выполняются до тех пор, пока переменная цикла $X \leq X_{кон.}$, это в случае увеличения переменной цикла X, и до тех пор пока переменная цикла $X \geq X_{кон.}$ в случае её уменьшения.

Пример 1. (листинг)

```
Sub Example1()
Nstb = 1
s = 0
For Nstr = 2 To 10 Step 2
    s = s + Cells(Nstr, Nstb)
Next Nstr
MsgBox “Ответ s=”& s
End Sub
```

Пояснения к примеру 1. Процедура определяет сумму *s* чисел расположенных в каждой второй строке первого столбца таблицы Excel, включающего в себя десять строк. Предполагается, что соответствующие ячейки таблицы заполнены численными данными. Переменная цикла Nstr (номер строки) изменяется от 2 до 10 с шагом 2. Cells (ячейки) – это один из способов доступа к ячейкам таблиц Excel, позволяет записывать в ячейки и получать данные из ячеек таблицы. Запись Cells(Nstr, Nstb) означает обращение к ячейке таблицы с номером строки ячейки Nstr и с номером её столбца Nstb и передачу её содержимого в процедуру. При каждой итерации цикла переменная *s* увеличивается на численное значение каждой второй ячейки 1-го столбца таблицы Excel.

При организации циклов вход в цикл следует осуществлять только через его начало (заголовок), Нельзя изменять внутри цикла его параметры (X, $X_{нач.}$, $X_{кон.}$, ΔX). Если служебное слово **Step** (шаг) отсутствует, то это означает, что шаг по умолчанию задан равным единице.

Если необходимо осуществить выход из цикла до его завершения, то следует использовать служебное словосочетание **Exit For** , которое обычно располагают в управляющей конструкции **If**, например:

If условие **Then Exit For**

При таком выходе из цикла (до его завершения) значение переменной цикла X останется таким, каким оно было в момент выхода. В случае если цикл отработал до конца, т.е. все итерации цикла были выполнены, то переменная цикла X примет значение равное конечному $X_{кон.}$ плюс единица. Приведённые обстоятельства могут быть использованы при решении различных задач.

Оператор цикла For Each

В этом цикле не используется счётчик итераций цикла. В цикле выполняется столько итераций, сколько имеется элементов в определённой группе, например, такой как коллекция объектов или массив (см. следующие лаб. работы).

Синтаксис оператора:

For Each *элемент* **In** *группа*

тело цикла a

Next [*элемент*]

где *элемент* - переменная, используемая для итерации по всем элементам в группе;

группа - это объект коллекции или массив;

тело цикла - один, несколько или ни одного оператора.

Пример 2. (листинг)

```
Sub Example2()
s = 0: k=0
For Each cl In Selection  'Для каждого элемента cl в
                          'группе Selection
    s = s + cl: k=k+1
Next cl
MsgBox "Ответ s= " & s & " k=" & k
End Sub
```

Пояснения к примеру 2. Процедура определяет сумму и количество всех чисел расположенных в выделенных ячейках таблицы Excel. Предполагается, что перед запуском программы какое-то количество ячеек таблицы Excel заполнено численными данными, и они выделены. Объект Selection обеспечивает программную работу

с выделенными ячейками таблицы, это и есть *группа*. Пользователь может выделить ячейки в *групп*у с помощью мыши, и запустить процедуру. В переменную cl, которая является элементом группы, на каждой итерации, поочерёдно, будут попадать численные значения ячеек выделенной группы. При этом переменная s на каждой итерации будет увеличиваться на соответствующее численное значение переменной cl, а переменная k на единицу. Цикл закончиться, когда через его итерации пройдет каждая из выделенных ячеек. В результате в переменную s будет записана сумма, а в k – количество чисел расположенных в выделенных ячейках таблицы Excel.

Оператор цикла Do

Оператор цикла **Do** (делай) относится к неопределённым циклам, в которых количество итераций заранее неизвестно и зависит от заданных условий. Существует пять разновидностей данной циклической конструкции. При использовании первых двух цикл либо выполняется один раз или несколько раз, либо не выполняется вообще. Рассмотрим синтаксис первой разновидности этой конструкции:

Do While *условие*

тело цикла

Loop

Если *условие* истинно (True), то происходит выполнение операторов расположенных в *теле цикла*, если оно ложно (False), осуществляется переход к оператору, расположенному после служебного слова **Loop**(цикл), указывает на окончание цикла, *тела цикла*, т. е. цикл ни разу не выполняется.

Вторая разновидность конструкции имеет синтаксис:

Do Until *условие*

тело цикла

Loop

Если *условие* не выполняется (False), то происходит выполнение операторов расположенных в *теле цикла*, если оно истинно (True), осуществляется переход к оператору, расположенному после служебного слова **Loop**, т. е. цикл ни разу не выполняется.

В случае использования следующих двух циклических конструкций цикла, цикл будет выполняться хотя бы один раз. Синтаксис третьей конструкции записывается так:

Do

тело цикла

Loop While *условие*

В этом случае операторы тела цикла выполняются до тех пор, пока условие истинно, иначе выполнение цикла заканчивается.

Четвёртая разновидность конструкции имеет синтаксис:

Do

тело цикла

Loop Until *условие*

В этом случае операторы тела цикла выполняются до тех пор, пока условие не истинно, иначе выполнение цикла заканчивается.

Пятая разновидность циклической конструкции имеет следующий вид:

Do

тело цикла

Loop

В данном случае выход из цикла осуществляется с помощью, записываемого в *тела цикла,* условного оператора **if**, например:

if *условие* **Then Exit Do**

Когда необходимо прервать выполнение любой из рассмотренных конструкций оператора цикла **Do** применяется оператор **Exit DO,** как правило, совместно с условным оператором **if.** Происходит завершение цикла без выполнения**,** каких либо операций. Управление передаётся оператору, следующему за концом (служебное слово **Loop**) цикла.

Вложенные циклы

Цикл можно размещать внутри другого цикла. Размещение одной конструкции цикла в другой называется вложением циклов, а сам цикл вложенным. При организации циклов, в том числе и вложенных, необходимо соблюдать следующие правила:

- вход в циклы осуществляется через их заголовки;
- вход во внутренний цикл осуществляется только после входа во внешний;
- параметры циклов не должны изменяться внутри циклов;
- при вложении циклов имена параметров внешнего и внутренних циклов не должны повторяться;
- вложенные циклы не должны пересекаться, т.е. начало и конец внутреннего цикла должны находиться во внешнем цикле;
- допускается делать вложение разных конструкций циклов.

Пример .3. (пример выполнения задания 1)

Написать и отладить процедуру вычисления значений ции $y = \frac{ax}{\sqrt{1+ax^2}}$, для каждого из заданных значений параметра *a* (0,5;1,0;1,5;2,0) при всех заданных значениях аргумента *x* (от 1 до 7 с шагом 0,25), а также процедуру построения средствами VBA

графиков заданной функции. Результаты вывести на лист таблицы Excel.

Процедура вычисления значений функции (листинг)

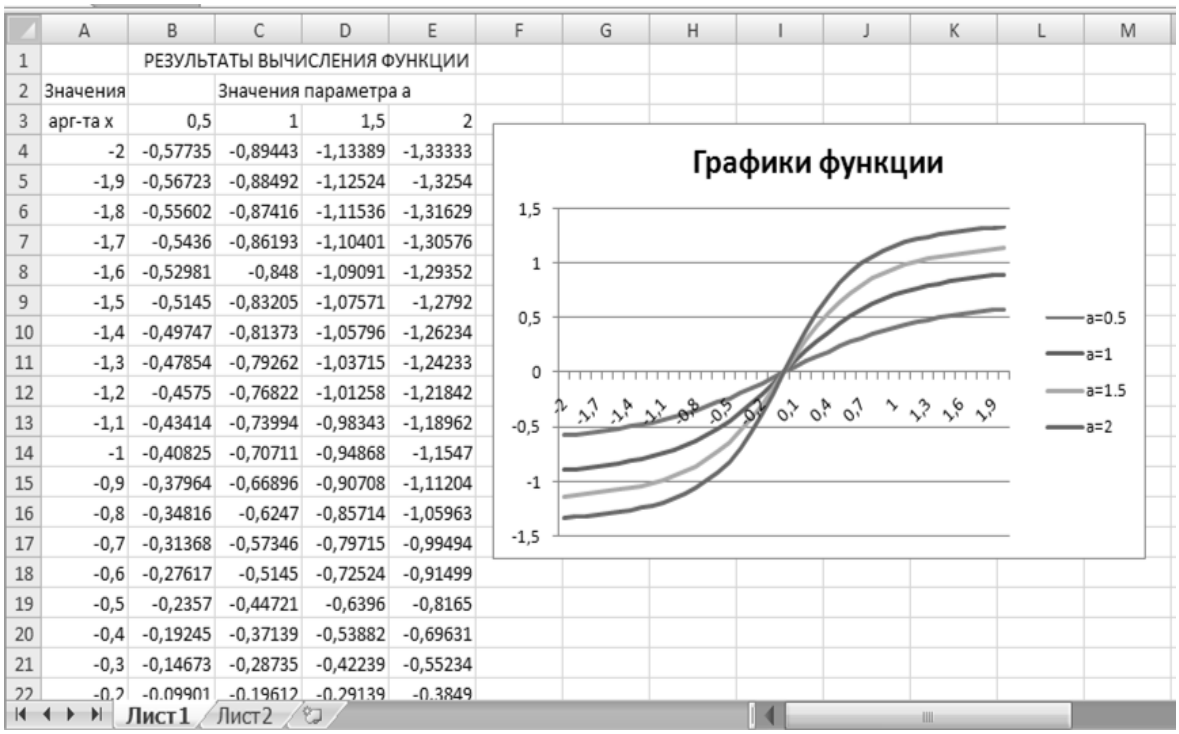
```
Sub FUNC()
Cells(1, 2) = "РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЯ ФУНКЦИИ"
Cells(2, 1) = "Значения": Cells(2, 3) = "Значения параметра a"
Cells(3, 1) = " арг-та x"
k = 2
For a = 0.5 To 2 Step 0.5          'Начало внешнего цикла по a
Cells(3, k) = a
n = 4
For x = -2 To 2.001 Step 0.1       'Начало внутреннего цикла по x
If a = 0.5 Then Cells(n, 1) = x
y = a * x / Sqr(1 + a * x ^ 2)
Cells(n, k) = y
n = n + 1
Next x                             'Конец  внутреннего цикла по x
k = k + 1
Next a                             'Конец внешнего цикла по a
End Sub
```

Процедура построения графиков функции (листинг)

```
 Sub GRAF()
Range("B4:E44").Select   'Задание диапазона значений функции
ActiveSheet.Shapes.AddChart.Select
ActiveChart.SetSourceData Source:=Range("'Лист1'!$B$4:$E$44")
```

```
Selection.Left = 250: Selection.Top = 40
With activeChart
.ChartType = xlLine
.SetElement (msoElementChartTitleAboveChart)
.ChartTitle.Text = "График функции"
.SeriesCollection(1).XValues = "='Лист1'!$A$4:$A$44"
.SeriesCollection(1).Name = "=""a=0.5"""
.SeriesCollection(2).Name = "=""a=1"""
.SeriesCollection(3).Name = "=""a=1.5"""
 .SeriesCollection(4).Name = "=""a=2"""
End With
End Sub
```

