МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__» _____ 2022 г.

Информатика. Часть 1.

Методические указания по выполнению
практических работ
для студентов направления подготовки 43.03.03

Курск 2022

УДК 004.2

Составители: И.Н. Ефремова, В.В. Ефремов

Рецензент
Кандидат технических наук, доцент *Т.М.Белова*

Информатика. Часть 1.: методические указания по выполнению практических работ / Юго-Зап. гос. ун-т; сост.: И.Н. Ефремова. В.В.Ефремов, Курск, 2022. - 31 с.

Содержат описание восьми практических работ по дисциплине «Информатика».

Предназначены для студентов направления 43.03.03

Текст печатается в авторской редакции

Подписано в печать Формат 60х84 1/16.
Усл. печ. л. 1,8. Уч.-изд. л. 1,7. Тираж 100 экз. Заказ 48. Бесплатно 1185
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Содержание

Практическая работа №1.

Текстовый редактор Microsoft Word
Оформление текстового документа

1. Краткие теоретические положения.

При загрузке Microsoft Word появляется окно, верхняя строка которого содержит название загруженного файла, ниже расположена строка меню, еще ниже – панели инструментов. Выбрать или убрать требуемые панели инструментов можно с помощью пункта меню Панели инструментов. Рабочее поле (белый лист) обрамлено горизонтальной и вертикальной линейками (убрать или установить которые можно с помощью меню Вид) и горизонтальной и вертикальной полосами прокрутки. В самом низу расположена строка, показывающая номер текущего листа, позицию курсора, общее количество листов документа и т.д.

СОЗДАНИЕ, СОХРАНЕНИЕ, ОТКРЫТИЕ И ЗАКРЫТИЕ ДОКУМЕНТА. Для того, чтобы создать документ следует выбрать пункт меню Файл - Создать и выбрать обычный или новый документ, или, щелкнуть на соответствующей кнопке стандартной панели инструментов. При загрузке Microsoft Word новый документ создается автоматически.

Чтобы сохранить документ следует выбрать пункт меню Файл - Сохранить или Файл Сохранить как… и выбрать папку и имя документа, или, щелкнуть на соответствующей кнопке стандартной панели инструментов. При этом если документ уже сохранялся ранее, команда Файл - Сохранить сохранит его в прежнем каталоге и с прежнем именем.

Открыть (закрыть) имеющийся документ можно с помощью пунктов меню Файл - Открыть (Закрыть) или специальных кнопок.

НАБОР ТЕКСТА. Для переключения на русский/ английский язык используется, как правило, сочетание клавиш <левый Alt> + <Shift> или <Ctrl> + <Shift>, или, специальный значок на панели задач. Для переключения регистров используется клавиша CapsLock (горящая лампочка индицирует верхний регистр (прописные буквы)). Временное переключение регистров можно осуществить, удерживая нажатой клавишу Shift.

При наборе текста осуществляется автоматический переход на следующую строку. Для того, чтобы начать следующий абзац, нажмите клавишу Enter. Отсутствующие на клавиатуре символы можно ввести, выбрав их с помощью меню Вставка - Символ.

ПЕРЕМЕЩЕНИЕ КУРСОРА ПО ТЕКСТУ осуществляется с помощью стрелок - на одну позицию в указанном направлении, End – в конец строки, Home – в начало строки, PageDone – на страницу назад, PageUp – на страницу вперед. Нажатие совместно с Ctrl клавиш: позволяет переместиться на одно слово в указанном направлении, или на один абзац в указанном направлении, End – в конец текста, Home – в начало текста.

Переместить курсор можно также, щелкнув левой кнопкой мыши в требуемой позиции текста. При этом просмотреть текст можно с помощью вертикальной и горизонтальной полос прокрутки. Изменить масштаб текста на экране можно с помощью меню Вид - Масштаб или специальной вкладки на стандартной панели инструментов.

ВЫДЕЛЕНИЕ ФРАГМЕНТА ТЕКСТА. Для того, чтобы выделить фрагмент текста надо установить мышиный курсор в начало фрагмента, нажать левую кнопку мыши и, не отпуская ее, протянуть выделение. При этом выделенный фрагмент подсвечен. Можно выделять фрагменты текста также следующими способами:

- поставив курсор в начало фрагмента и щелкнув в конце фрагмента левой кнопкой мыши с нажатой клавишей Shift,

- щелчок слева от текстового поля, когда мышиный курсор принимает вид стрелки, приводит к выделению строки, если кнопку мыши не отпускать, можно выделять текст строками,

- весь текст можно выделить, выполнив команду Выделить все.

РЕДАКТИРОВАНИЕ ТЕКСТА. Для удаления символа, стоящего перед курсором, нажмите клавишу Backspace, для удаления символа, стоящего после курсора, нажмите клавишу Delete. Можно заменить выделенный фрагмент текста новым, набрав его на клавиатуре. Для того, чтобы вырезать выделенный фрагмент текста в буфер, выберите пункт меню Вырезать (или нажмите Ctrl+X) или щелкните на соответствующей кнопке стандартной панели инструментов. Для того, чтобы копировать выделенный фрагмент текста в буфер, выберите пункт меню Копировать (или нажмите Ctrl+C) или щелкните на соответствующей клавише стандартной панели инструментов. Для того, чтобы вставить фрагмент текста, находящийся в буфере, в позицию курсора, выберите пункт меню Вставить (или нажмите Ctrl+V) или щелкните на соответствующей клавише стандартной панели инструментов.

Если в результате редактирования текста было произведено ошибочное действие, отменить команду можно с помощью пункта меню Отменить или специального значка на стандартной панели инструментов.

ПРОВЕРКА ПРАВОПИСАНИЯ осуществляется автоматически или с помощью меню Правописание. При автоматической проверке правописания, слова, отсутствующие в словаре, подчеркиваются красной волнистой линией, фрагменты предложения, отличающиеся конструктивно от имеющихся вариантов, подчеркиваются зеленой чертой. При этом указанные подчеркивания не выводятся на печать.

Если щелкнуть правой кнопкой мыши на подчеркнутом участке, Word выведет некоторые пояснения или список предлагаемых вариантов, выбрав из которых, можно автоматически исправить ошибку.

ФОРМАТИРОВАНИЕ СИМВОЛОВ выделенного фрагмента осуществляется с помощью меню Шрифт путем выбора шрифта, размера шрифта, начертания (обычный, курсив, полужирный, полужирный курсив), подчеркивания и др., а также с помощью специальных вкладок и кнопок на панели Форматирование. Проведенные установки действительны также для текста, набираемого ниже.

ФОРМАТИРОВАНИЕ АБЗАЦЕВ. Для того, чтобы отформатировать несколько абзацев, надо их выделить, чтобы отформатировать один, достаточно установить в него курсор. Форматирование осуществляется с помощью пункта Абзац путем выбора отступов абзаца слева и справа, интервалов перед абзацем и после него, межстрочного интервала, вида первой строки (отступ, выступ или нет), вида выравнивания (также можно задать с помощью соответствующих кнопок на панели Форматирования) и др..

Задать отступы слева и справа, а также первой строки можно также, перетащив соответствующие движки на линейке.

Проведенные установки также действительны для ниже набираемого текста.

Для того, чтобы автоматически расставить переносы следует выполнить команду Расстановка переносов.

ЗАДАНИЕ ПАРАМЕТРОВ СТРАНИЦЫ осуществляется с помощью пункта Параметры страницы путем выбора размеров полей, размера бумаги и ее ориентации (книжная или альбомная).

ПРЕДВАРИТЕЛЬНЫЙ ПРОСМОТР И ПЕЧАТЬ ДОКУМЕНТА осуществляются с Печать (можно задать номера печатаемых страниц и количе-

ство копий), а также с помощью специальных кнопок на стандартной панели инструментов.

СОЗДАНИЕ ТАБЛИЦ

Установите текстовой курсор в том месте, где вы хотите расположить таблицу и выберите команду меню ВСТАВИТЬ ТАБЛИЦУ. В появившемся диалоговом окне укажите необходимое число столбцов и строк.

После вставки таблицы вы можете включить или выключить изображение пунктирной сетки, показывающей границы ячеек таблицы.

Для перемещения текстового курсора по ячейкам таблицы используется клавиша Tab. Ниже приведен перечень клавиш и их комбинаций для работы с таблицами.

Чтобы добавить в таблицу строку, выделите строку и вызовите меню ТАБЛИЦА-ВСТАВИТЬ СТРОКИ. Выше текущей строки появится пустая строка. Для удаления строки, выделите строку и вызовите меню ТАБЛИЦА-УДАЛИТЬ СТРОКИ.

Добавление и удаление столбцов происходит аналогично, только необходимо выделить столбец и вызывать меню ВСТАВИТЬ СТОЛБЕЦ, УДАЛИТЬ СТОЛБЕЦ.

Чтобы удалить всю таблицу целиком, необходимо выделить ее и вызвать меню ТАБЛИЦА-УДАЛИТЬ СТРОКИ.

Ячейки таблицы могут содержать текст, рисунки и другие обьекты.

Чтобы выровнять текст в таблице или ее части (строке, столбце, ячейке) по левому краю, по центру и т.п., необходимо выделить эту часть таблицы и выполнить операцию выравнивания так же, как и для обычного текста.

ВЫСОТА ЯЧЕЕК. При изменении размера шрифта в ячейках таблицы высота строки таблицы автоматически увеличивается или уменьшается так, чтобы вместить текст. Если необходимо принудительно изменить высоту строки, то установите курсор в нужную строку и вызовите меню ТАБЛИЦА-ВЫСОТА И ШИРИНА ЯЧЕЕК.

ШИРИНА ЯЧЕЕК. При создании таблицы все столбцы имеют одинаковую ширину. Чтобы изменить ширину столбца, установите курсор мыши на разделительную линию между столбцами, и когда появится двунаправленная стрелка, нажмите кнопку мыши и перетащите разделительную линию. Если необходимо установить ширину столбца в миллиметрах, вызовите меню ТАБЛИЦА-ВЫСОТА И ШИРИНА ЯЧЕЕК. Если необходимо установить одинаковую ширину в нескольких столбцах, выделите эти столбцы и вызовите меню ТАБЛИЦА-ВЫСОТА И ШИРИНА ЯЧЕЕК.

ОБРАМЛЕНИЕ ТАБЛИЦЫ

Чтобы создать рамку таблицы и расчертить ее линиями, необходимо выполнить следующие действия:

Выделите таблицу.

Разверните список кнопки ГРАНИЦЫ

Выберите мышью нужный тип обрамления.

ВСТАВКА РИСУНКОВ В ТЕКСТ.

Чтобы вставить в документ готовый рисунок, выполните следующие действия:

- Установите текстовый курсор в позицию, где должен быть рисунок.
- Выберите команду *Вставка-Рисунок*.

- В появившемся диалоговом окне выберите файл.

Чтобы изменить размер рисунка, щелкните на нем мышью. Вокруг рисунка появится рамка с маленькими черными квадратами - маркерами. Щелкните мышью на одном из маркеров и перетаскивайте его, удерживая левую кнопку мыши, пока не подберете подходящий размер рисунка.