Результаты работы процедур



	A	B	C	D	E
1		РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЯ ФУНКЦИИ			
2	Значения		Значения параметра а		
3	арг-та х	0,5	1	1,5	2
4	-2	-0,57735	-0,89443	-1,13389	-1,33333
5	-1,9	-0,56723	-0,88492	-1,12524	-1,3254
6	-1,8	-0,55602	-0,87416	-1,11536	-1,31629
7	-1,7	-0,5436	-0,86193	-1,10401	-1,30576
8	-1,6	-0,52981	-0,848	-1,09091	-1,29352
9	-1,5	-0,5145	-0,83205	-1,07571	-1,2792
10	-1,4	-0,49747	-0,81373	-1,05796	-1,26234
11	-1,3	-0,47854	-0,79262	-1,03715	-1,24233
12	-1,2	-0,4575	-0,76822	-1,01258	-1,21842
13	-1,1	-0,43414	-0,73994	-0,98343	-1,18962
14	-1	-0,40825	-0,70711	-0,94868	-1,1547
15	-0,9	-0,37964	-0,66896	-0,90708	-1,11204
16	-0,8	-0,34816	-0,6247	-0,85714	-1,05963
17	-0,7	-0,31368	-0,57346	-0,79715	-0,99494
18	-0,6	-0,27617	-0,5145	-0,72524	-0,91499
19	-0,5	-0,2357	-0,44721	-0,6396	-0,8165
20	-0,4	-0,19245	-0,37139	-0,53882	-0,69631
21	-0,3	-0,14673	-0,28735	-0,42239	-0,55234
22	-0,2	-0,09901	-0,19612	-0,29139	-0,3849

Рис.4.1

Пояснения к примеру 3. Вещественные числа в двоичной системе ЭВМ представляются приближённо, как результат округления бесконечной периодической дроби. Из-за этого в процедуре FUNC переменная *x* цикла примет конечное значение равное не 2-м, как должно быть по условию задачи, а примерно 2,0000001. Это означает, что последняя итерация при x=2 не будет выполнена. Чтобы обеспечить прохождение последней итерации при x=2, в процедуре конечное значение переменной *x* цикла задано несколько выше, чем 2-а, а именно 2.001.

При построении графиков функции для выделения (выбора) полученного в процедуре FUNC диапазона значений функции B4:E44 используется объект Range (диапазон) и метод (действие, выполняемое над объектом) Select (выбор).

Range-диапазон определяется указанием первой и последней входящих в него ячеек заключённых в ковычки- Range("B4:E44"). Далее через точку указывается метод Select, который позволяет выделить указанный диапазон. Фактически это способ обращения к нужным ячейкам рабочего листа Excel.

Для работы с графиками (диаграммами) используется объект Chart (диаграмма). Чтобы добавить график на лист Excel необходимо его активизировать, используя запись ActiveSheet. Далее используется свойство Shapes (формы) для создания нового объекта и метод AddChart для добавления графика (диаграммы), который с помощью Select выделяется. Это подготовительный этап, самого графика пока нет.

Посредством записи ActiveChart активизируется объект Chart (диаграмма). Далее SetSourceData устанавливает источник данных графика, а Source:= диапазон по которому строится график-Range ("'Лист1'!$B$4:$E$44").

Записи Selection.Left = 250: Selection.Top = 40 позволяют установить расстояния в пунктах соответственно от левой границы поля таблицы Excel до левого края поля графика 250 и от верхней границы поля таблицы до верхнего края поля графика 40.

Далее в процедуре GRAF используется конструкция операторов **With…End With**, которая позволяет ссылаться на свойства и методы данного объекта без указания полной объектной ссылки при каждом обращении к ним. Так указанная после оператора **With** запись ActiveChart, при выполнении процедуры, будет автоматически подставляться впереди всех записей расположенных внутри конструкции **With…End With.** Использование данной конструкции упрощает написание процедуры и делает её более читабельной.

Записи в первых трёх строчках внутри конструкции **With…End With** задают соответственно тип графика, месторасположение текста заголовка к графику и сам текст. Запись в четвёртой строке

.SeriesCollection(1).XValues = "='Лист1'!$A$4:$A$44"

передаёт данные для горизонтальной оси. Последние четыре строки передают элементы легенды (ряды), т.е. соответствующее обозначение для каждого графика.

Суммирование рядов

Выше было рассмотрено использование циклических конструкций для вычисления, накапливания сумм (примеры 1,2). Накапливание сумм используется при вычислении, суммировании рядов (их членов). С помощью рядов могут вычисляться различные функции. Рассмотрим такую задачу на примере.

Пример 4. (пример выполнения задания 2)

Написать и отладить процедуру для приближённого вычисления функции y=sin(x) с помощью ряда $s(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$. Суммирование членов ряда проводить включительно до члена ряда, значение которого по абсолютной величине меньше чем 10^{-6}.

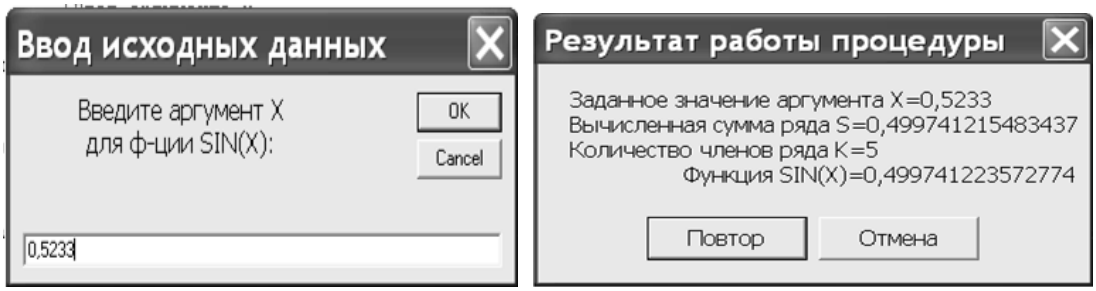
Вычислить сумму членов заданного ряда s(x) для $x = \frac{\pi}{6} \approx 0{,}5233$ и сравнить полученное значение со значением, вычисленным непосредственно с помощью функции y=sin(x).

Процедура (листинг)

```
Sub Example4()
Dim x, y, s, a, p As Double
Dim n As Integer
                    'ВВОД АРГУМЕНТА x
10 x = CSng(InputBox("   Введите аргумент X " _
& vbLf & " для ф-ции SIN(X):"
'ВЫЧИСЛЕНИЕ СУММЫ ЧЛЕНОВ ЗАДАННОГО РЯДА
s = 0: p = 1
k = 1: n = 3
a = x                          'Первый член ряда
Do While (Abs(a) >= 0.000001)
    s = s + a                  'Суммирование членов a ряда
    a = x ^ n                  'Вычисление числителя члена ряда
    p = -p * (n - 1) * n       'Вычисление знаменателя члена ряда
    a = a / p                  'Очередной член a ряда
    n = n + 2: k = k + 1
```

```
Loop
y = Sin(x)  'Непосредственное вычисление заданной функции
                    'ВЫВОД РЕЗУЛЬТАТОВ
Ответ = MsgBox("Заданное значение аргумента X=" & x _
& vbLf & "Вычисленная сумма ряда S=" & s _
& vbLf & "Количество членов ряда K=" & k _
& vbLf & "                Функция SIN(X)=" & y, _
vbRetryCancel, "Результат работы процедуры")
Select Case Ответ
Case vbRetry: GoTo 10
Case vbCancel: GoTo 20
End Select
    20 End Sub
```

Работа процедуры



2.Практическая часть

Задание 1. Составить и отладить процедуру вычисления заданной в таблице 4.1 функции $y(x)$ для каждого из заданных значений параметра a и при всех заданных значениях аргумента x. Результаты вывести в таблицу Excel и построить средствами VBA графики заданной функции.

таблица 4.1

№	Заданная функция $y(x)$	Значение аргумента $x_{нач}$; $x_{кон}$; $Дx$	Значение параметра a
1.	$y=(1/a)\cdot\exp(-(x/a)^2)$	-1,5; 1,5; 0,1	1; 1,1; 1,2; 1,3
2.	$y=x\cdot e^{-x/a}$	0; 4; 0,2	0,25;0,5;0,75;1,0
3.	$y=1/\sqrt{(1-x^2)^2+4a^2x^2}$	0; 2; 0,05	0,1;0,2;0,3; 0,4
4.	$y=x\cdot tga-x^2/\cos^2 a$	0; 0,5; 0,02	15^0;30^0; 45^0;60^0
5.	$y=a\cdot x^a\cdot e^{-x/a}$	0; 10; 0,25	1;1,25;...2,0
6.	$y=e^{-xa}\cdot\sin x$	0; π; π/36	0; 0,5;...2,0
7.	$y=((x+a)^{2/3}-(x-a)^{2/3})/a$	-4; 4; 0,2	1; 2; 3; 4
8.	$y=a^{-x}-a^{-a\cdot x}$	0; 2; 0,05	10; 8; 6;4; 2
9.	$y=(1-\exp(-(x/a)^2))/a$	-2; 2; 0,1	1,25; 1,5; 1,75; 2
10.	$y=a^3/(a^2+x^2)$	-2; 2; 0,05	0,5; 1,0; 1,5;2,0
11.	$y=\frac{\sqrt[4]{(a+1)x}+e^{-x^3}}{\sqrt{2ax}}$	1; 7; 0,25	0,5;0,75;1,0;1,25
12.	$y=\frac{\sqrt[3]{ax^2}+e^{-x^2}}{\sqrt{(a+1)x}}$	1; 7; 0,25	0,5;0,75;1,0;1,25
13.	$y=\frac{\sqrt[3]{x}+ae^{-x^2}}{\sqrt{ax}}$	1; 7; 0,25	0,5;0,75;1,0;1,25
14.	$y=(\sqrt{x}+ae^{-x^2})/(ax^2)$	1; 7; 0,25	0,5;0,75;1,0;1,25

15.	$y = \frac{arctg(\frac{x^2}{2a})}{x^3 + a}$	0; 2,5; 0,1	0,5; 1,0; 1,5; 2,0
16.	$y = \frac{arctg(\frac{x}{2a})}{x^2 + 2a}$	0; 3; 0,1	0,5;0,75;1; 1,25
17.	$y = \frac{ax}{a + \sqrt[3]{1 + x^2}}$	-3; 3; 0,2	0,5; 1,0; 1,5, 2,0
18.	$y = Cos^2(2ax)/(3a)$	0^0; 360^0; 6^0	1; 1,25; 1,5, 2,0
19.	$y = Sin^2(ax)/(a+2)$	0^0; 360^0; 6^0	1; 1,25; 1,5; 2,0
20.	$y = aCos(2x)/(a+4)$	0^0; 360^0; 6^0	1; 2; 3; 4
21.	$y = a^2 e^{-x}/(2 + a^2)$	-4; 4; 0,25	1; 2; 3; 4
22.	$y = \frac{\sqrt[3]{ax^2} + e^{-x}}{ax^2}$	1; 7; 0,25	0,5; 1,0; 1,5; 2,0
23.	$y = \frac{\sqrt{x} + a^2 e^{-x^2}}{a + x}$	1; 7; 0,25	0,5; 1,0; 1,5;2,0
24.	$y = (x^2 + a)^{\frac{xa}{x-1}}$	0; 0,8; 0,05	0,5; 1,0; 1,5, 2,0
25.	$y = (x + 4a)^{\frac{ax}{x-1}}$	0; 0,8; 0,05	0,5; 1,0; 1,5;2,0
26.	$y = \ln^2\left\|\frac{xa}{a+x}\right\|$	2; 12; 0,5	0,5; 1,0; 1,5; 2,0
27.	$y = \ln\left\|\frac{ax}{1+ax}\right\|$	2; 12; 0,5	0,5; 1,0; 1,5; 2,0
28.	$y = -\ln\left\|\frac{x+a}{1+x^2}\right\|$	1; 10; 0,5	1; 2; 3; 4
29.	$y = \frac{ax^2}{\sqrt{1 + ax^2}}$	-2; 2; 0,1	1; 2; 3; 4
30.	$y = -\ln\left\|\frac{ax}{a+x}\right\|$	1; 11; 0,5	1; 2; 3; 4

Задание 2. Составить и отладить процедуру для приближённого вычисления заданной функции y=f(x) путём суммирования членов заданного её ряда s(x) (см. таблицу 4.2). Суммирование членов ряда проводить включительно до члена ряда, значение которого по абсолютной величине будет меньше чем 10^{-6}.