Если необходимо передвинуть рисунок в другое место:

- Выделите рисунок щелчком мыши.
- Перетащите рисунок в нужное место, удерживая кнопку мыши.

Возможно изменить параметры заливки рисунка текстом, его привязку и прочее.

Автофигуры.

С помощью встроенного в Word для Windows графического редактора вы можете создавать рисунки в тексте документа, используя функции рисования элементарных геометрических объектов: линий, прямоугольников, кругов и т. д. с помощью графических примитивов (линия, стрелка, овал, прямоугольник и др.). Щелкните мышью на кнопке и поместите выбранную фигуру в документ. Если это необходимо, можно сгруппировать несколько графических элементов, и, наоборот, разгруппировать ранее сгруппированные.

После того как вы нарисовали какую-нибудь фигуру, выделите ее щелчком мыши. Теперь вы можете:

- изменить цвет и стиль контурной линии с помощью кнопки ЦВЕТ ЛИНИИ на панели инструментов РИСОВАНИЕ;

- выбрать цвет заливки для внутренней области фигуры с помощью кнопки ЦВЕТ ЗАПОЛНЕНИЯ;

- выбрать узор заполнения из предложенных шаблонов, для чего щелкните по обьекту два раза, и в окне свойств обьекта выберите вкладку ЗАПОЛНЕНИЕ.

Чтобы изменить фигуру, выделите ее щелчком мыши. Чтобы выделить несколько фигур на рисунке, нажмите на кнопку ВЫДЕЛЕНИЕ РИСОВАННЫХ ОБЬЕКТОВ (белая стрелка) и обведите нужные обьекты. Появится рамка с восемью маркерами, перетаскивая которые вы можете изменять размеры выбранного элемента. Если держать нажатой клавишу Shift, то изменения будут происходить по двум направлениям пропорционально. Это свойство используется для масштабирования кругов и квадратов. Если держать нажатой клавишу Ctrl, то масштабирование будет происходить относительно центра.

Вы можете перемещать выделенный обьект или группу обьектов, перетаскивая его мышью.

Можно указать какой элемент должен находится на переднем плане, а какой на заднем. Для этого нужно выделить соответствующий элемент и нажать кнопку ПОМЕСТИТЬ НАВЕРХ или ПОМЕСТИТЬ НАЗАД.

С помощью редактора можно поворачивать элемент относительно вертикальной или горизонтальной оси, используя соответствующие кнопки. Эти возможности используются для создания симметричных рисунков. Чтобы повернуть обьект используйте кнопку СВОБОДНОЕ ВРАЩЕНИЕ из панели инструментов РИСОВАНИЕ.

2.Задание к Л.Р. "MS Word"

1.Установить параметры страницы: Поле верхнее, нижнее, левое - 2см., правое -1 5см., переплёт - 1 см., размер бумаги - А4, ориентация - книжная.

2.Создать титульный лист, расположение надписей определить стандартными
средствами текстового редактора.

3.Далее все пункты задания начинать с заголовка.

4.Набрать три абзаца текста произвольного осмысленного содержания на русском или английском языке, каждый абзац должен содержать несколько предложений и занимать не менее пяти строк (порядка 350 символов). Каждый абзац должен быть оформлен и отформатирован произвольным уникальным способом (->формат->шрифт, ->формат->абзац и т.п.)

5.Создать таблицу не менее чем 5х5, произвольное осмысленное содержание, используя объединение и разбиение ячеек, параметры границ, заливки, положения, размеры и поля ячеек задать вручную. Таблицу расположить на отдельном листе, листу с таблицей задать альбомную ориентацию.

6.Скопировать три абзаца текста на следующий лист, отформатировать их, применив единый стиль, оформить колонками. Установить параметры автоматической расстановки переносов.

7.На той же странице создать многоуровневый список произвольного осмысленного содержания, один из уровней должен быть нумерованным, один из уровней должен быть маркированным.

8.Создать простейший рисунок (например, блок-схему алгоритма), используя автофигуры и группировку. Вставить рисунок из коллекции MS Office или с диска Положение, параметры перекрытия, обтекания и размеры задать вручную. Вставить название рисунка.

9.Создать надпись (бэдж или визитку), задать цвет, объем.

10. Вставить математическую формулу, используя мастер формул.

11. Вставить гиперссылку на любимый сайт и свой адрес электронной почты.

12. На следующем листе создать оглавление (->вставка->ссылка->оглавление и указатели)

13. Оформить колонтитулы, содержащие фамилию автора, дату создания и номера страниц. При этом,титульный лист и оглавление колонтитулов содержать не должны.

14. Проверить правописание и исправить ошибки.

15. Сохранить файл в форматах doc, rtf, html

3.Содержание отчёта

Титульный лист.

Результаты выполнения работы.

Ответы на контрольные вопросы.

4.Контрольные вопросы

1.Назначение текстового редактора?

2.Возможности редактора Word?

3.Элементы текста в Word?

4.Какие бывают расширения файлов, созданных в Word?

Практическая работа №2.

Microsoft Excel. Создание таблицы.

Предлагаемые исходные данные для работы

Штатное расписание

ФИО	Стаж	Разряд	Разряд-ный коэфф.	Оклад, руб.	Число часов	Всего начисле-но	Подоход. налог	Проф-союз. сборы	Всего удержа-но	К выдаче	Процент от сум-марной выдачи
Иванов А.А.	5	10	3,3		21						
Петров Б.Б.	7	11	3,4		30						
Левин С.М.	1	10	3,3		22						
Котова Ю.П.	2	10	3,3		35						
Кротова С.М.	10	12	3,45		18						
Седых А.Ю.	20	12	3,45		20						
Итого											

Порядок выполнения работы

Запустите Microsoft Excel.

Наберите таблицу.

1. Введите заголовок «Штатное расписание» в ячейку А1
2. Введите строку заголовков столбцов в ячейки А2, В2 и т.д. Чтобы длинные названия выводились в 2 строки, выделите строку заголовков и выполните команду ФОРМАТ-ЯЧЕЙКИ-ВЫРАВНИВАНИЕ-ПЕРЕНОСИТЬ ПО СЛОВАМ.
3. Введите данные в столбцы «ФИО, стаж, разряд, разрядный коэффициент, число часов» таблицы. После ввода данных в ячейку нажимайте Enter. Чтобы отредактировать уже введенные в ячейку данные, выполните

двойной щелчок мыши на ячейке. Чтобы стереть часть таблицы, выделите ее и нажмите Del.

4. Выделите таблицу и выровняйте данные в ячейках по центру значком «по центру» на панели задач или командой ФОРМАТ-ЯЧЕЙКИ-ВЫРАВНИВАНИЕ-ПО ЦЕНТРУ.

5. Выделите данные таблицы и задайте для них нужный тип и размер шрифта.

6. Отрегулируйте ширину и высоту ячеек. Чтобы изменить ширину столбца, поместите курсор мыши в строку имен столбцов A B C D E … на разделительную линию между столбцами. Когда курсор примет вид двунаправленной стрелки, перетяните границу мышью влево или вправо, чтобы уменьшить или увеличить ширину столбца. Высота строки изменяется аналогично, с использованием столбца имен строк 1 2 3 4 … .

7. Чтобы при выводе на печать таблица была расчерчена линиями, выделите таблицу, раскройте список «Границы» на панели инструментов и щелкните на пиктограмме нужного типа обрамления.

Ввод формул и заполнение пустых столбцов таблицы.

1. Введем данные в столбец «оклад». В ячейку E3 введите формулу =(D3+1)*200 и нажмите Enter. В ячейке появится значение формулы.

2. Чтобы не вводить одну и ту же формулу для всех групп, скопируем ее в остальные ячейки столбца. Для этого щелкните мышью на ячейке с формулой. В правом нижнем углу рамки ячейки появится маленький черный квадрат. Установите на нем курсор мыши. При этом курсор изменит форму на черный крестик. Нажмите на кнопку мыши и растяните рамку, чтобы захватить остальные ячейки столбца. При этом формула для каждой строки изменяется.

3. Для копирования можно также использовать стандартный способ Windows: щелкнуть на ячейке, затем на пиктограмме копирования. Содержимое ячейки запомнится в буфере. Затем выделите область, куда будет копироваться ячейка и щелкните на пиктограмме вставки.

4. Введите данные в столбец «всего начислено». Введите в ячейку G3 формулу =E3*F3/18*1,3. Задайте число знаков после запятой, равное двум. Для этого выделите эту ячейку и выполните команду ФОРМАТ-ЯЧЕЙКИ-ЧИСЛО - ЧИСЛОВОЙ и установите число десятичных знаков 2. Можно также установить денежный формат.

5. Скопируйте формулу в остальные ячейки столбца (см. п. 2).

6. Введите данные в столбец «подоходный налог», вычислив его как 12% от начисленного. Для этого введите в ячейку H3 формулу =G3/100*12.

7. Скопируйте формулу в остальные ячейки столбца (см. п. 2).

8. Введите данные в столбец «профсоюзные сборы», аналогично вычислив 1% от начисленного.

9. Скопируйте формулу в остальные ячейки столбца (см. п. 2).

10. Введите данные в столбец «всего удержано», вычислив сумму налогов и сборов. Для этого введите в ячейку J3 формулу =H3+I3.

11. Скопируйте формулу в остальные ячейки столбца (см. п. 2).

12. Введите данные в ячейки столбца «к выдаче», вычислив разницу начисленного и удержанного. Для этого введите в ячейку K3 формулу =G3-J3.

13. Скопируйте формулу в остальные ячейки столбца (см. п. 2).

14. Посчитайте итоговое количество часов. Для этого выделите ячейку F9, нажмите значок автосуммы и задайте область суммирования, выделив ячейки F3:F8.

15. Аналогичным образом подсчитайте сумму следующих столбцов.

16. Введите данные в ячейки столбца «процент от суммарной выдачи». Для этого введите в ячейку L3 формулу =K3/K9. Знак $ в формуле необходим для того, чтобы при копировании формулы номер ячейки не изменялся.

17. Выразите результат в процентах. Для этого выделите эту ячейку и выполните команду ФОРМАТ-ЯЧЕЙКИ-ПРОЦЕНТ.

18. Скопируйте формулу в остальные ячейки столбца (см. п. 2).

Отсортируйте полученную таблицу.

1. Выделите строки 2-8 таблицы.
2. Выберите пункт меню ДАННЫЕ -СОРТИРОВКА.
3. Выполните сортировку по столбцу «к выдаче» в порядке убывания.

Сохраните файл.

Практическая работа №3. Microsoft Excel. Построение диаграмм.

Порядок выполнения работы

I. В построенной ранее таблице построить гистограмму доходов сотрудников

1. Выделить ячейки с данными для диаграммы. Для этого потребуется выделить 3 области: область с фамилиями A2:A8, область «всего начислено» G2:G8 и область «к выдаче» K2:K8. Чтобы выделить две или более несмежные области, выделите мышью первую область, затем нажмите Ctrl, и не отпуская его, выделяете следующую область.

2. Найти в панели инструментов кнопку с изображением диаграммы и щелкнуть на ней мышью (или вызовите пункт меню Вставка - Диаграммы).

3. Выберите понравившуюся вам гистограмму и нажмите кнопку «Далее».