Вычислить значения суммы ряда s(x) при указанных в таблице 4.2 контрольных данных аргумента x и сравнить полученные значения со значениями, вычисленными непосредственно с помощью функции y= f(x).

таблица 4.2

№	Ряд	Контрольные данные: аргумент x, функция y(x)
1	$S=1+\frac{x}{2}-\frac{1\cdot x^2}{2\cdot 4}+\frac{1\cdot 3\cdot x^3}{2\cdot 4\cdot 6}-\frac{1\cdot 3\cdot 5\cdot x^4}{2\cdot 4\cdot 6\cdot 8}+...$	-0.84; 1; 2; $y=\sqrt{x+1}$
2	$S=1+\frac{x}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+...$	1; 10; -10; $y=e^x$
3	$S=\sum_{n=1}^{\infty}(-1)^{n-1}\frac{2^{2n-1}}{(2n)!}x^{2n}$	π/6; 13π/6; 25π/6 ; $y=\sin^2 x$
4	$S=1+\sum_{n=1}^{\infty}(-1)^n\frac{-1\cdot 1\cdot 3\cdot 5\cdot ...\cdot(2n-3)}{n!\cdot 2^{3n}}(x-4)^n$	2; 1; 0; $y=0.5\cdot\sqrt{x}$
5	$S=\frac{x}{2}+\frac{x^2}{2^2}+\frac{x^3}{2^3}+...$	-1; 1; 1.9; $y=\frac{x}{2-x}$
6	$S=\frac{x}{1}+\frac{1\cdot x^3}{2\cdot 3}+\frac{1\cdot 3\cdot x^5}{2\cdot 4\cdot 5}+\frac{1\cdot 3\cdot 5\cdot x^7}{2\cdot 4\cdot 6\cdot 7}+...$	0.5; $\sqrt{2}/2$; -1; $y=\arcsin x$
7	$S=\frac{x}{1}-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+...$	$1/\sqrt{3}$; $-1/\sqrt{3}$; 1; $y=arctgx$
8	$S=1+\sum_{n=1}^{\infty}\frac{(n^2+1)\cdot x^n}{n!\cdot 2^n}$	2; 20; -15; $y=(z^2+z+1)e^x$, где $z=x/2$
9	$S=\sum_{n=1}^{\infty}\sin(\pi n/4)\frac{\sqrt{2^n}}{n!}x^n$	π/4; π; -5.5π; $y=e^x\sin x$

	*см. сноску	
10	$S = x + \sum_{n=1}^{\infty} (-1)^n \frac{2^{2n} \cdot x^{2n+1}}{(2n)!}$	π/6; π; 4π; $y = x \cdot \cos 2x$
11	$S = 2 \cdot \left(\frac{x}{1} + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \ldots \right)$	0.5; 0.9; 0.99; $y = \ln \frac{1+x}{1-x}$
12	$S = e^{-2} \left(1 + \sum_{n=1}^{\infty} \frac{(x+2)^n}{n!} \right)$	1; 9; -15; $y = e^x$
13	$S = 1 + \frac{2 \cdot x}{1!} + \frac{3 \cdot x^2}{2!} + \frac{4 \cdot x^3}{3!} + \ldots$	1; 9; -11; $y = (1+x) \cdot e^x$
14	$S = \frac{2 \cdot x}{1} - \frac{2^2 x^2}{2} + \frac{2^3 \cdot x^3}{3} - \ldots$	1/4; -1/3; 1/2; $y = \ln(1+2x)$
15	$S = 3x + 8x^2 + \ldots + n(n+2)x^n + \ldots$	0.2; 0.6; 0.9; $y = \frac{x(3-x)}{(1-x)^3}$
16	$S = 1 + \frac{x \ln 2}{1!} + \frac{x^2 \ln^2 2}{2!} + \frac{x^3 \ln^3 2}{3!} + \ldots$	2; 10; -10; $y = 2^x$
17	$S = 1 - \frac{2^2 x^2}{2!} + \frac{2^4 x^4}{4!} - \frac{2^6 x^6}{6!} + \ldots$	π/6; π; 5π; $y = \cos 2x$
18	$S = 1 + \sum_{n=1}^{\infty} \frac{\cos(n\pi/4)}{n!} x^n$ *см. сноску	π/6; π; -5π; $y = e^z \cos z$ где $z = x/\sqrt{2}$
19	$S = 1 - 2x + 3x^2 - 4x^3 + \ldots$	0.2; 0.6; 0.9; $y = 1/(1+x)^2$
20	$S = \sum_{n=1}^{\infty} x^n \cdot \sin(n\pi/4)$ *см. сноску	0.5; 0.8; 0.99; $y = (x/\sqrt{2})/(1 - \sqrt{2} \cdot x + x^2)$
21	$S = 1 + \frac{\cos(x)}{1!} + \frac{\cos(2x)}{2!} + \ldots + \frac{\cos(nx)}{n!} + \ldots$ *см. сноску	π/6; -π; 10π; $y = e^{\cos x} \cos(\sin x)$

22	$S = -\frac{(2x)^2}{2!} + \frac{(2x)^4}{4!} - \ldots (-1)^n \frac{(2x)^{2n}}{(2n)!} + \ldots$	$\pi/6;\ 13\pi/6;\ 25\pi/6;$ $y = 2(\cos^2 x - 1)$
23	$S = \sum_{n=0}^{\infty} \frac{1}{2n+1} \cdot \left(\frac{x-1}{x+1}\right)^{2n+1}$	$0.5;\ 0.1;\ 0.01;$ $y = 0.5 \cdot \ln(x)$
24	$S = \frac{1}{\sqrt{2}} + \sum_{n=1}^{\infty} \cos(\frac{2n+1}{4} \cdot \pi) \cdot \frac{(x-\pi/2)^n}{n! \cdot 2^n}$	$2\pi/3;\ 8\pi/3;\ 28\pi/3;$ $y = \cos(x/2)$
25	$S = x + \sum_{n=1}^{\infty} (-1)^n \frac{1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-1) \cdot x^{2n+1}}{2^n \cdot n! \cdot (2n+1)}$	$0.5;\ 0.9;\ 1;$ $y = \ln(x + \sqrt{1+x^2})$
26	$S = x + 2 \cdot \left(\frac{x^3}{1 \cdot 3} - \frac{x^5}{3 \cdot 5} + \frac{x^7}{5 \cdot 7} - \frac{x^9}{7 \cdot 9} + \ldots\right)$	$1/\sqrt{3}; -1/\sqrt{3};\ 1$ $y = (1+x^2) \cdot arctg(x);$
27	$S = \frac{\sqrt{3}}{2} + \sum_{n=1}^{\infty} \sin(\pi/3 + n\pi/2) \cdot \frac{\pi^n (x-1)^n}{3^n \cdot n!}$	$0.5;\ 6.5;\ 13.5;$ $y = \sin(\pi x/3)$
28	$S = \sum_{n=1}^{\infty} (-1)^{n-1} (1+2^n) \frac{x^n}{n}$	$0.25;\ 0.45;\ 0.5;$ $y = \ln(1+3x+2x^2)$
29	$S = -1 + \frac{x+1}{3} + \sum_{n=2}^{\infty} \frac{2 \cdot 5 \cdot 8 \cdot \ldots \cdot (3n-4)}{3^n n!} (x+1)^n$	$-1.331;\ -0.729;\ 0;$ $y = \sqrt[3]{x}$
30	$S = 4 + \sum_{n=1}^{\infty} (2+2^n) \frac{x^n}{n!}$	$2;\ -1;\ -10;$ $y = (1+e^x)^2$

*) для рядов, отмеченных звёздочкой, при оценке погрешности в членах ряда не учитывать синусы и косинусы.

ЛАБОРАТОРНАЯ РАБОТА № 5
Обработка одномерных массивов

1.Краткие теоретические сведения

Массив – это множество однотипных элементов. Каждый массив имеет уникальное собственное имя. Каждый элемент данных, хранящийся в массиве, называется элементом массива. Для доступа к определенному элементу любого массива необходимо указать имя массива и некоторое число, которое называется индексом элемента массива.

Массивы в программе должны быть определены с помощью оператора **Dim**, синтаксис которого имеет следующий вид

Dim *VarName([Subscripts]) [***As** *Type]*

Varname – имя существующего массива;
Subscripts – измерение (или измерения) массива;
Type – любой допустимый тип VBA или тип данных, определенный пользователем.

VBA представляет программисту возможность объявлять массивы, имеющие до 60 измерений. Одномерному массиву соответствует один *Subscripts* (индекс); двумерному массиву – два индекса, разделенные запятой. Каждый индекс увеличивает размерность массива на единицу. Если опустить *Type*, все элементы массива будут иметь тип **Variant**.

Следующая инструкция

Dim Array(30) **As String**

объявляет массив Array, содержащий 30 элементов. По умолчанию, индексация массивов начинается с нуля и поэтому индекс последнего элемента массива – 29, а не – 30.

Операторы

```
Option Base 0                    ' установка по умолчанию;
                                 'индексы массивов начинаются с 0
Option Base 1                    ' индексы массивов начинаются с 1
```

позволяют задавать 0 или 1 число по умолчанию для индексов массива. Следует отметить, что только верхний предел массива является обязательным. В этом случае нумерация массива происходит в зависимости от установки Option Base.

```
Dim StrArray(1 To 100) As String

Dim NumArray(-50 To 50) As Integer

Dim ArrByte(0 To 30) As Byte
```

Этот оператор эквивалентен оператору

```
Dim ArrByte(30) As Byte
```

Существуют два вида массивов – статические и динамические. Статическими называются массивы, размерность которых известна заранее и может быть указана при их объявлении. Однако, весьма часто возникают ситуации, когда размерность массива величина неизвестная и в этом случае VBA разрешает при объявлении массива не указывать размерность, а просто определить или переопределить ее во время исполнения процедуры.

ReDim *Varname(Subscripts) [***As** *Type]*

Синтаксис для оператора **ReDim** такой же как для оператора Dim. Включение оператора *Subscripts* в объявление массива создает статический массив. Следовательно, при создании динамического массива оператор *Subscripts* использовать нельзя, хотя круглые скобки должны присутствовать обязательно.

Оператор *Subscripts* имеет следующий синтаксис

[lower **To***] upper [,[lower* **To***] upper] . . .*

lower − нижний предел допустимых индексов массива *Varname*;

upper − верхний предел (обязательный параметр);

Примеры:

```
Dim A() As String                  ' объявляет динамический массив A
Redim A(1 To 25)                   ' изменяет размер массива до 25эл.
Dim A(50)                          ' изменяет размер массива до 50 эл.
Dim Day() As Integer               ' объявляет динамический массив Day
Redim Preserve Day(1 To 20)        ' изменяет размер массива до 20 эл.
                                   ' сохраняя содержимое
Dim B As Variant                   ' объявляет переменную типа Variant
Redim B(50) As Byte                ' создает массив из 50 переменных
                                   'типа Byte
                                   ' внутри переменной типа Variant
```

Размерность массива может быть записана в виде арифметического выражения, значение которого должно быть положительным числом или равно нулю. Если арифметическое выражение имеет дробное значение, то оно округляется, и целая часть определяет размерность массива.

Ввод-вывод массивов

Заполнить массив, т. е. определить значения элементов массива можно следующими способами:

при помощи оператора присваивания;

непосредственным вводом с клавиатуры;

подготовкой и вводом данных из текстового файла;

использования датчика случайных чисел;

заполнением массива при помощи стандартных функций

Пример 1. Заполнить одномерный массив непосредственным вводом с клавиатуры. Результат поместить в столбец С рабочей книги Excel.

```
Sub InputArray1()
'*********************************************************
'  Ввод с клавиатуры
'*********************************************************
   Dim A(10) As Integer
   Dim Title As String
   Worksheets("Лист1").Cells(1, 1) = "Массив А:"
   Title = "Заполнение одномерного массива"
   For i = 1 To 10
      A(i) = InputBox("Введите A(" & i & "):", Title)
      Worksheets("Лист1").Cells(i + 1, 1) = A(i)
   Next i
End Sub
```

Объявляем статический массив А, содержащий 10 элементов. Метод Cells используется для ссылки на одну ячейку. Доступ к

ячейке можно также получить, используя метод Range, однако, метод Cells в этом случае является более удобным. Так Cells(1, 1), например, указывает на ячейку A1 рабочей книги Excel, В эту ячейку и будет записана строка ″Массив А:″.

Синтаксис метода Cells:

Object.Cells(Rowlndex, Columnlndex)

Object – *ссылка или на* ***Worksheet****, или на объект* ***Range****, содержащий ячейку, с которой необходимо работать. Если Object опущен, метод применяется к объекту* ***ActiveSheet.***

Rowlndex – *номер строки, в которой находится ячейка. Если Object – лист, то Rowlndex, равный 1, ссылается на строку 1 листа. Если Object – диапазон значений, то Rowlndex, равный 1, ссылается на первую строку диапазона.*

Columnlndex – *номер столбца (буква или число, задающее столбец), в котором расположена ячейка. Если Object – лист, то Columnlndex, равный "A" или 1, ссылается на столбец А листа. Если Object – диапазон значений, то Columnlndex, равный "A" или 1, ссылается на первый столбец диапазона.*

Следующая строка предлагает увеличить ширину столбца С для того, чтобы строка, которая была записана в ячейку, не выходила за пределы ячейки С, а полностью находилась в ней. Подбор ширины столбца происходит методом проб и ошибок. Далее в цикле **For…Next**, с использованием функции InputBox, происходит заполнение массива А. Данный прием является очень удобным средством организации взаимодействия между пользователем и программой.

Функция InputBox *имеет следующий синтаксис:*

InputBox *(Prompt [, Title])*

Prompt – *любое строковое выражение (обязательный аргуент).* InputBox *отображает эту строку как запрос в диалоговом окне;*

Title – *это строка, используемая в качестве заголовка для окна ввода.* InputBox *отображает текст этой строки в строке заголовка диа- логового окна.*

Пример 2. Заполнить одномерный массив, считав данные из файла File1.txt. Результат поместить в файл File1.txt..

```
Sub InputArray2()
'************************************************
'  запись и считывание данных из файла
'************************************************
   Dim A(10) As Variant
   Dim StrNum As String
   Open "Z:\VBA\File1.txt" For Input As #1
   Open "Z:\VBA\File2.txt" For Output As #2
   For i = 1 To 10
      Input #1, StrNum
      A(i) = StrNum
   Next i
   For i = 1 To 10
      StrNum = A(i) & Chr(13)
      Print #2, StrNum
   Next i
```

```
    Close
End Sub
```

Объявляем массив A(10) и строковую переменную StrNum. Открываем File1.txt для считывания и File2.txt для записи. Далее в цикле **Do…Loop** считываем построчно из файла File1.txt и помещаем результат в строковую переменную StrNum. Заполняем массив при помощи операции присваивания. Последний этап – запись массива A в файл File2.txt. Функция EOF(filenumber) возвращает значение True при достижении конца файла открытого для последовательного доступа. Аргумент filenumber содержит допустимый номер файла. Оператор **Close** закрывает все открытые файлы. Если этого не сделать, то при повторном запуске процедуры, компилятор сообщит о наличии ошибки.

Пример 3. Заполнить одномерный массив с помощью датчика случайных чисел. Результат поместить в файл File3.txt.

```
Sub InputArray3()
'*****************************************************
'  Заполнение одномерного массива при   помощи датчика
'случайных чисел
'*****************************************************
    Dim A() As Integer
    Dim Title As String
    Open "Z:\VBA\File3.txt" For Output As #3
    Randomize
    Title = "Заполнение одномерного массива"
    N = InputBox("Размер массива", Title)
    ReDim A(1 To N)
```

```
    For i = 1 To N
        ArrayValue = Int((100 * Rnd) + 1)
        A(i) = ArrayValue
        Worksheets("Лист1").Cells(i + 1, 3) = A(i)
        Print #3, A(i)
    Next i
    Close
End Sub
```

Объявляем массив A целых чисел, причем в данном случае необходимо создать динамический массив. Открываем File3.txt для записи. Далее с помощью оператора Randomize инициализируем функцию Rnd, которая генерирует случайные числа в диапазоне от 0 до 1. Оператор Int((100 * Rnd) + 1) заполняет массив случайными числами в интервале от 0 до 100. Полученный файл записывается в File3.txt и на Лист1 рабочей книги.

Представление данных в виде массивов очень удобный, а иногда и единственно возможный метод, который использует программист для решения тех или иных практических задач. Поэтому далее рассматриваются некоторые приемы, которые необходимо знать и уметь применять при обработке данных, представленных в матричном виде.

Пример 4. Вычислить сумму и среднее арифметическое всех элементов одномерного массива, состоящего из 20-ти элементов.

```
Sub InputArray4()
'*************************************************************
'  Вычислить сумму и среднее арифметическое всех   элементов
'одномерного массива, состоящего   из 20-ти элементов.
'  Заполнение одномерного массива происходит при   помощи
'датчика случайных чисел
'*************************************************************
   Dim A(1 To 20) As Integer
   Dim Title As String
   Dim String1 As String
   Dim String2 As String
   String1 = "Сумма = "
   String2 = "Среднее арифметическое = "
   Randomize
   Title = "Сумма элементов одномерного массива"
   Sum = 0
   For i = 1 To 20
      ArrayValue = Int((100 * Rnd) + 1)
      A(i) = ArrayValue
   Next i
   For i = 1 To 20
      Sum = Sum + A(i)
   Next i
```

```
    mA = Sum / 20
    MsgBox String1 & Sum & Chr(13) & String2 & mA, , Title
End Sub
```

Первый цикл **For…Next** используется для заполнения массива A случайными числами в диапазоне от 0 до 100. Вычисление суммы Sum элементов массива A происходит во втором цикле **For…Next**. Переменная этого цикла одновременно является индексом массива. На каждом шаге цикла к сумме элементов массива добавляется очередной элемент. По окончании цикла переменная Sum будет содержать полную сумму всех элементов массива A(i).

Пример 5. Вычислить сумму всех четных элементов одномерного массива, состоящего из 15-ти элементов.

```
Sub InputArray5()
'*************************************************************
'Вычислить сумму всех четных элементов   одномерного массива,
'состоящего из 15-ти элементов.  Заполнение одномерного
'массива происходит при помощи датчика случайных чисел.
'*************************************************************
    Dim A(1 To 15) As Integer
    Dim Title As String
    Dim String1 As String
    String1 = "Сумма = "
    Randomize
    Title = "Сумма четных элементов одномерного массива"
    Sum = 0
```

```
    For i = 1 To 15
        ArrayValue = Int((100 * Rnd) + 1)
        A(i) = ArrayValue
    Next i
    For i = 1 To 15
        If (Int(A(i) / 2) * 2 - A(i)) = 0 Then
            Sum = Sum + A(i)
        End If
    Next i
    MsgBox String1 & Sum, , Title
End Sub
```

Основное отличие данного примера от предыдущего заключается в том, что в переменной Sum накапливается сумма четных элементов массива. Для определения четных элементов используется выражение Int(A(i) / 2) * 2 - A(i), которое равно 0, если элемент четный и 1, если элемент нечетный.

Пример 6. Определить индексы второго положительного и третьего отрицательного элементов одномерного массива, состоящего из 20-ти элементов.

```
Sub InputArray6()
'*************************************************************
'  Определить индексы второго положительного и  третьего
'отрицательного элементов одномерного массива, состоящего из
'20-ти элементов.  Заполнение одномерного массива происходит
'при  помощи датчика случайных чисел.
'*************************************************************
   Dim A(1 To 20)
   Dim pA, mpA, m, mmA As Integer
   Dim Title As String
   Dim String1 As String
   Dim String2 As String
   Dim String3 As String
   Randomize
   Title = "Индексы элементов одномерного массива"
   String1 = "Второе положительное число = "
   String2 = "Третье отрицательное число = "
   String3 = ",   индекс = "
   For i = 1 To 20
      ArrayValue = 100 - Int((200 * Rnd) + 1)
      A(i) = ArrayValue
   Next i
   i = 0
   jNum = 0
   Do
```

```
    i = i + 1
    If A(i) > 0 Then jNum = jNum + 1
    If jNum = 2 Then Exit Do
  Loop While i < 20
  pA = A(i)
  mpA = i
  i = 0
  jNum = 0
  Do
    i = i + 1
    If A(i) < 0 Then jNum = jNum + 1
    If jNum = 3 Then Exit Do
  Loop While i < 20
  mA = A(i)
  mmA = i
  MsgBox String1 & pA & String3 & mpA & Chr(13) & _
      String2 & mA & String3 & mmA, , Title
End Sub
```

Сначала в цикле **For…Next** происходит заполнение массива A случайными числами, находящимися в диапазоне от -100 до +100. Далее в цикле **Do…Loop** необходимо определить является ли очередной элемент массива положительным или нет. Если элемент – положительный, то переменная jNum, которая определяет число найденных положительных элементов увеличивается на 1. При выполнении условия jNum = 2 происходит выход из цикла и сохранение в переменных pA и mpA второго положительного элемента

массива и его индекса. Аналогично происходит поиск третьего отрицательного элемента массива и его индекса.