4. Введите данные для названия диаграммы и меток осей координат. Нажмите кнопку «Готово».

5. Переместите изображение диаграммы на рабочем листе. Используя маркеры вокруг изображения диаграммы, измените размер диаграммы.

II. Построить круговую диаграмму распределения начисленных средств.

Для этого необходимо выделить последовательно (удерживая клавишу Ctrl) все ячейки с заголовками, затем все ячейки с данными.

Например, выделить следующую последовательность ячеек: H2; I2; K2; H5; I5; K5 и затем нажать на кнопку построения диаграммы.

1. Построить график зависимости доходов сотрудников от разряда и количества часов работы.

Для этого необходимо выделить области C2:C8; F2:F8; G2:G8, и выбрать в нестандартных диаграммах логарифмическую или другой вид.

IV. Оформить диаграммы. Построенные диаграммы по умолчанию являются цветными. Для вывода на матричный принтер диаграммы должны быть оформлены в черно-белых тонах. Чтобы изменить цвет графика или фона, активизируйте диаграмму двойным щелчком мыши. Вокруг диаграммы появится мерцающая рамка. Теперь двойной щелчок мыши на элементе диаграммы вызовет диалоговое окно изменения свойств элемента.

V. Сохранить рабочий лист.

VI. Вывести таблицу с диаграммами на печать.

Так как таблица имеет большую ширину, то лист целесообразно развернуть по горизонтали. Для этого вызовите команду ПАРАМЕТРЫ СТРАНИЦЫ-АЛЬБОМНАЯ. На рабочем поле появятся пунктирные линии, показывающие разметку страницы.

Измените ширину столбцов таблицы и размеры диаграмм так, чтобы таблица и диаграммы поместились на одной странице.

Выполните команду ПРОСМОТР, чтобы просмотреть, как будет выглядеть страница при печати.

Закройте окно просмотра и выведите файл на печать командой ПЕЧАТЬ.

Практическая работа №4. Microsoft Excel. Использование мастеров, фильтров.

Добавление данных в таблицу. Сортировка, фильтрация данных.

Откройте свой файл, созданный в EXCEL.

Добавьте данные для двух сотрудников, вставив строки перед строкой «итого». Для этого выделите строку «итого», щелкнув левой кнопкой мыши на ее номере, и примените команду ВСТАВКА → СТРОКИ.

добавьте в таблицу столбец "отдел" и заполните эту колонку названиями отделов так, чтобы число отделов составляло 3-4, число сотрудников в каждом отделе 2-5.

добавьте столбец "материальная помощь", начислите материальную помощь 200 руб. сотрудникам, оклад которых меньше среднего. Среднее значение вычислите с помощью функции СРЗНАЧ. Для начисления материальной помощи используйте функцию ЕСЛИ (используйте при этом мастер функций, который вызывается в меню ВСТАВКА). Добавьте величину материальной помощи в сумму "всего начислено".

задайте фильтр (Данные - Фильтр - Автофильтр) и сделайте видимыми на экране данные, относящиеся к одному отделу.

отсортируйте данные для этого отдела по фамилиям сотрудников в алфавитном порядке.

отсортируйте данные для другого из отделов в порядке возрастания величины всего начислено.

сохраните данные в свою папку на жестком диске компьютера и на дискете.

покажите результаты работы преподавателю.

Содержание отчёта

Титульный лист.

Результаты выполнения работы.

Ответы на контрольные вопросы.

Контрольные вопросы

Назначение электронных таблиц?

Возможности Excel?

Что может содержать ячейка таблицы?

Элементы формулы?

Типы адресации ячейки в формуле?

Практическая работа №5. Система Управления Базами Данных Microsoft Access. Создание таблиц, форм, отчетов.

Ход работы:

Запустите Microsoft Access, создав новую базу данных в каталоге своей группы.

Создайте структуру таблицы. Для этого, откройте вкладку базы данных «Таблицы», нажмите кнопку «Создать» и выберите «Конструктор». Далее, введите перечень и описание столбцов (полей) создаваемой таблицы:

Имя поля	Тип данных	Размер поля
N зачетной книжки	Числовой	Целое
ФИО	Текстовый	30
Курс	Числовой	Байт
Группа	Текстовый	5
Адрес	Текстовый	50
Дата рождения	Дата/время	
Стипендия	Денежный	

Задайте поле «N зачетной книжки» как ключевое, для этого, выделите его и выберите пункт меню Ключевое поле. Закройте таблицу, сохранив ее под именем Студенты.

Создайте форму для набора данных. Для этого, откройте вкладку базы данных «Формы», нажмите кнопку «Создать», выберите «Мастер форм» и источник данных – таблицу «Студенты». Ответьте на вопросы мастера, на каждом шаге нажимая кнопку «Далее»: выберите все поля таблицы (кнопкой >>), внешний вид и стиль формы, задайте ее имя.

Закройте форму, откройте ее в режиме конструктора, нажав кнопку «Конструктор», и отрегулируйте размеры полей, так, чтобы вмещались названия столбцов; закройте форму в режиме конструктора, сохранив изменения.

Наберите данные в таблицу. Для этого, откройте форму кнопкой «Открыть» и внесите 8-10 записей (строк таблицы), т.е. данные о 8-10 человек. Закройте форму, сохранив изменения.

Создайте отчет. Для этого, откройте вкладку базы данных «Отчеты», выберите «Мастер отчетов» и источник данных – таблицу «Студенты». Ответьте на вопросы мастера:

выберите все поля таблицы,

добавьте группировку по полю «Группа», для этого в окне мастера Группировка, выделите поле «Группа» и нажмите кнопку >.

добавьте сортировку по полю «ФИО» в алфавитном порядке, для этого в окне мастера Сортировка, откройте вкладку 1 и выберите поле «ФИО».

добавьте суммарную стипендию в группе, для этого в окне мастера Сортировка, нажмите кнопку «Итоги» и поставьте галочку в окошке Sum Стипендия.

выберите внешний вид и стиль отчета, задайте его имя.

Осуществите предварительный просмотр отчета; если необходимо, отредактируйте его в режиме конструктора; выведите отчет на печать.

Практическая работа №6. Система Управления Базами Данных Microsoft Access. Создание запросов на выборку.

Ход работы:

Откройте свою базу данных в Microsoft Access.

Откройте вкладку базы данных «Запросы».

1. Создайте запрос, выводящий адрес студента с заданными фамилией, именем и отчеством. Для этого, создайте запрос в режиме конструктора, добавив вашу таблицу. В окне конструктора последовательно выберите поля таблицы ФИО и Адрес; запишите в ячейке Условие отбора для поля ФИО - фамилию, имя, отчество студента из вашей таблицы (например, ="Иванов Олег Андреевич"). Для того, чтобы просмотреть вашу таблицу, можно свернуть запрос, открыть вкладку базы данных «Таблицы» и таблицу; после этого следует восстановить окно запроса. Запустите запрос, выбрав в меню Запрос пункт Запуск. Далее, закройте запрос, сохранив его под определенным именем.

2. Создайте запрос, выводящий фамилии в алфавитном порядке и шифры групп студентов, проживающих в определенном населенном пункте. Для этого, снова создайте запрос в режиме конструктора, добавив вашу таблицу. В окне конструктора выберите поля ФИО, Группа и Адрес, задайте порядок сортировки для поля ФИО, уберите для поля Адрес галочку "Вывод на экран" и запишите для него условие отбора (например, Like"г.Курск*" , т.е. адреса, начинающиеся текстом *г. Курск*; в операторе Like следует записать данные из вашей таблицы). Аналогично п.3.1., запустите и закройте запрос.

3. Создайте запрос, выводящий ФИО студентов с номерами зачеток, большими, чем заданное число. Для этого, снова создайте запрос в режиме конструктора, добавив вашу таблицу. В окне конструктора выберите поля ФИО, N зачетной книжки, задайте порядок сортировки для поля N зачетной

книжки и запишите для него условие отбора (например, >1003). Аналогично п.3.1., запустите и закройте запрос.

4. Создайте запрос, выводящий в алфавитном порядке ФИО студентов нескольких заданных групп. Для этого, снова создайте запрос в режиме конструктора, добавив вашу таблицу. В окне конструктора выберите поля ФИО, Группа. Задайте порядок сортировки для поля ФИО, уберите для поля Группа галочку "Вывод на экран" и запишите для него условия отбора (например, ="БУ-01" OR ="БУ-02" , т.е. группа БУ-01 или БУ-02). Аналогично п.3.1., запустите и закройте запрос.

5.Создайте запрос, выводящий ФИО и группы студентов с датой рождения, ограниченной заданными датами. Для этого, снова создайте запрос в режиме конструктора, добавив вашу таблицу. В окне конструктора выберите поля ФИО, Группа, Дата рождения. Для поля Дата рождения запишите условие отбора (например, Between #01.01.82# and #31.12.84# , т.е. 1982-1984 гг.).

6.Создайте запрос, выводящий минимальное, максимальное и среднее значение стипендии. Для этого, снова создайте запрос в режиме конструктора, добавив вашу таблицу. В окне конструктора выберите 3 раза поле Стипендия. Установите с помощью меню Вид групповые операции. Выберите в разных полях конструктора для поля Стипендия таблицы групповые операции Min (минимум), Max (максимум), Avg (среднее). Аналогично п.3.1., запустите и закройте запрос.

7.Создайте запрос, выводящий ФИО студентов определенного курса, получающих стипендию. Для этого, снова создайте запрос в режиме конструктора, добавив вашу таблицу. В окне конструктора выберите поля ФИО, курс, стипендия. Задайте условие отбора для поля курс (например, =1) и условие отбора для поля стипендия >0. Аналогично п.3.1., запустите и закройте запрос.

Покажите результаты выполнения работы преподавателю.

Содержание отчёта

Титульный лист.

Результаты выполнения работы.

Ответы на контрольные вопросы.

Контрольные вопросы

Что такое база данных, СУБД?

Типы СУБД?

Назначение СУБД Access?

Элементы базы данных, их назначение?

Практическая работа №7. Система Управления Базами Данных Microsoft Access. Разработка и создание базы данных.

Задание

1. Разработайте базу данных, состоящую из двух связанных таблиц по интересующей вас тематике
2. Создайте таблицы в режиме конструктора
3. Создайте формы и заполните базы данных содержимым
4. Создайте запросы на выборку информации
5. Создайте отчет базы данных
6. Оформите отчет о проделанный работе

Содержание отчёта

Титульный лист.

Результаты выполнения работы.

Ответы на контрольные вопросы.

Контрольные вопросы

Что такое база данных, СУБД?

Типы СУБД?

Назначение СУБД Access?

Элементы базы данных, их назначение?

Практическая работа №8. СОЗДАНИЕ ПРЕЗЕНТАЦИЙ в MS POWER POINT

Краткие теоретические положения

Один из способов создания презентации состоит в выборе шаблона оформления.

Презентация состоит из последовательности слайдов.

Для добавления слайда нажмите кнопку Создать слайд, выберите подходящую разметку. Слайд может содержать текст, рисунки, таблицу, диаграммы и прочие элементы.