Пример 7. Задан одномерный массив размером N. Сформировать два массива, включая в первый массив четные элементы, а во второй − нечетные элементы исходного массива. Отсортировать полученные массивы в порядке возрастания и вывести их во второй и третий столбец второго листа рабочей книги.

```
Sub InputArray7()
'*********************************************************
' Задан одномерный массив размером N. Сформировать два
'массива, включая в первый массив четные элементы, а во второй
'нечетные элементы исходного массива. Отсортировать
'полученные массивы в порядке возрастания и вывести и во
'второй и третий столбец второго листа рабочей книги.
'*********************************************************
   Dim A() As Integer
   Dim B() As Integer
   Dim C() As Integer
   Randomize
   N = InputBox("Размер массива", "Заполнение одномерного массива")
   ReDim A(1 To N)
   For i = 1 To N
      ArrayValue = Int((100 * Rnd) + 1)
      A(i) = ArrayValue
      Worksheets("Лист2").Cells(i + 1, 1) = A(i)
```

```
    Next i
    ReDim B(1 To N)
    ReDim C(1 To N)
    Worksheets("Лист2").Cells(1, 1) = " A "
    Worksheets("Лист2").Cells(1, 2) = " B "
    Worksheets("Лист2").Cells(1, 3) = " C "
'   Заполнение массивов B и C
    jB = 0
    jC = 0
    For i = 1 To N
       TestValue = A(i) - Int(A(i) / 2) * 2
       Select Case TestValue
          Case 0
             jB = jB + 1
             B(jB) = A(i)
          Case 1
             jC = jC + 1
             C(jC) = A(i)
       End Select
    Next i
'   Сортировка массива B
    For i = 1 To jB - 1
       For j = 1 To jB - 1
          If B(j + 1) < B(j) Then
```

```
            temp = B(j)
            B(j) = B(j + 1)
            B(j + 1) = temp
        End If
    Next j
Next i
' Сортировка массива C
For j = 1 To jB
    Worksheets("Лист2").Cells(j + 1, 2) = B(j)
Next j
For i = 1 To jC - 1
    For j = 1 To jC - 1
        If C(j + 1) < C(j) Then
            temp = C(j)
            C(j) = C(j + 1)
            C(j + 1) = temp
        End If
    Next j
Next i
For j = 1 To jC
    Worksheets("Лист2").Cells(j + 1, 3) = C(j)
Next j
End Sub
```

Как и в предыдущих примерах, в первом цикле **For…Next** происходит заполнение массива A случайными числами, а далее во

втором цикле в массив B записываются четные числа, а в массив C – нечетные числа. После этого начинается процесс сортировки массива B. Сравниваются два соседних элемента массива и если предыдущий элемент больше последующего, то производится перестановка этих элементов, которая осуществляется с использованием вспомогательной переменной temp:

```
temp = B(j)
B(j) = B(j + 1)
B(j + 1) = temp
```

Аналогично происходит сортировка массива C. Массивы A, B и C выводятся соответственно в колонки A, B и C Листа 2 рабочей книги Excel. Для решения поставленной задачи используются динамические массивы.

2.Практическая часть

Задание к лабораторной работе

Для заданного условия составить процедуру и придумать несколько наборов тестовых данных для отладки. Возможно использование как статических массивов, так и динамических. Ввод исходных данных осуществить из файла или с клавиатуры. Предусмотреть вывод результата в файл или на лист рабочей книги Excel.

1. Задан одномерный массив размером N. Определить количество отрицательных чисел, количество положительных чисел и среднее арифметическое всех чисел.
2. Задан одномерный массив размером N. Сформировать два массива размером $N/2$, включая в первый элементы исходного массива с четными индексами, а во второй − с нечетными. Вычислить суммы элементов каждого из массивов.

3. Определить среднее арифметическое значение первого отрицательного и последнего положительного элементов одномерного массива размером *M*
4. Определить число отрицательных элементов, расположенных перед наибольшим положительным элементом одномерного массива размером *M*
5. В заданном одномерном массиве размером *N* поменять местами первый и последний положительные элементы.
6. Написать процедуру для вычисления суммы и среднего арифметического значения всех элементов заданного одномерного массива A, состоящего из 10-ти элементов.
7. Написать процедуру для вычисления суммы положительных элементов, заданного массива A, состоящего из 20-ти элементов.
8. Написать процедуру для вычисления суммы четных элементов заданного массива A, состоящего из 20-ти элементов.
9. Написать процедуру для определения количества положительных элементов, заданного массива A, состоящего из 20-ти элементов.
10. Написать процедуру для определения индексов положительных элементов, заданного массива A, состоящего из 20-ти элементов.
11. Написать процедуру для вычисления среднего арифметического значения всех элементов заданного массива D. Для отрицательных элементов использовать их абсолютные значения.
12. Написать процедуру для поиска в заданном массиве B, состоящем из 10-ти элементов, третьего положительного эле-

мента и его индекса. Известно, что хотя бы один положительный элемент в массиве B имеется.

13. Написать процедуру поиска в заданном массиве B, состоящем из 20-ти элементов, третьего положительного элемента и его индекса.
14. Написать процедуру для поиска в заданном массиве A(15) наибольшего значения элемента и его индекса.
15. Написать процедуру, в которой производится перестановка четных и нечетных элементов, заданного массива C.
16. Для заданного массива A, состоящего не более чем из 50-ти элементов, найти наименьший элемент и переставить его со вторым по порядку отрицательным элементом массива.
17. Написать процедуру по упорядочению элементов заданного массива B в следующем порядке: сначала идут положительные числа, потом – нули и в конце – отрицательные.
18. Написать процедуру сортировки по возрастанию заданного массива B, состоящего из 10-ти элементов.
19. Написать процедуру. Для заданного массива B, состоящего из 10-ти элементов, изменить порядок следования его элементов на обратный.
20. Написать процедуру, в которой для заданного массива B, состоящего из 10-ти элементов, его элементы перемещались бы например на 7 позиций вправо. При этом 7 элементов из конца массива перемещаются в начало.
21. Задан массив A. Поместить положительные элементы этого массива в массив B, а отрицательные элементы – в массив C
22. В заданном векторе (одномерном массиве) найти сумму первого и последнего отрицательных элементов

23. В заданном векторе (одномерном массиве) найти: разность первого и последнего нечетного элементов
24. В заданном векторе (одномерном массиве) найти: индексы наименьшего и наибольшего из элементов
25. В заданном векторе (одномерном массиве) найти: произведение трех наименьших элементов вектора
26. В заданном векторе (одномерном массиве) найти: сумму элементов вектора с четными индексами
27. В заданном векторе (одномерном массиве) найти: разность первого положительного и последнего отрицательного элемента
28. В заданном векторе (одномерном массиве) найти: число отрицательных элементов с нечетными индексами
29. В заданном векторе (одномерном массиве) найти: число элементов, расположенных после его наибольшего отрицательного элемента.
30. В заданном векторе (одномерном массиве) найти: наибольший отрицательный и наименьший положительные элементы.

ЛАБОРАТОРНАЯ РАБОТА № 6
Обработка двумерных массивов

1.Краткие теоретические сведения

Любой ***массив*** (одномерный, двумерный или даже многомерный) представляет собой просто ***набор*** (или ***коллекцию***) однотипных данных, которые называются элементами массива. При обращении к элементу указывается имя массива и индекс (номер элемента последовательности), если массив одномерный. Положение элемента двумерного массива определяют два индекса, которые задают номер строки и номер столбца двумерной таблицы, на пересечении которых расположен элемент.

Прежде чем использовать массив, его следует описать. При описании статического массива необходимо определить нижнюю и верхнюю границы каждого индекса, причем эти границы не могут быть изменены внутри программы.

Примеры:

```
Dim arrA(l To 10, 3 To 12) As Integer   'Переменная arrA представляет
                                        'собой  двумерный массив с тип-
                                        'ом данных каждого элемента
                                        'Integer. Первый элемент масси-
                                        'ва – arrA(1, 3), последний
                                        'arrA(10, 12), число элементов
                                        'массива 10*10=100
```

```
Dim strB(-10 To 10, 25) As String     ' Переменная strB представляет
                                      ' собой двумерный массив строк.
                                      'Так как в  скобках указано
                                      'только одно целое  число, то
                                      'это – верхняя граница. Если
                                      'инструкция Option Base
                                      'отсуствует, то первый 'эле-
                                      'мент массива – arrA(-10, 0), по-
                                      'следний – arrA(10,25), число
                                      'элементов массива
                                      ' 21*26=546
```

Обработка двумерных массивов

Главное отличие двумерного массива от одномерного заключается в следующем: для обработки двумерного массива необходимо сформировать все возможные сочетания численных значений двух индексов. Следовательно, это можно сделать, если при обработке двумерных массивов использовать два цикла, один из которых реализуется по индексам строк, а второй – по индексам столбцов. В зависимости от того какой из этих циклов будет внешним, а какой – внутренним, возможны две схемы обработки массивов.

```
' обработка массива A(M, N) по
' строкам
Option Base 1
For i = 1 To M
   For j = 1 To N
         A(i, j) = . . .
   Next j
Next i
```

```
' обработка массива A(M, N) по
' столбцам
Option Base 1
For j = 1 To N
   For i = 1 To M
         A(i, j) = . . .
   Next i
Next j
```

Следует отметить, что хотя указание переменной цикла в операторе **Next** является необязательной опцией, желательно привыкнуть к такому стилю программирования. Это позволит избежать ненужных ошибок в дальнейшей работе.

Ввод двумерных массивов

Как и в случае одномерных массивов, ввод двумерных массивов можно осуществить аналогичными методами:

при помощи оператора присваивания;

непосредственным вводом с клавиатуры;

подготовкой и вводом данных из текстового файла;

использования датчика случайых чисел;

заполнением массива при помощи стандартных функций

Пример 1. Заполнить одномерный массив непосредственным вводом с клавиатуры. Результат поместить в столбцы A - D рабочей книги Excel.

```
Option Base 1
Sub InputArray1()
'*************************************************************
'  Ввод с клавиатуры
'*************************************************************
   Dim A(3, 5) As Integer
   Dim Title As String
   Title = "Заполнение двумерного массива"
   For i = 1 To 3
      For j = 1 To 5
         A(i, j) = InputBox("Введите A(" & i & ", " & j & "):", Title)
```

```
        Worksheets("Лист1").Cells(i + 1, j) = A(i, j)
    Next j
  Next i
End Sub
```

Процедура достаточно проста и отличается от случая одномерного массива только наличием двух циклов по i и по j. Сначала происходит ввод по столбцам и заполняется первая строка, затем – вторая и, наконец, заполняется третья (последняя) строка.

При использовании оператора **Option Base** его следует помещать в область объявлений модуля перед объявлениями любых переменных, констант или процедур. Нельзя помещать **Option Base** внутри процедуры и, кроме того, в модуле должен быть только один оператор **Option Base**.