Добавление анимационных эффектов

С помощью команды Настройка анимации из меню Показ слайдов можно установить все анимационные эффекты для слайда. Например, можно обеспечить появление текста по буквам, словам или абзацам. Графические изображения и другие объекты (диаграммы, кино) могут появляться постепенно; также возможна анимация элементов диаграммы. Вы можете изменять порядок возникновения объектов на слайде и устанавливать время показа каждого объекта.

Порядок добавления анимации.

1 В режиме слайдов отобразите слайд, для текста или объектов которого настраивается анимация.

2 Выберите в меню Показ слайдов команду Настройка анимации, затем перейдите на вкладку Время.

3 В группе Объекты без анимации выделите текст или объект, подлежащий анимации, затем выберите Включить.

4 Чтобы показ анимации начинался по щелчку в тексте или объекте, установите переключатель По щелчку мыши.

Для автоматического запуска показа анимации установите переключатель Автоматически.

5 Щелкните вкладку Эффекты. Выберите эффект и звук выберите нужные настройки параметров.

Для изменения порядка появления объектов, в группе Порядок анимации выделите объект, очередность которого изменяется, затем, щелкая одну из стрелок, переместите объект вверх или вниз по списку.

Чтобы предварительно просмотреть анимацию в режиме слайдов, выберите в меню Показ слайдов команду - Просмотр анимации. Чтобы начать показ, щелкните эту миниатюру слайда.

Для изменения порядка слайдов в презентации

1 В режиме структуры перенесите значок слайда на новое место.

2 В режиме сортировщика слайдов перенесите слайд на новое место.

Задание

1. Выберите тему, обзаведитесь материалом
2. Разработайте структуру презентации
3. Создайте не менее 10 слайдов, используя различную разметку
4. Просмотрите презентацию и отредактируйте ее в случае необходимости
5. Оформите отчет о проделанный работе

Содержание отчёта

Титульный лист.

Результаты выполнения работы.

Ответы на контрольные вопросы.

Контрольные вопросы

Что такое Power Point?

Какие вы знаете принципы грамотного размещения информации на слайдах?

Зачем нужна презентация?

Что может содержать слайд, типы разметки?

Список используемых источников

1. Информатика. Базовый курс [Текст] : учебное пособие / под ред. С. В. Симоновича. - 2-е изд. - СПб. : Питер, 2010. - 640 с. : ил. - (Учебник для вузов). (99 экз.)

2. Мотов, В. В. Word, Excel, PowerPoint [Текст] : учебное пособие / В. В. Мотов. - М. : ИНФРА-М, 2012. - 206 с.

3. Кузин, А. В. Разработка баз данных в системе Microsoft Access [Текст] : учебник / А. В. Кузин, В. М. Демин. - 3-е изд. - М. : ФОРУМ, 2012. - 224 с.

4. Колокольникова, А. И. Информатика [Электронный ресурс] : учебное пособие / А. И. Колокольникова, Е. Прокопенко, Л. Таганов. - Москва : Директ-Медиа, 2013. - 115 с. // Режим доступа - http://biblioclub.ru/

5. Борзов, Д. Б. Информатика [Текст] : учебное пособие / Д. Б. Борзов, И. Е. Чернецкая, Е. А. Титенко ; Курский государственный технический университет. - Курск : КурскГТУ, 2007. - 128 с.

6. Борзов, Д. Б. Информатика [Электронный ресурс] : учебное пособие / Д. Б. Борзов, И. Е. Чернецкая, Е. А. Титенко ; Курский государственный технический университет. - Курск : КурскГТУ, 2007. - 128 с.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__» _____ 2022 г.

Информатика. Основы программирования.

Методические указания и задания к выполнению
практических работ
для студентов направления 43.03.03

Курск 2022

УДК 004.2

Составители: В.В. Ефремов, И.Н. Ефремова

Рецензент
Кандидат технических наук, доцент Т.М.Белова

Информатика. Основы программирования: методические указания к выполнению практических работ / Юго-Зап. гос. ун-т; сост.: В.В.Ефремов, И.Н. Ефремова. Курск, 2022. - 157 с.

Содержат описание пяти практических работ по дисциплине «Информатика».

Предназначены для студентов направления 43.03.03

Текст печатается в авторской редакции

Подписано в печать Формат 60х84 1/16.
Усл. печ. л. 1,8. Уч.-изд. л. 1,7. Тираж 100 экз. Заказ 48. Бесплатно 1184
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Содержание

1. Введение в программирование на VBA

1.1. Языки программирования

Язык программирования — искусственный (формальный) язык, предназначенный для записи алгоритмов.

Алгоритм – точное предписание исполнителю совершить определенную последовательность действий для достижения поставленной цели за конечное число шагов.

Язык высокого уровня – согласно ГОСТ 19781-90, язык программирования, понятия и структура которого удобны для восприятия человеком.

Язык низкого уровня (машинный язык) – язык программирования, элементами которого являются команды компьютера. Конструкции машинного языка обрабатываются непосредственно аппаратурой.

Алфавит – фиксированный для каждого алгоритмического языка набор символов, из которых должен состоять любой текст на этом языке. Никакие другие символы в тексте не допускаются.

Любой язык полностью определяется совокупностью правил синтаксиса и семантики.

Семантика – система правил истолкования отдельных языковых конструкций. Семантика определяет смысловое значение предложений языка.

Синтаксис – набор правил построения фраз алгоритмического языка, позволяющий определить осмысленные предложения в этом языке.

Каждая инструкция представляет собой предложение языка и должна соответствовать определённым для неё правилам. Синтаксис приводится при описании того или иного типа инструкций и записывается в форме Бэкуса-Наура с использованием ряда условных обозначений. Например, синтаксис оператора ветвления выглядит так:

```
if <условие> then <инструкция then> [else
<инструкция else>]
```

Обязательные слова, которые в конкретных инструкциях должны быть записаны так же, как и при описании синтаксиса (ключевые

слова), не выделяются никак: в данном случае это слова *if, then, else.* Обязательная изменяемая часть инструкции, в которую при её использовании записываются необходимые слова, выделяются угловыми скобками. В данном случае это *<условие>, <инструкция then>, <инструкция else>*. Необязательная часть инструкции записывается в квадратных скобках: *[else <инструкция else>]*. Таким образом, приведённая запись синтаксиса означает следующее: начинается инструкция обязательно со слова *if*. После *if*, через пробел, обязательно должно быть записано какое-либо *условие*, после которого, так же через пробел, обязательно записывается слово *then*, после которого, опять же через пробел, обязательно записывается *инструкция then*. После этого следует необязательный блок, но если он применяется, то должен быть записан следующим образом: после *инструкции then* через пробел должно идти слово *else*, после которого, через пробел, обязательно должна идти *инструкция else*.

Трансляция – процесс перевода программы с языка высокого уровня на язык процессора.

Компиляция – трансляция, результатом которой является исполняемый файл программы.

Интерпретация – трансляция с одновременным выполнением. В таком случае прежде, чем программа будет выполнена, необходимо, чтобы был запущен интерпретатор языка высокого уровня (VBA – интерпретатор, VB из Visual Studio – компилятор). Исполняемый файл программы в процессе интерпретации, как правило, не создаётся.

Интегрированная среда разработки программ – приложение, объединяющее совокупность программных средств, обеспечивающих пользователю возможность создания, отладки, трансляции и выполнения разрабатываемых программ. Для VBA интегрированной средой разработки является приложение, называемое «Редактор VBA».

Проект – объект, представляющий собой документ или шаблон Word. Может включать в себя текст документа, модули, формы и ссылки.

Модуль – часть документа или шаблона, в которой сохраняются VBA-макросы.

VBA-макрос – программа на языке Visual Basic, позволяющая автоматизировать работу в приложениях MS Office и расширить возможности MS Office путем реализации обработки данных документа по определённому алгоритму.

1.2. Объекты

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

Объектно-ориентированное программирование – методика анализа, проектирования и написания приложений с помощью объектов.

Объект – объединение данных и кода, предназначенного для их обработки в нечто целое.

Семейство (**collection**) – объект, содержащий несколько других объектов, как правило, одного и того же типа. Например, объект *workbooks* (рабочие книги) содержит все открытые объекты *workbook* (рабочая книга). Каждый элемент семейства нумеруется и может быть идентифицирован либо по номеру, либо по имени. Например, *worksheets(1)* обозначает первый рабочий лист активной книги, а *worksheets("Лист1")* – рабочий лист с именем «Лист1».

Наследование – в ООП, свойство объекта, заключающееся в том, что характеристики одного объекта (объекта-предка) могут передаваться другому объекту (объекту-потомку) без их повторного описания. Наследование упрощает описание объектов.

Полиморфизм – в ООП, способность объекта выбирать правильный метод в зависимости от типа данных, полученных в сообщении.

Инкапсуляция – в ООП, сокрытие внутренней структуры данных и реализации методов объекта от остальной программы. Другим объектам доступен только интерфейс объекта, через который осуществляется все взаимодействие с ним.

Иерархия объектов. Объектная библиотека VBA располагает несколькими сотнями объектов, находящихся на различных уровнях

иерархии. Иерархия определяет связь между объектами и показывает пути доступа к ним.

Информацию об иерархии объектов приложения можно почерпнуть из справки (Рис.1.1). Например, в MS Word: *справка->справка по программированию->MS Word VB Reference-> MS Word Object Model*. Аналогично в структуре справки MS Excel и других приложений MS Office. Помимо этого в справке приложения можно найти примеры использования каждого объекта и приёмы программирования для решения различных задач.