Пример 2. Заполнить одномерный массив, считав данные из файла File1.txt. Результат поместить в файл File2.txt.

```
Sub InputArray2()
'*****************************************************
'  запись и считывание данных из файла
'*****************************************************
  Dim A(10, 10) As Variant
  Dim StrNum As String
  Dim endStr As String
  Open "Z:\VBA\File4.txt" For Input As #1
  Open "Z:\VBA\File5.txt" For Output As #2
  For i = 1 To 10
    For j = 1 To 10
```

```
        Input #1, StrNum
            A(i, j) = StrNum
        Next j
    Next i
    For i = 1 To 10
        StrNum = ""
        For j = 1 To 10
            endStr = ", "
            If j = 10 Then endStr = Chr(13)
            StrNum = StrNum & A(i, j) & endStr
        Next j
        Print #2, StrNum
    Next i
    Close
End Sub
```

Данный пример аналогичен примеру, который был рассмотрен для одномерного случая. Отличие заключается в том, что в первом случае происходит считывания одного элемента, а здесь поэлементно производится считывание – целой строки.

Пример 3. Заполнить двумерный массив `A(N, 3)` с помощью датчика случайных чисел. Первый столбец заполнить вещественными числами, второй – целыми, а третий столбец – значениями функции $y = \sin(x)$ Результат поместить в файл File3.txt.

```
Sub InputArray3()
'*********************************************************
'  заполнение двумерного массива при  помощи датчика случайных
'чисел
'*********************************************************
   Dim A() As Variant
   Title = "Заполнение двумерного массива"
   Open "Z:\VBA\File6.txt" For Output As #3
   Randomize
   N = InputBox("Число строк массива", Title)
   ReDim A(1 To N, 1 To 3)
   Pi = 3.14159
   x = -Pi / 2
   For i = 1 To N
      ArrayValue = Int((100 * Rnd) - 50)
      A(i, 1) = ArrayValue
      ArrayValue = (Int((100 * Rnd) - 50)) / 10
      A(i, 2) = ArrayValue
      ArrayValue = Sin(x)
      A(i, 3) = ArrayValue
      x = x + Pi / (N - 1)
      Worksheets("Лист1").Cells(i + 1, 1) = A(i, 1)
      Worksheets("Лист1").Cells(i + 1, 2) = A(i, 2)
      Worksheets("Лист1").Cells(i + 1, 3) = A(i, 3)
      Print #3, A(i, 1), A(i, 2), A(i, 3)
```

```
    Next i
    Close
End Sub
```

В первый столбец вводятся случайные числа в диапазоне от -50 до + 50, а во второй – в диапазоне от - 5 до + 5. В третий столбец вводятся значения функции $y = \sin(x)$ с шагом $\pi/(\mathrm{N}-1)$. Первый, второй и третий столбцы выводятся соответственно в столбцы A, B, C таблицы рабочего листа Excel.

Пример 4. Заполнить двумерный массив A(M, N), используя для ввода массива функцию InputBox, а для вывода – функцию MsgBox

```
Sub InputArray4()
'*********************************************************
'  Ввод с клавиатуры Заполнить двумерный  массив A(M, N),
'используя  для ввода массива функцию InputBox,  а для вывода –
'функцию MsgBox
'*********************************************************
    Dim A() As Integer
    Dim msgString As String
    Dim Title As String
    msgString = ""
    Title = "Заполнение двумерного массива"
    M = InputBox("Число строк", "Размерность массива")
    N = InputBox("Число столбцов", "Размерность массива")
    ReDim A(1 To M, 1 To N)
    For i = 1 To M
```

```
    For j = 1 To N
        A(i, j) = InputBox("Введите A(" & i & ", " & j & "):", Title)
    Next j
  Next i
  For i = 1 To M
    For j = 1 To N
      msgString = msgString & A(i, j) & ", "
    Next j
    msgString = msgString & Chr(13)
  Next i
  MsgBox "Массив A:" & Chr(13) & msgString, , "Вывод массива"
End Sub
```

Этот пример демонстрирует заполнение динамического массива A() с помощью функции InputBox. Сначала вводится число строк M и число столбцов N, а затем последовательно все элементы массива A. Далее происходит формирование переменной msgString – переменной вывода. В эту переменную с помощи операции конкатенации (&) записываются все элементы первой строки, второй строки и т. д. до конца массива.

Пример 5. Вычислить сумму и среднее арифметическое всех элементов массива A(M, N)

```
Sub InputArray5()
'*********************************************************
' Заполнить двумерный массив при  помощи датчика случайных чи-
'сел. Вычислить сумму и среднее арифметическое всех'элементов
'*********************************************************
   Dim A() As Variant
   Dim strMes As String
   Title = "Заполнение двумерного массива"
   Randomize
   M = InputBox("Число строк массива", Title)
   N = InputBox("Число строк массива", Title)
   ReDim A(1 To M, 1 To N)
   For i = 1 To M
      For j = 1 To N
         ArrayValue = 100 - Int((200 * Rnd) + 1)
         A(i, j) = ArrayValue
      Next j
   Next i
   Sum = 0
   For i = 1 To M
      For j = 1 To N
         Sum = Sum + A(i, j)
         msgString = msgString & A(i, j) & ", "
      Next j
      msgString = msgString & Chr(13)
```

```
    Next i
    MsgBox "Массив A:" & Chr(13) & msgString, , "Вывод массива"
    mA = Sum / (M * N)
    MsgBox "Массив A:" & Chr(13) & "Сумма элементов = " & Sum
& Chr(13) & _
        "Среднее арифметическое = " & mA, , "Вывод массива"
End Sub
```

Данный пример аналогичен примеру 4 предыдущей работы и отличается только организации двух циклов по номерам строк и столбцов. По окончании двойного цикла происходит формирование переменной msgString – переменной вывода, в которой находится сумма и среднее арифметическое всех элементов массива A

Пример 6. Переставить в массиве A(M, N) строки с номерами 2 и M - 1

```
Sub InputArray6()
'*********************************************************
'  Заполнить двумерный  массив A(M, N),
'  используя датчик случайных чисел.
'  Переставить в этом массиве строки 2 и M - 1
'*********************************************************
    Dim A() As Integer
    Dim msgString As String
    msgString = ""
    M = InputBox("Число строк", "Размерность массива")
    N = InputBox("Число столбцов", "Размерность массива")
    ReDim A(1 To M, 1 To N)
```

```
Randomize
For i = 1 To M
   For j = 1 To N
      A(i, j) = Int((100 * Rnd) + 1)
   Next j
Next i
For i = 1 To M
   For j = 1 To N
      msgString = msgString & A(i, j) & ", "
   Next j
   msgString = msgString & Chr(13)
Next i
MsgBox "Массив A:" & Chr(13) & msgString, , _
      "Вывод массива (до перестановки)"
For j = 1 To N
   Temp = A(2, j)
   A(2, j) = A(M - 1, j)
   A(M - 1, j) = Temp
Next j
msgString = ""
For i = 1 To M
   For j = 1 To N
      msgString = msgString & A(i, j) & ", "
   Next j
```

```
    msgString = msgString & Chr(13)
  Next i
  MsgBox "Массив A:" & Chr(13) & msgString, , _
      "Вывод массива (после перестановки)"
End Sub
```

В первом двойном цикле массив A заполняется случайными числами, а во втором двойном цикле происходит формирование переменной msgString, которая содержит массив A() до перестановки строк. Далее в цикле по столбцам происходит перестановка второй и предпоследней строки. Формирование переменной вывода, которая содержит массив после перестановки строк аналогично предыдущему.

Пример 7. Найти в массиве A(M, N) строку, сумма элементов которой является максимальной, а в ней минимальный по величине элемент.

```
Sub InputArray7()
'*************************************************************
' Заполнить двумерный массив A(M, N), используя датчик случай-
'ных чисел. Найти в этом массиве строку, сумма элементов  кото-
'рой является максимальной, а в ней минимальный по величине
'элемент.
'*************************************************************
  Dim A() As Integer
  Dim msgString As String
  msgString = ""
  M = InputBox("Число строк", "Размерность массива")
```

```
N = InputBox("Число столбцов", "Размерность массива")
ReDim A(1 To M, 1 To N)
Randomize
For i = 1 To M
   For j = 1 To N
      A(i, j) = Int((100 * Rnd) + 1)
   Next j
Next i
For i = 1 To M
   For j = 1 To N
      msgString = msgString & A(i, j) & ", "
   Next j
   msgString = msgString & Chr(13)
Next i
Max = -10 ^ 9
For i = 1 To M
   Min = 10 ^ 9
   Sum = 0
   SumMax = 0
   For j = 1 To N
      Sum = Sum + A(i, j)
      If A(i, j) < Min Then
         Min = A(i, j)
         Numj = j
      End If
```

```
        Next j
        If Sum > Max Then
            Max = Sum
            Numi = i
            Amin = Min
            Nmin = Numj
        End If
    Next i
msgString = msgString & "Максимальная сумма = " & Max & Chr(13)
msgString = msgString & "Номер строки = " & Numi & Chr(13)
msgString = msgString & "Минимальный элемент = " & Amin _
&Chr(13)
 msgString = msgString & "Номер столбца = " & Nmin
MsgBox "Массив A:" & Chr(13) & msgString, , "Вывод массива"
End Sub
```

2. Практическая часть

Задания к лабораторной работе

Для заданных условий составить процедуру и придумать несколько наборов тестовых данных для отладки. Возможно использование, как статических массивов, так и динамических Ввод исходных данных осуществить из файла или с клавиатуры. Предусмотреть вывод результата в файл или на лист рабочей книги Excel.

1. Написать процедуру вычисления суммы всех элементов для заданного двумерного массива A состоящего из 5-ти строк и 6-ти столбцов.

2. Написать процедуру вычисления для заданного массива A(5,6) среднего арифметического элементов значения его положительных элементов. Известно, что хотя бы один элемент массива имеет положительное значение
3. Для заданного массива A(4,6) вычислить среднее арифметическое значение положительных элементов каждой строки. Результаты поместить в одномерный массив B(4). Известно, что в каждой строке массива хотя бы один элемент имеет положительное значение.
4. Вычислить для заданного массива B(20,30) наибольший элемент каждого столбца. Результаты поместить в одномерный массив BM(30).
5. Написать процедуру поиска для заданного массива A(4,5), строки с наибольшим средним арифметическим значением положительных элементов. Результат поместить в одномерный массив D(5). Считаем, что хотя бы один положительный элемент в каждой строке имеется.
6. Написать процедуру. Для заданного двумерного массива A(6,4), переставить местами 2-ю и 4-ю строки.
7. Написать процедуру. Для заданного массива B(4, 5), переставить местами столбец с наибольшим количеством нулевых элементов и столбец последний по порядку следования в массиве B.
8. Написать процедуру для подсчета в заданном массиве A(5,5) количества элементов превышающих заданное число B и лежащих на главной диагонали и выше нее.
9. Написать процедуру для вывода на печать отрицательных элементов, лежащих на главной диагонали заданного массива A(4, 4).

10. Написать процедуру перестановки элементов заданного массива B(4,4), лежащих на главной и побочной главной диагонали.
11. Написать процедуру пересылки двумерного массива A(5,6) в одномерный массив B(30) того же размера, по строкам с сохранением порядка следования элементов.
12. Написать процедуру по транспонированию заданной квадратной матрицы A максимальная размерность которой 100.
13. Написать процедуру перемножения матрицы A на вектор B.
14. Написать процедуру перемножения матрицы A(5,10) на матрицу B(10,4).
15. Задан массив размером $N \times N$. Разделить элементы каждого столбца на элемент главной диагонали расположенный в соответствующем столбце.
16. Задан двумерный массив размером $N \times N$. Сформировать из его диагональных элементов одномерный массив.
17. Задан массив размером $N \times N$. Определить максимальный и минимальный элементы главной диагонали и переставить местами столбцы в которых лежат эти элементы.
18. Просуммировать элементы заданного двумерного массива размером $N \times N$, расположенные в его верхней части, ограниченной главной и побочной диагоналями, включая элементы, расположенные на этих диагоналях.
19. Для заданного двумерного массива размером $N \times N$ просуммировать элементы, расположенные на диагоналях, параллельных главной. Результаты поместить в одномерный массив.

20. В заданном двумерном массиве размером $N \times M$ поменять местами элементы первого и второго столбца, третьего и четвертого и т. д.
21. Задан массив размером 16. Сформировать из него массив размером 4x4 по строкам.
22. В заданном двумерном массиве размером $N \times M$ переместить нулевые элементы каждого столбца в конец столбца.
23. Задан двумерный массив размером $N \times N$. Максимальный элемент каждой строки поменять местами с диагональным элементом соответствующей строки.
24. Поместить в одномерный массив элементы, расположенные по периметру заданного двумерного массива размером $N \times N$. Использовать один цикл.
25. Заданный двумерный массив размером $N \times N$ повернуть вправо на 90°, без использования вспомогательных массивов.
26. В заданном двумерном массиве размером $N \times N$ поменять местами элементы, расположенные в верхней и нижней частях массива, между главной и побочной диагоналями за исключением элементов, расположенных на этих диагоналях.
27. Двумерный массив размером $N \times N$ задан в виде одномерного массива по столбцам. Вывести на печать верхний треугольник массива по строкам, включая элементы главной диагонали.
28. Написать процедуру, задающую треугольную матрицу произвольного размера. Число, определяющее размер матрицы вводится с клавиатуры во время выполнения процедуры
29. Сформировать двумерный массив, у которого элементы равны произведению своих индексов
30. Найти сумму элементов главной и произведение элементов побочной диагоналей квадратной матрицы

ЛАБОРАТОРНАЯ РАБОТА № 7
Подпрограммы

1.Краткие теоретические сведения

В целях ускорения работы программиста и упрощения применяемых алгоритмов применяют подпрограммы. Применение подпрограмм позволяет разбить сложную задачу на ряд простых, взаимосвязанных задач. Проделав эту операцию неоднократно применительно к вновь полученным подпрограммам, можно получить алгоритм решения задачи в виде набора простых, понятных, легко осуществимых подпрограмм.

В VBA любая программа написанная программистом является процедурой. Различают два вида процедур:

процедура **(Sub),**

процедура функция **(Function)**.

Процедура

Процедура является самостоятельной частью кода, которая имеет имя и может содержать аргументы, выполнять последовательность инструкций (операторов) и изменять значения своих аргументов.

Синтаксис:

[Private | Public] [Static] **Sub** Имя (*СписокФормальныхПараметров*)

[Инструкции]

[**Exit Sub**]

[Инструкции]

End Sub

Элементы описания:

Public	Указывает, что процедура **Sub** доступна для всех других процедур во всех модулях
Private	Указывает, что процедура **Sub** доступна для других процедур только того модуля, в котором она описана
Static	Указывает, что локальные переменные процедуры **Sub** сохраняются в промежутках времени между вызовами этой процедуры
Имя	Имя процедуры **Sub**, удовлетворяющее стандартным правилам именования переменных
СписокФормальныхПараметров	Список переменных, представляющий аргументы, которые процедура **Sub** должна получить, чтобы их обработать. Имена переменных разделяются запятой
Инструкции	Любая группа инструкций, выполняемых в процедуре **Sub**

Инструкция Exit Sub приводит к немедленному выходу из процедуры **Sub**.

Аргументы обеспечивают процедуру данными, использующимися в ее инструкциях. Аргумент может *передавать* следующие данные:

–переменная;

–константа;

–массив;

–объект.

Способ вызова процедуры **Sub** производится с помощью инструкции

ИмяПроцедуры СписокФактическихПараметров

или

Call *ИмяПроцедуры (СписокФактическихПараметров)*

Обратите внимание, что в этом случае список фактических параметров заключается в скобки. В первом способе скобки не использовались.

СписокФактическихПараметров - это список из имён переменных, массивов, констант, которые передаются в вызываемую процедуру (которые она реально обрабатывает). VBA позволяет вводить фактические параметры через имена аргументов в любом порядке и опускать необязательные (optional). При этом после имени аргумента ставятся двоеточие и знак равенства, после которого помещается значение аргумента (фактический параметр).

Процедура **Sub** – это аналог макроса. Для её выполнения надо обратиться к имени процедуры в меню макросов или набрать ранее назначенную комбинацию клавиш.

При создании процедур необходимо учитывать следующие правила:

-процедура может не иметь списка формальных параметров;

-процедура может иметь фиксированное количество аргументов в списке формальных параметров;

-процедура может иметь неопределенное количество аргументов в списке формальных параметров;

-не все аргументы в списке формальных параметров процедуры являются обязательными;

-все аргументы в списке формальных параметров могут быть необязательными.

Процедура типа **Function**, в отличие от процедуры типа **Sub**, обязательно возвращает значение определённого типа в вызывающую процедуру. Чтобы вызвать функцию достаточно обратиться к ней по имени. Например, y=sin(x) *(вызывает встроенную функцию синуса).*

Процедура типа **Function** универсальна и используется в двух ситуациях:

−как часть выражения в процедуре VBA;

−в формулах, которые создаются на рабочем листе.

Процедуру типа **Function** можно использовать везде, где применяются функции Excel и встроенные функции VBA. . Как и встроенные функции, процедуры типа Function имеют аргументы.

[Private | Public] **Function** *Имя (СписокФормальныхПараметров)_*

As *ТипПеременной*

[Инструкции]

End Function

Область определения переменной и процедуры

Область определения переменной задает область, в которой может быть использована переменная. В VBA имеется три соответствующих уровня переменных:

1. Переменные уровня процедуры (private) используются только внутри процедуры, в которой они описаны при помощи инструкции Dim, размещенной в процедуре.
2. Переменные уровня модуля используются только в модуле, в котором они описаны при помощи инструкции Dim,

размещенной в области описания модуля, т. е. перед описанием процедур.

3. Общие переменные, используемые во всех модулях данного проекта. Описываются при помощи инструкции Public, размещенной в области описания модуля.

Время жизни переменной

Личная (private) переменная сохраняет свои значения, только пока выполняется процедура, в которой эта переменная описана. При завершении процедуры значение переменной теряется, и при повторном запуске процедуры его надо заново инициализировать. Переменные, описанные при помощи инструкции Static, сохраняют свое значение по выходу из процедуры, но пока работает программа

Приведенный ниже пример показывает основные способы передачи параметров в процедуры.

Пример 1

```
Dim c As Double                    ' c - глобальный параметр
Function F(ByVal x As Integer) As Integer
F = x ^ 2
End Function
'**********************
Sub Assistant(ByVal a As Integer, ByVal b As Integer)
' Процедура, находящая сумму двух чисел и выводящая
' результат в диалоговом окне
c = a + b
```

```
MsgBox CStr(c)
End Sub
Sub Main ()
' Процедура, находящая сумму двух чисел и выводящая
' результат в диалоговом окне
Dim x, y As Double
' x, y —фактические параметры
' Вызов процедуры с числами как фактическими параметрами
Assistant 1, 3
' Первоначальное присвоение переменным значений,
' с последующим вызовом процедуры
x = 1: y = 1 Assistant x, y + 2
' Использование процедуры как фактического параметра
x = 1: y = 3
Assistant F(x), y
' Вызов процедуры с указанием фактических параметров по имени
Assistant a:=l, b:=3
End Sub
```

Приведем пример (**пример 2)** процедуры с необязательными параметрами. Функция с именем СторонаТреугольника позволяет найти длину недостающей стороны прямоугольного треугольника, где переменные A и B отведены под длины катетов, а переменная C — под гипотенузу. Например, формула Excel

=СторонаТреугольника(;B2;C2)

вычисляет катет A по введенным в ячейки B2 и C2 длинам катета B и гипотенузы C. При работе с необязательными переменными необходимо использовать функцию isMissing, возвращающую значение True, если соответствующий аргумент не был передан в процедуру, и False в противном случае.

Пример 2.

```
Function СторонаТреугольника(Optional A,_Optional B, Optional C)
If Not (IsMissing(A)) And Not (IsMissing(B)) Then
    СторонаТреугольника = Sqr(A ^ 2 + B ^ 2)
End If
If Not (IsMissing(A)) And Not (IsMissing(C)) Then
  СторонаТреугольника = Sqr(C ^ 2 - A ^ 2)
End If
If Not (IsMissing(B)) And Not (IsMissing(C)) Then
  СторонаТреугольника = Sqr(C ^ 2 - B ^ 2)
End If
End Function
```

Составить фрагмент программы вычисления следующих функций с помощью процедуры типа функция:

$$C = \frac{4{,}5e^{x^2}}{e^x + 8{,}5\sin(x)} \quad \text{и} \quad D = \frac{1}{8\,e^{x^4} - \sin(x^4) + x^3}$$

Фрагмент процедуры типа Function

```
'описание функции
Function FNC (ByVal A As Double, ByVal X As Double, _
ByVal Z Double)As Double
```

```
 FNC= A∗EXP (X) + Z ∗ SIN (X)
End Function

'использование функции
        …
        C = FUN( A, X**2, 0. ) / FUN( 1., X, 8.5 )
        B = -1.
        D = 1/(FUN (8., X**4, B) + X ** 3)
        . . .
```

Функция с именем FNC и формальными параметрами A, X, Z оформлена в данном примере в виде функции $a \cdot e^x + z \cdot \sin x$. В дальнейшем эта функция используется при вычислении C и D.

При вычислении C в качестве фактических параметров при первом и втором обращениях к оператору-функции соответственно используются A, X ** 2,0. и 1., X, 8.5, которые замещают формальные параметры A, X, Z оператора-функции FNC.

Что касается аргументов, то процедуры подобны функциям Excel в следующем: существует два способа передачи аргументов в процедуру: по ссылке и по значению. При передаче аргумента по ссылке (этот метод используется по умолчанию) в процедуру передается всего лишь ***адрес*** хранения переменной в памяти. При передаче аргумента по значению в процедуру передается ***копия*** исходной переменной. Следовательно, изменение аргумента в процедуре не вызывает изменения исходной переменной.

Следующий пример (**пример 3)** подтверждает высказанную концепцию. Аргумент процедуры Process передается по ссылке (по умолчанию). После того, как процедура Main присваивает переменной MyValue значение 10, она вызывает процедуру Process и

передает MyValue в качестве аргумента. Процедура Process умножает значение своего аргумента (с названием YourValue) на 10. По окончании процедуры Process возобновляется выполнение процедуры Main, а функция MsgBox отображает MyValue: 100.