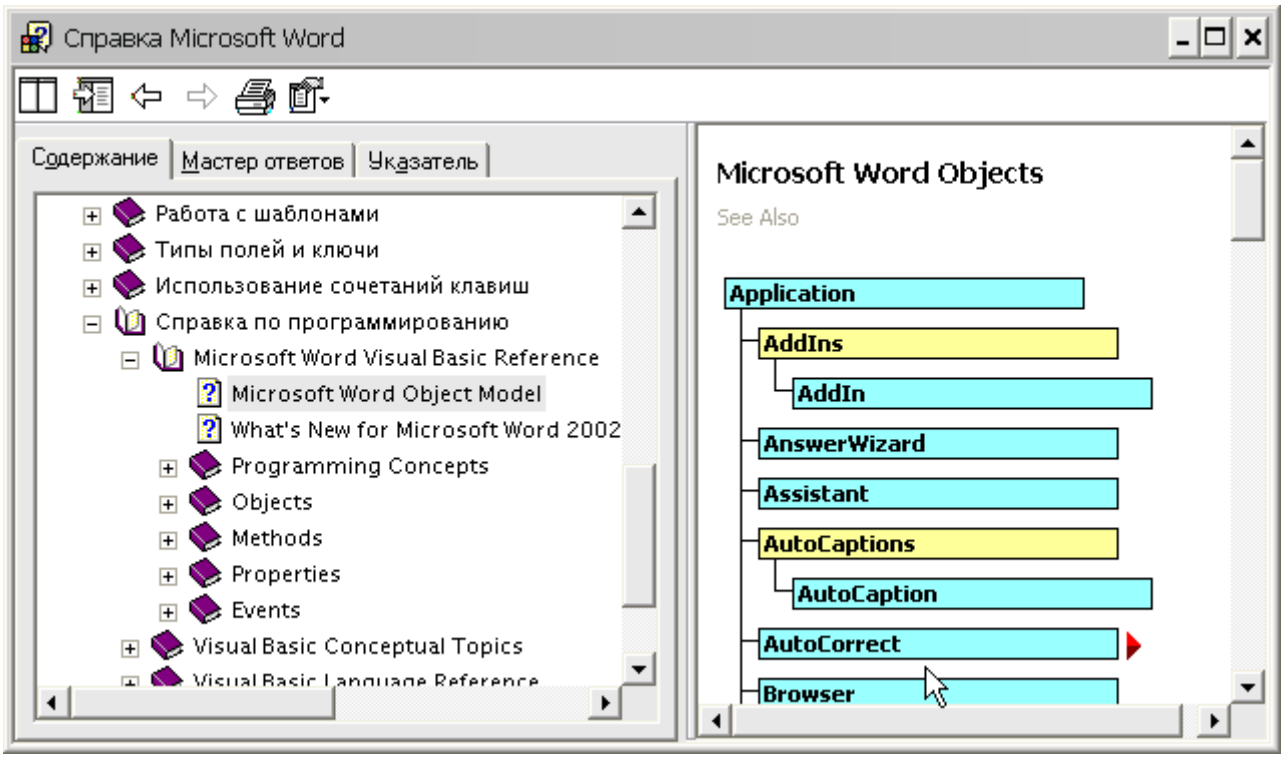


Рис.1.1. Окно справки для объектной модели MS Word

Объектная модель приложения MS Office представляет собой то, что отражено в названии: в рамках этой модели в форме иерархической зависимости отражены все объекты, составляющие запущенное приложение, такие как само приложение – объект `Application`, его свойства, надстройки `(AddIns)`, документы `(Documents)`, параметры документов (`Name` – имя, `Password` – пароль, `PageSetup` – свойства страниц, и т.п.), объекты документов (`Paragraphs` – абзацы, `Lists` – списки, `Shapes` – рисунки и т.п.), параметры этих объектов и так далее до последней буковки.

Доступ к нужному объекту, представленному в модели, осуществляется по ссылке, в которой используются имена объектов. Полная ссылка на объект состоит из ряда имен вложенных последовательно друг в друга объектов. Разделителями имен объектов в этом ряду являются точки, ряд начинается с объекта *Application* и заканчивается именем самого объекта. Например, полная ссылка на ячейку «A1» рабочего листа «Лист1» рабочей книги с именем «Архив» имеет вид:

```
Application.Workbooks ("Архив").Worksheets
("Лист1").Range ("A1").
```

Приводить каждый раз полную ссылку на объект не обязательно. Обычно достаточно ограничиться только неявной ссылкой на объект. В неявной ссылке, в отличие от полной, объекты, которые активны в данный момент, как правило, можно опускать. В рассмотренном случае, если ссылка на ячейку «A1» дана в программе, выполняемой в среде Excel, то ссылка на объект *Application* может быть опущена, т. е. достаточно привести относительную ссылку:

```
Workbooks ("Архив").Worksheets ("Лист1").Range
("A1")
```

Если рабочая книга «Архив» является активной, то ссылку можно записать еще короче:

```
Worksheets ("Лист1").Range ("A1")
```

Если и рабочий лист «Лист1» активен, то в относительной ссылке вполне достаточно ограничиться упоминанием только диапазона «A1»:

```
Range ("A1")
```

Методы – части объекта, выполняющие действия над данными объекта. Синтаксис применения метода:

```
<Имя объекта>.<Имя метода>
```

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

```
Application.Quit
```

Метод можно применять ко всем объектам семейства. В данном примере к семейству *chartobjects* (диаграммы) рабочего листа «Лист1» применен метод *Delete* (удалить), который приводит к удалению всех диаграмм с рабочего листа «Лист1»:

```
Worksheets ("Лист1").Chartobjects.Delete
```

Свойства – атрибуты объекта, определяющие его характеристики, такие как размер, цвет, положение на экране и состояние объекта (например, доступность или видимость). Чтобы изменить характеристики объекта, надо просто изменить значения его свойств. Синтаксис установки значения свойства:

```
<Имя объекта>.<Имя свойства> = <Значение
свойства>
```

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

```
Application.Caption = "Пример"
```

Свойство можно изменять сразу у всех объектов семейства. В приведенном ниже примере с помощью установки свойству *visible* (видимость) значения *False* (ложь) все рабочие листы активной книги (семейство объектов *worksheets*) скрываются:

```
Worksheets.Visible = False
```

Среди свойств особое место занимают свойства, возвращающие объект, такой, как открытый документ, параграф документа или ячейку таблицы.

1.3. События

Событие представляет собой действие, распознаваемое объектом (например, щелчок кнопкой мыши, нажатие клавиши или вызов программы на VBA из меню макросов), для которого можно запрограммировать отклик. События возникают в результате действий пользователя или программы или же могут быть вызваны системой.

Суть программирования на VBA как раз и заключается в этих двух понятиях: событие и отклик на него. Если пользователь производит какое-то воздействие на систему, скажем, нажимает кнопку, тогда в качестве отклика выполняется код созданной пользователем

процедуры. Если такой отклик не создан, т. е. не написана соответствующая процедура, то система никак не реагирует на данное событие, и оно остается безответным.

Таким образом, действия, происходящие в системе, являются событиями, а отклики на них – процедурами. Этот специальный вид процедур, генерирующих отклик на события, называется процедурами обработки событий. В целом программирование на VBA состоит в создании кода программ, которые генерируют прямо или косвенно отклики на события.

Лабораторная работа №1

Цель работы: изучение интегрированной среды разработки программ на языке Visual Basic for Application (VBA).

Порядок выполнения работы

1. Запустите Microsoft Word.
2. Создайте новый документ и дайте ему имя, выбрав пункт *Файл -> Сохранить как* (Рис. 1.2.).

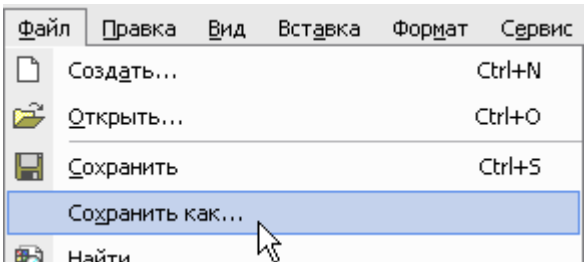


Рис. 1.2. Меню создания и сохранения докумен-

3. Для запуска встроенного редактора VBA выберите пункты *Сервис -> Макрос -> Редактор Visual Basic* (Рис. 1.3**Рис. 1.**), или нажмите *Alt-F11*.

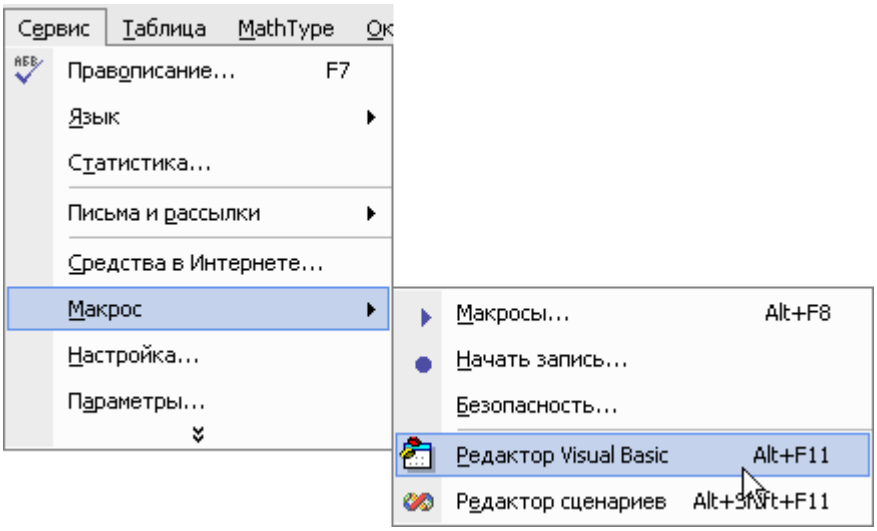


Рис. 1.3. Меню запуска редактора VBA

4. В редакторе Visual Basic при запуске отображается окно «Менеджер проектов» и окно «Свойства объектов» (Рис. 1.4).

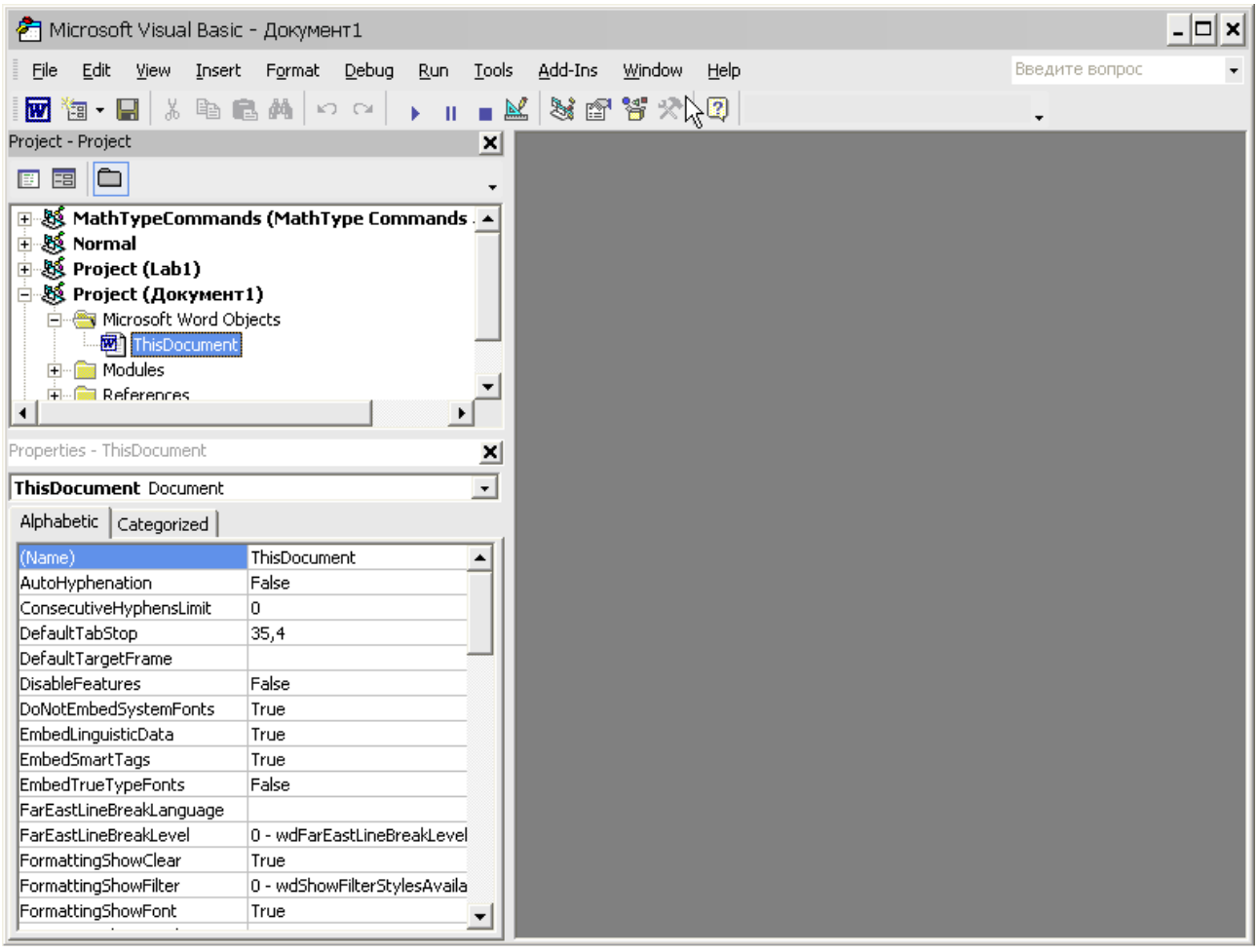


Рис. 1.4 Окно редактора VBA после запуска

4.1 Окно менеджера проектов содержит дерево открытых в данный момент шаблонов и документов (Рис. 1.5), и, как минимум, там должны присутствовать:
– шаблон *normal.dot*, открываемый при запуске Word по умолчанию
– открытые в данный момент документы.