Пример 3.

```
Sub Main
MyValue = 10
Call Process(MyValue)
MsgBox MyValue
End Sub
Sub Process(YourValue)
YourValue = YourValue * 10 'по ссылке меняет MyValue
End Sub
```

Если необходимо, чтобы вызываемая процедура не изменяла переменные, полученные как аргументы, то надо изменить список аргументов вызываемой процедуры так, чтобы аргументы передавались ***по значению***, а не ***по ссылке***. Для этого добавьте перед аргументом ключевое слово **ByVal**. Тогда вызываемая процедура будет управлять копией переданных данных, а не самими данными.

В следующей процедуре, например, изменения, которые происходят с YourValue в процедуре Process, не влияют на значение переменной MyValue в процедуре Main. В результате функция MsgBox
отображает 10, а не 100.

Пример 3. (Другой вариант)

```
Sub Process(ByVal YourValue)
YourValue = YourValue * 10 'по значению не меняет MyValue
```

End Sub

В большинстве случаев вполне подходит метод, используемый по умолчанию, – передача аргументов по ссылке. Однако если процедура призвана изменить данные, которые передаются ей в виде аргументов, а первоначальные данные должны остаться неизменными, то необходимо использовать передачу данных по значению. Процедуру можно рассматривать как команду, которая выполняется пользователем или другой процедурой. С другой стороны, процедуры типа Function обычно возвращают отдельное значение (или массив), подобно функциям рабочих листов Excel и встроенным функциям VBA.

Рекурсивные процедуры

В VBA возможно создание рекурсивных процедур, т. е. процедур, вызывающих самих себя. Стандартным примером рекурсивной процедуры является процедура вычисления факториала, т. е. функции, возвращающей результат произведения первых п натуральных чисел, где п — аргумент функции. Для этой функции имеется стандартное обозначение:

Fact(n) =n!, где Fact(0) =1.

Ясно, что

Fact(n) = n*Fact(n – 1)

Основываясь на данном соотношении, приводимая ниже рекурсивная функция вычисляет значение факториала.

Function Fact(n **As Integer**) **As Integer**

If n<l **Then**

Fact = 1 Else

Fact = Fact(n - 1)*n

End If

End Function

Несмотря на элегантность рекурсивных процедур, применять их надо с осторожностью, т. к. их неаккуратное использование может привести к проблемам с памятью — многократный вызов такой процедуры быстро исчерпывает стековую память.

Пример 4. Заданы два одномерных массива A(10), B(30). Написать процедуру определения максимального элемента и его индекса.

Пример 4.

```
'глобальные массивы
Dim A(10) As Double
Dim B(30) As Double
'процедура поиска MAX элемента и его индекса
Sub ARMAX(AR, MAR, NAR, MAX_N_AR)
Dim NAR As Integer                    'размер массива
Dim MAX_N_AR As Integer               'индекс MAX элемента
Dim AR(NAR) As Double                 ' массив
Dim MAR As Double                     ' MAX элемент
Dim k As Integer
MAR = AR(1)
For k = 1 To NAR
   If (MAR < AR(k)) Then
      MAR = AR(k)
      MAX_N_AR = k
```

```
    End If
Next
End Sub
'основная процедура
Sub main()
A_max                    'переменная, хранящая max элемент A
B_max                    'переменная, хранящая max элемент B
i_a_max                  'переменная, хранящая индекс max A
i_b_max                  'переменная, хранящая индекс max B
'ввод значений в массивы A,B
' .....
ARMAX A, A_max, 10, i_a_max          'обработали массив A
ARMAX b, B_max, 30, i_b_max          'обработали массив B
'вывод результатов
   MsgBox "MAX элемент Массива A:", A_max, "номер", i_a_max
   MsgBox "MAX элемент Массива B:", B_max, "номер", i_b_max
End Sub
```

2.Практическая часть

Задание к лабораторной работе

Написать программу, осуществляющую заданные вычисления с использованием процедур. Вид используемых процедур определить самостоятельно.

1. Заданы две матрицы A(4,4) и B(4,4). Написать программу суммирования двух векторов X и Y. Где X - вектор, в котором размещены элементы столбца матрицы A с минимальным средним арифметическим значением, Y - то же для матрицы B.

2. Заданы две матрицы A(4,4) и B(4,4). Написать программу вычисления вектора Z = X + У, где X - строка матрицы A, включающая минимальный элемент ее главной диагонали, Y- то же для матрицы B.
3. Заданы две матрицы A(4,4) и B(3,3) и два вектора C(4) и D(3). Написать программу вычисления произведений матриц на соответствующие им вектора. Определить минимальное среднее арифметическое значение для полученных векторов.
4. Заданы две матрицы B(4,4) и D(3,3). Написать программу транспонирования каждой из заданных матриц с последующим перемножением транспортированной матрицы на соответствующую ей исходную.
5. Заданы две матрицы A(3,3) и B(3,3). Написать программу проверки - является ли произведение этих матриц перестановочным, т.е. проверить равенство AB = BA.
6. Заданы дне матрицы C(4,4) и D(3,3). Написать программу определения количества симметричных матриц. Матрица называется симметричной, если транспонированная матрица равна исходной. Для каждой симметричной матрицы вычислить сумму элементов, лежащих вне главной диагонали.
7. Заданы два массива C(6) и D(10). Для каждого из них осуществить перестановку первого по порядку элемента со вторым отрицательным элементом массива.
8. Заданы три квадратных уравнения $AX^2 + BX + C = 0$; $DX^2 + FX + E = 0$ и $ZX^2 + YX + S = 0$. Написать программу нахождения максимального значения корня среди действительных корней этих уравнений.
9. Заданы два вектора A(0.052; 0.9; 0.15; 0.84; 0.67) и B(0.948; 0.1; 0.33; 0.16; 0.85). Написать программу, позволя-

ющую расположить элементы одного вектора по возрастанию, а другого - по убыванию. Вычислить сумму полученных векторов.

10. Заданы два двумерных массива A(4,4) и B(3,3). Для каждого из них переставить столбцы с максимальным и минимальным элементами.
11. Заданы координаты четырёх точек A, B, C, D на плоскости. Определить наибольший из периметров треугольников ABC, ABD, ACD.
12. Три треугольника заданы координатами своих вершин. Написать программу вычисления площадей треугольников и определения минимальной площади.
13. Заданы два двумерных массива A(4,4) и B(3,3). Для каждого из них переставить последнюю строку со строку с максимальным средним арифметическим значением.
14. Заданы две матрицы A(4,4) и B(3,3). Написать программу вычисления вектора Z = X – Y, где X -столбец матрицы A. включающий минимальный элемент матрицы; Y –то же для матрицы B.
15. Заданы три вектора X, Y, Z. Написать программу вычисления $S = \sum_{i=1}^{n} C_i \cdot D_i$, где $\mathbf{C_i}$ и $\mathbf{D_i}$ - компоненты векторов C и D, которые соответственно равны C = X + Y; D = X - Z
16. Заданы матрицы C(4,4) и D(3,3). Определить индексы максимального элемента каждой из матриц среди элементов, расположенных выше главной диагонали.
17. Заданы вектора A(5), B(10), D(15). Дли каждого из них определить максимальный и минимальный элементы и их индексы.

18. Заданы координаты вершин трех треугольников. Определить сколько треугольников лежит внутри окружности радиуса R с центром в начале координат.
19. Заданы координаты десяти точек плоскости и координаты точки-полюса. Найти точку, максимально удалённую от полюса, среди первых четырёх заданных точек и такую же точку - среди последних шести. Определить также расстояние между найденными точками.
20. Заданы массивы Y(4,4) и Z(8,8). Для каждого из них вычислить среднее арифметическое значение положительных элементов, расположенных выше главной диагонали.
21. Заданы координаты вершин трех треугольников. Определить треугольник с максимальным периметром.
22. Заданы три матрицы размерами 2x2, 3x3, 4x4. Для каждой из матриц определить среднее арифметическое значение положительных элементов главной диагонали.
23. Заданы три одномерных массива разной размерности. Для каждого из массивов определить повторяющиеся элементы.
24. Заданы координаты четырех точек на плоскости. Определить номера точек, расстояние между которыми минимальное.
25. Заданы две окружности $(x - a)^2 + (y - b)^2 = r^2$ и $(x - c)^2 + (y - d)^2 = R^2$ и координаты четырех точек. Определить количество точек, лежащих внутри каждой окружности.
26. Заданы три одномерных массива A(5), B(10), C(15). Написать программу решения уравнения $px^2 + dx + r = 0$,
    где p - минимальный элемент матрицы A,
    d - минимальный элемент матрицы B,
    r - минимальный элемент матрицы C.

27. Заданы одномерные массивы X(5) и Y(7). Для каждого из них определить количество и сумму элементов, которые без остатка делятся на заданное число B.
28. Составить программу заполнения массивов A(5) и B(10) факториалами значений индексов их элементов. Вычисление факториала выполнить в подпрограмме.
29. Заданы матрицы A(4,4), B(6,6). Для каждой матрицы определить среднее арифметическое значение положительных элементов, расположенных не выше главной диагонали.
30. Матрицами F(2,2) и E(2,2) в декартовой системе координат заданы две окружности положением своего центра и точки на дуге. Вычислить параметры окружностей.

ЛИТЕРАТУРА

1. Архипов В.Н., Калядин В.И., Любин А.Н., Макаров А.И. Программирование на фортране:методические указания к лабораторным работам. - М.;МГТУ «МАМИ», 2007., -144с.
2. Гарнаев А. Самоучитель VBA технология создания пользовательских приложений. - М. BHV, 2006., – 153с.
3. Калядин В.И., Макаров А.И. основы работы на персональном компьютере: сборник лабораторных работ. - М.;МГТУ «МАМИ», 2010., -85с.
4. Кузьменко В.Г. VBA эффективное использование М. БИНОМ 2009., -617с.
5. Лобанов А.С., Туманова М.Б. Решение задач на языке Visual Basic for Application: учебное пособие. - М.;МГТУ «МАМИ», 2009., -90с.
6. Слепцова Л.Д. Программирование на VBA в Microsoft Office 2010. - М. «Диалектика», 2010., -431с.
7. Уокенбах Дж. Профессиональное программирование на VBA в Excel 2003.- М. «Диалектика», 2005., -800с.

Содержание

Учебное издание

Антомони Валерий Иосифович
Архипов Владимир Николаевич
Любин Александр Николаевич
Тихомиров Василий Николаевич

ПРОГРАММИРОВАНИЕ НА VBA В MICROSOFT OFFICE
Сборник лабораторных работ

Под редакцией авторов

Оригинал-макет подготовлен редакционно-издательским отделом МГТУ «МАМИ»

По тематическому плану внутривузовских изданий учебной литературы на 2011 г.

Подписано в печать _____Формат 60×90 1/16 Бумага 80 г/м^2

Гарнитура «Таймс». Ризография. Уч. печ. л.___

Тираж___экз. Заказ №_____

МГТУ «МАМИ»

107023, г. Москва, Б. Семёновская ул. д. 38.