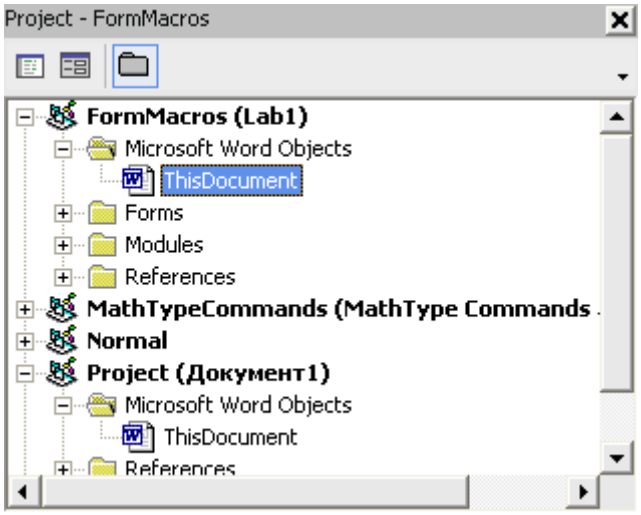


Рис. 1.5. Менеджер проектов.

4.2 Окно «Свойства объектов» (Рис. 1.6) содержит свойства объекта текущего выбора из окна менеджера проектов.

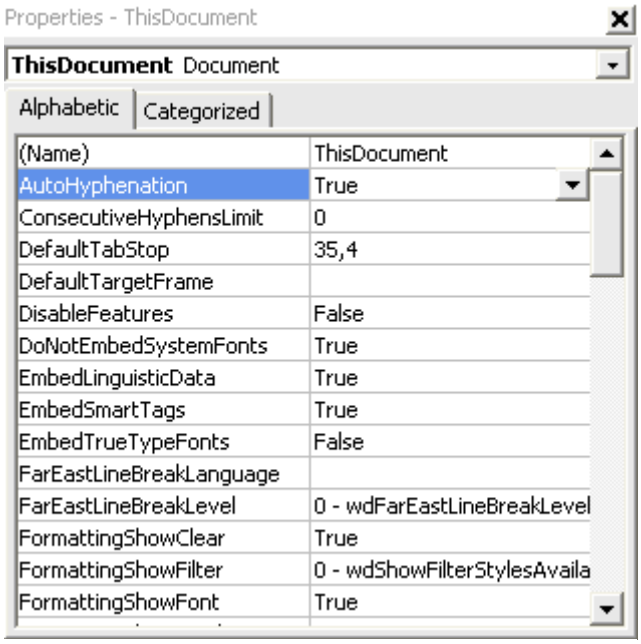


Рис. 1.6. Менеджер свойств

5. Для того чтобы написать программу на Visual Basic, необходимо вставить модуль в документ.

5.1 Выберите проект, содержащий созданный вами документ (имя документа содержится в скобках) (Рис. 1.7).

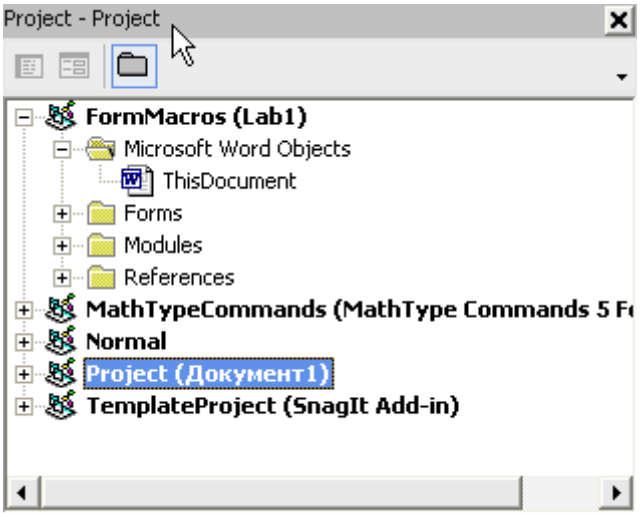


Рис. 1.7. Выбор нужного проекта в менеджере проектов

5.2 Выберите команды *Insert -> Module* в меню редактора Visual Basic (Рис. 1.8), или, нажав на соответствующем проекте в менеджере проектов правой кнопкой мыши, выберете соответствующие команды в контекстном меню (Рис. 1.9). Редактор добавит новый модуль в проект и откроет окно «Редактор кода» (Рис. 1.10) созданного модуля, в котором можно написать программу.

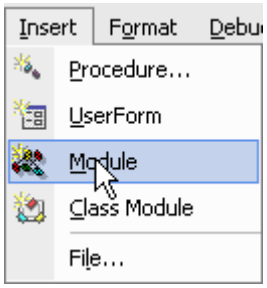


Рис. 1.8 Вставка нового модуля из меню редактора VBA

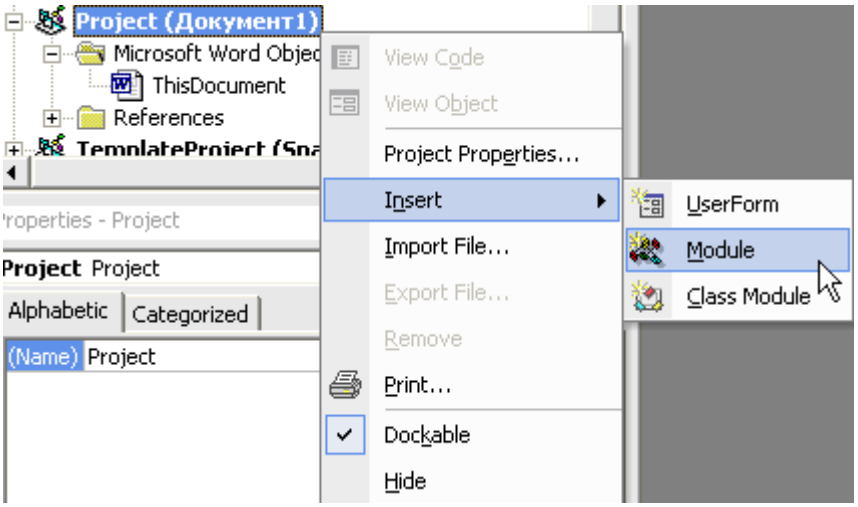


Рис. 1.9. Вставка нового модуля в проект с помощью контекстного меню

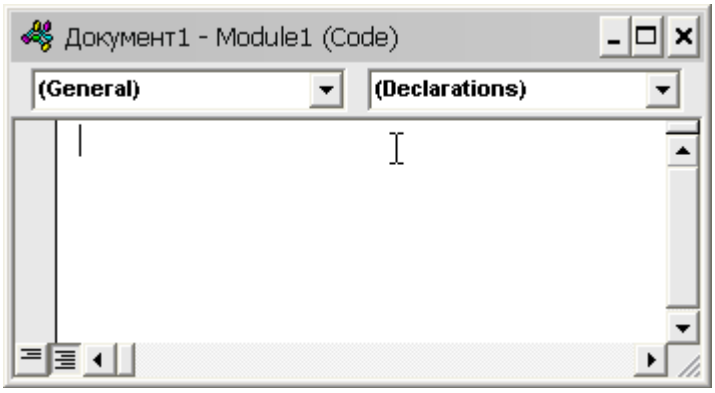


Рис. 1.10. Окно редактора кода.

5.3 Переименуйте новый модуль, дав ему имя *HelloWorld*. Для этого:

– В менеджере проектов выберите новый модуль (Рис. 1.11);
– В окне «Свойства» измените свойство *Name* (Рис. 1.12).

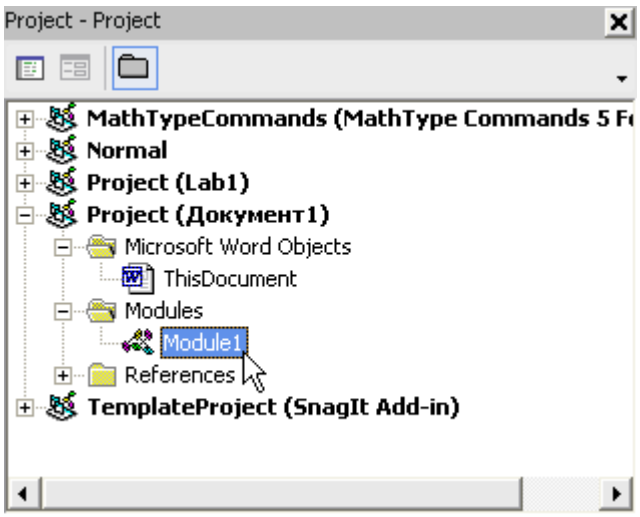


Рис. 1.11. Выбор модуля в менеджере проектов

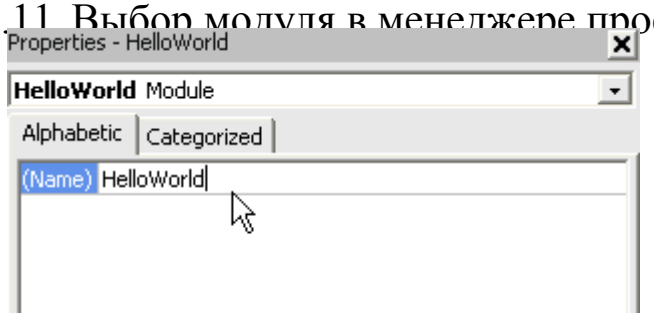


Рис. 1.12. Изменение имени модуля

6. Напишите первую программу «Hello World»:
   6.1 Перейдите в окно кода;
   6.2 Введите следующие строки:

```
Sub HelloWorld ()
  MsgBox "Hello, World!"
End Sub
```

На экране это будет выглядеть примерно так, как показано на Рис. 1.13.

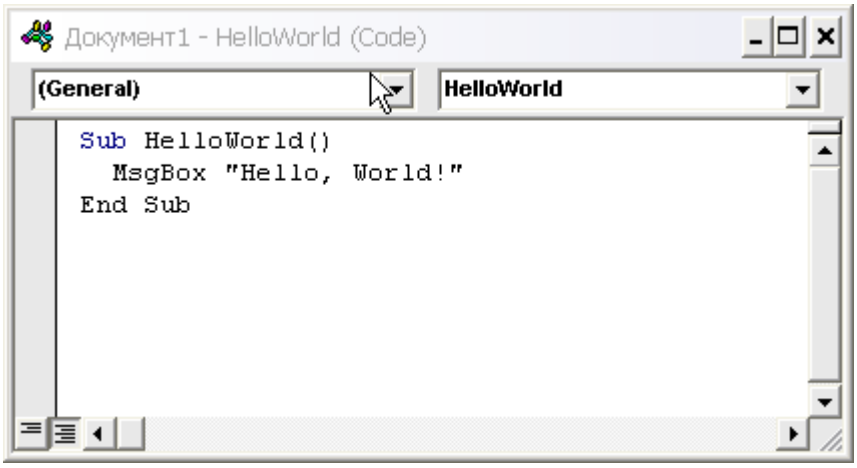


Рис. 1.13. Окно редактора кода с текстом программы

Строка `Sub HelloWorld()` означает начало процедуры с именем HelloWorld, которая не использует внешних параметров.

– Строка `End Sub` означает окончание процедуры и появляется автоматически.
– Команды, располагаемые между этими строками, составляют тело процедуры и выполняются последовательно.
– В данном случае тело процедуры состоит из одной команды. Команда `MsgBox "Hello, World!"` приводит к выводу на экран окна с надписью «Hello, World!».
– После набора `MsgBox` редактор VBA выведет подсказку с перечислением параметров, которые можно передавать в процедуру `MsgBox` (Рис. 1.14).

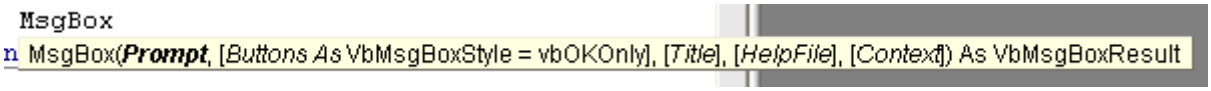


Рис. 1.14. Всплывающая подсказка

7. Запустите программу на выполнение, выбрав пункт меню *Run -> Run Sub/UserForm* (Рис. 1.15), или нажав кнопку ▸ на панели инструментов.

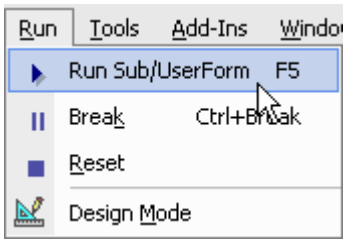


Рис. 1.15. Запуск программы из меню VBA

7.1 По команде `MsgBox "Hello, World!"` в программе будет выведено окно, содержащее выводимый текст (Рис. 1.16).

Рис. 1.16. Окно MsgBox

7.2 Перейдите в созданный документ.

7.3 Выберите пункт меню *Сервис -> Макрос -> Макросы* (Рис. 1.17)

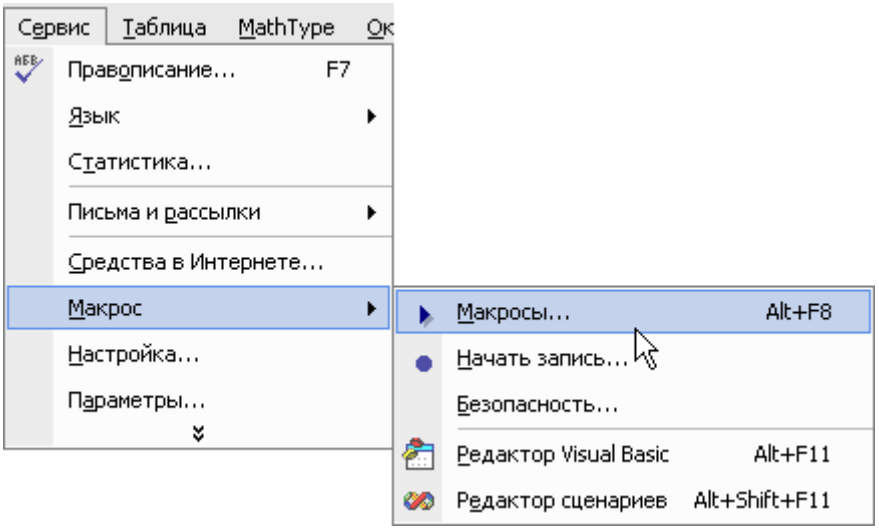


Рис. 1.17. Открытие списка макросов в меню MS

7.4 В списке макросов выберите *HelloWorld* (Рис. 1.18) и нажмите на кнопку выполнить.

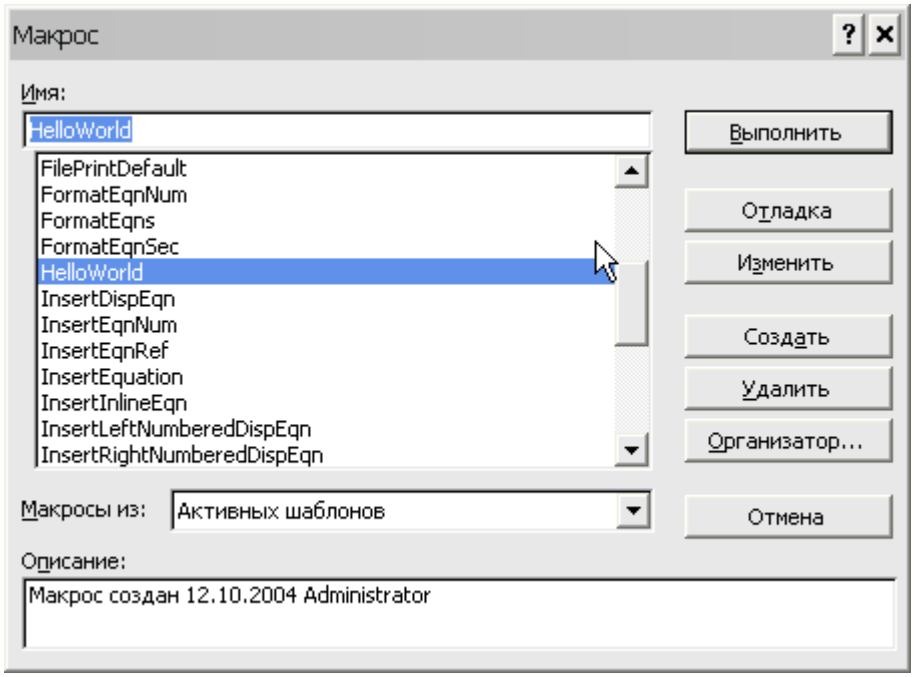


Рис. 1.18. Выбор нужного макроса из списка

7.5 Выполнится написанная программа (Рис. 1.16).

8. Измените вторую строку на

```
MsgBox "Hello, World!",vbYesNo,"Окно приветствия"
```

Выполните программу, посмотрите, как изменился результат.

9. Сохраните документ. Вместе с ним сохранится всё, что входит в проект. Чтобы сохранить модуль отдельно от документа, воспользуйтесь пунктом меню *File -> Export File…*

10. Напишите программу с использованием объектной модели MS Word.

10.1 Создайте в имеющемся модуле процедуру без параметров.

10.2 Введите в тело процедуры строку

```
ThisDocument.PageSetup.LeftMargin = 300
```

(Можно воспользоваться появляющимся списком (Рис. 1.19)).

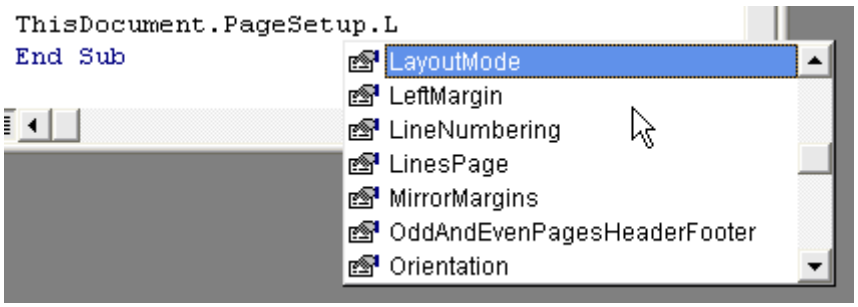


Рис. 1.19. Список свойств объекта

– *ThisDocument* означает обращение к документу проекта (вы можете это видеть в окне менеджера проектов (Рис. 1.20)).

Рис. 1.20. Объекты проекта

– *PageSetup* означает обращение к параметрам страницы текущего документа.
– *LeftMargin* – параметр «отступ слева».
– Параметру присваивается значение *300* (пунктов).

10.3На следующей строке наберите команду

```
ThisDocument.Activate
```

для активизации документа проекта.

10.4Наберите команду

```
Selection.TypeText "Hello, world"
```

– *Selection* – обращение к выделенному фрагменту документа или к позиции курсора.
– *TypeText* – процедура вывода строки текста. Строка текста записывается в качестве параметра.

11. Запустите программу и убедитесь, что с документом проекта была выполнена запрограммированная вами последовательность действий.

Контрольные вопросы

1. Определите понятие «Язык программирования VBA».
2. Охарактеризуйте язык программирования VBA.
3. Какие возможности предоставляет интегрированная среда разработки программ на VBA?
4. Как описывается синтаксис инструкций?
5. Как программа на VBA может получить доступ к данным, открытым в приложении MS Office?
6. Что общего и чем различаются свойства и методы объекта?
7. Какое событие может привести к запуску программы на VBA?
8. Что такое «проект» в MS Office?
9. Зачем нужны окно свойств и окно менеджера проектов в редакторе VBA?
10. В каком месте вводится программа на VBA?
11. Как сохранить программу, написанную на VBA?
12. Что же произошло в пункте 6 при выполнении работы?
13. Каков должен быть результат выполнения инструкции, записанной в пункте 8?
14. Создайте схемы алгоритмов для программ, написанных в данной лабораторной работе.

2. Реализация линейных алгоритмов в VBA

Алгоритм – последовательность действий, приводящая к результату.

Линейным называется алгоритм, все действия которого обязательно должны быть выполнены в определённой последовательности одно за другим.

2.1. Данные в VBA

Автоматизация обработки данных с помощью компьютерной программы предполагает три стадии выполнения: ввод данных, обработка данных и вывод данных. Для сохранения, анализа и преобразования данных внутри программы предусмотрен ряд механизмов: обработка типов, организация временного и постоянного запоминания, создание структур.

Типы данных

Типы данных относятся к фундаментальным понятиям любого языка программирования. **Тип данных** определяет множество допустимых значений, объём памяти, необходимый для хранения значения, множество допустимых операций и способ двоичного кодирования.

Типы данных делятся на **простые**, рассчитанные на хранение одного элемента данных, и **структурированные**, рассчитанные на хранение множества элементов одного или нескольких типов. Простые типы можно сохранить в виде *переменной* или *константы*, к структурированным типам относятся *массивы, записи, файлы*.

Встроенные типы данных VBA отображены в Таблица 2.1. Среди них можно выделить целочисленные типы, числа с плавающей точкой, логический тип, тип дата/время и строковый тип. При использовании в программе значений этих типов строки записываются в кавычках `("строка")`, дата и время – в решётках `(#May 17, 1960#)`, числа и логические значения – без дополнительных обозначений. В качестве десятичного разделителя используется точка.

Таблица 2.1. Типы данных в VBA

Название	Размер	Диапазон значений
`Byte`	1	Целое число от 0 до 255
`Integer`	2	Целое число от -32768 до 32767
`Long`	4	Целое число от -2147483648 до 2147483647
`Decimal`	14	Масштабируемое целое число +/-79228162514264337593543950335 с 28 знаками справа от запятой; минимальное ненулевое значение имеет вид +/-0,0000000000000000000000000001
`single`	4	Число с плавающей запятой обычной точности: от -3.402823E38 до -1,401298E-45 от 1.401298E-45 до 3,402823E38
`Double`	8	Число с плавающей запятой двойной точности от -1.79769313486232E308 до -4.94065645841247E-324; от 4.94065645841247E-324 до 1.79769313486232E308.
`Currency`	8	Деньги от -922337203685477,5808 до 922337203685477,5807
`Boolean`	2	Логический: True или False
`Date`	8	Даты и время от 1.01.100 г. до 31.12.9999 г.
`Object`	4	Любой указатель объекта
`string`	10 + длина строки	Строка переменной длины от 0 до приблизительно 2 миллиардов
`string`	Длина строки	Строка постоянной длины от 1 до 65400
`Variant`	16	Числовые подтипы: любое числовое значение до границ диапазона типа *`Double`*
`Variant`	22 + длина строки	Строковые подтипы – как для строки (string) переменной длины
Тип, определяемый пользователем.	Объем определяется элементами	Определяется с помощью ключевого слова `Type`. Диапазон каждого элемента определяется его типом данных

Переменные

Переменная – именованная часть памяти, в которую могут помещаться разные значения переменной, причем в каждый момент времени переменная имеет единственное значение. В процессе выполнения программы значение переменной может изменяться. Тип переменных определяется типом данных, которые они представляют.

В VBA переменные можно не описывать. При выполнении программы переменная создаётся в момент появления в тексте и принимает тип, соответствующий значению, которое этой переменной присваивается в момент первого появления (в VBA – тип *Variant*).

Описание типа каждой переменной делает программу надежнее за счёт снижения уровня беспорядка и ускоряет ее работу, так как VBA не требуется тратить время на распознавание типа неописанной переменной при каждом обращении к ней. Для предотвращения использования неописанных переменных следует в начало модуля поместить инструкцию *Option Explicit*.

Инструкция *Dim* предназначена как для описания переменных, так и массивов.

Синтаксис

```
Dim [WithEvents] <ИмяПеременной> [([<Индексы>])]
[As [New] <Тип>] [,[WithEvents] <ИмяПеременной>
[([<Индексы>])] [As [New] <Тип>]]...
```

Аргументы

WithEvents – ключевое слово, указывающее, что аргумент *ИмяПеременной* является именем объектной переменной, которая используется при отклике на события, генерируемые объектом ActiveX (то есть объектом, который может быть открыт для других приложений и средств программирования).

ИмяПеременной – имя переменной, удовлетворяющее стандартным правилам именования переменных.

Индексы – размерности переменной массива; допускается описание до 60 размерностей. Для задания аргумента индексы используется следующий синтаксис:

```
[<Нижний> To] <Верхний> [, [<Нижний> To]
<Верхний>]
```

Если нижний индекс не задан явно, нижняя граница массива определяется инструкцией

```
Option Base <0|1>
```

и может быть равна 0 или 1. Если отсутствует инструкция *Option Base*, нижняя граница массива равняется нулю.

New – ключевое слово, включающее возможность неявного создания объекта. Если указано ключевое слово *New* при описании объектной переменной, новый экземпляр объекта создается при первой ссылке на него, поэтому нет необходимости присваивать ссылку на объект с помощью инструкции *Set*.

Тип – тип данных переменной. Для каждой описываемой переменной следует использовать отдельное предложение *As <Тип>*.

Инструкция *Dim* предназначена для описания типа данных переменной для различных областей видимости. Переменные, описанные с помощью ключевого слова *Dim* на уровне модуля, доступны для всех процедур в данном модуле. Переменные, описанные на уровне процедуры, доступны только в данной процедуре.

Например, следующая инструкция описывает переменную с типом *integer*:

```
Dim N As Integer
```

Инструкция *Dim* предназначена также для описания объектного типа переменных. Например, следующая инструкция описывает переменную для нового экземпляра рабочего листа:

```
Dim X As New Worksheet
```

Если при описании объектной переменной не используется ключевое слово *New*, то для использования объекта, на который ссылается переменная, существующий объект должен быть присвоен переменной с помощью инструкции *Set*.

Инструкция DefТип

Инструкция *DefТип* (вместо *Тип* в имени инструкции фигурируют буквы, обозначающие конкретный тип данных) используется

на уровне модуля для задания типа данных по умолчанию для переменных, аргументов, передаваемых в процедуры, и значений, возвращаемых процедурами *Function* и *Property Get*, имена которых начинаются с соответствующих символов (

Таблица 2.2).

Таблица 2.2. Задание типов данных по умолчанию

Синтаксис	Описание
DefBool <ДиапБукв>[, <ДиапБукв>]...	Тип Boolean
DefByte <ДиапБукв>[, <ДиапБукв>]...	Тип Byte
Deflnt <ДиапБукв>[, <ДиапБукв>]...	Тип Integer
DefLng <ДиапБукв>[, <ДиапБукв>]...	Тип Long
DefCur <ДиапБукв>[, <ДиапБукв>]...	Тип Currency
DefSng <ДиапБукв>[, <ДиапБукв>]...	Тип Single
DefDbl <ДиапБукв>[, <ДиапБукв>]...	Тип Double
DefDate <ДиапБукв>[, <ДиапБукв>]...	Тип Date
DefStr <ДиапБукв>[,<ДиапБукв>]...	Тип String
DefObj <ДиапБукв>[,<ДиапБукв>]...	Тип Object
DefVar <ДиапБукв>[,<ДиапБукв>]...	Тип Variant

Аргумент *ДиапБукв* имеет следующий синтаксис:

```
Буква1[-Буква2]
```

Аргументы *Буква1* и *Буква2* указывают границы диапазона имен, для которых задается тип данных по умолчанию.

В следующем примере инструкция устанавливает, что все переменные с именами, начинающимися с букв из диапазона от A до Q, имеют строковый тип:

```
DefStr A-Q
```

Инструкция *DefТип* действует только на модуль, в котором она используется.

При указании диапазона букв обычно определяется тип данных по умолчанию для переменных, которые начинаются с первых 128 символов набора. Однако при указании диапазона *A-Z* задается тип данных по умолчанию для всех переменных, включая те, что начинаются с международных символов из расширенной части набора (128—255).

Еще одним способом задания типа переменной по умолчанию является включение в конец имени специального символа, устанавливающего тип переменной (Таблица 2.3).

Таблица 2.3. Символы для задания типов по умолчанию

Символ	Тип
*	Integer
&	Long
|	Single
#	Double
@	Currency
$	String

Константы

Константа – именованная часть памяти, хранящая определённое значение, которое в процессе работы программы не может быть изменено.

Синтаксис

```
[Public|Private] Const <ИмяКонстанты> [As <Тип>]
= <Значение>
```

Аргументы

Public – ключевое слово, используемое на уровне модуля для описания констант, доступных всем процедурам во всех модулях. Не допускается в процедурах.

Private – ключевое слово, используемое на уровне модуля для описания констант, доступных только внутри модуля, в котором выполняется описание. Не допускается в процедурах.

Тип данных, определенный пользователем

Наряду с массивами, представляющими нумерованный набор элементов одного типа, существует еще один способ создания структурного типа – тип, определенный пользователем, или в привычной терминологии для программистов – запись.

Запись – это совокупность нескольких элементов, каждый из которых может иметь свой тип. Элемент записи называется **полем**.

Запись является частным случаем класса, в котором не определены свойства и методы.

Синтаксис

```
[Private|Public] Type <ИмяПеременной>
  <ИмяЭлемента> [([<Индексы>])] As <Тип>
  [<ИмяЭлемента> [([<Индексы>])] As <Тип>]
End Type
```

Аргументы

Public – используется пользователем для описания определяемых типов, которые доступны для всех процедур во всех модулях всех проектов.

Private – используется для описания определяемых пользователем типов, которые доступны только в модуле, в котором выполняется описание.

Тип – Тип данных элемента; поддерживаются типы: *Byte, Boolean, Integer, Long, Currency, Single, Double, Date, String* (для строк переменной длины), *String * <длина>* (для строк фиксированной длинны), *Object, variant* и другой, определяемый пользователем тип или объектный тип.

В данном примере инструкция *Type* используется для определения типа данных (только на уровне модуля). При появлении в модуле класса инструкции type должно предшествовать ключевое слово *Private*:

```
' Тип, определенный пользователем
Type Student 'Описание типа «Студент»
  Фамилия As String * 20
  Имя As String * 20
  Отчество As String * 20
  НомерЗачетки As Integer
  Группа As String * 10
  Курс As Long
  ДатаРождения As Date
End Type
```

```
' Описание переменной типа «Студент»
Dim s1 As Student
'Обращение к элементам переменной типа «Студент»
s1.Фамилия = "Иванов"
s1.НомерЗачетки = 12003
s1.Группа = "Менеджмент"
```

Допустимые имена

В VBA пользователь определяет имена переменных, функций, процедур, типов, постоянных и других объектов. Вводимые пользователем имена должны отражать суть обозначаемого объекта так, чтобы делать программу легко читаемой. В VBA имеются следующие ограничения на имена:

- Длина имени не должна превышать 255 символов.
- Имя не может содержать пробелов и следующих символов: % . , & ! # @ $
- Имя может содержать любую комбинацию букв, цифр и символов, начинающуюся с буквы.
- Имена должны быть уникальны внутри области, в которой они определены.
- Нельзя использовать имена, совпадающие с ключевыми словами VBA и именами встроенных функций и процедур.

Хотя регистр букв (верхний или нижний) в имени не имеет значения, умелое использование его может существенно облегчить понимание содержательной стороны переменной.

2.2. Инструкции VBA

Инструкция VBA представляет собой полную команду языка VBA. Она может содержать ключевые слова, операторы, переменные, константы и выражения. В VBA имеются следующие категории инструкций:

- инструкции описания;
- исполняемые инструкции;
- инструкции присваивания.

Инструкции описания предназначены для определения объектов и структуры программы, режимов её работы, и не предназначены для

выполнения действий. Примеры таких инструкций: *Option Explicit* для включения режима обязательного описания переменных, *Sub* для описания начала процедуры, *Dim* для описания переменных, *Type* для описания типа пользователя, *If Then Else* для описания алгоритмической структуры «Ветвление» и тому подобные.

Исполняемые инструкции предназначены для обозначения действий. Примеры исполняемых инструкций: *MsgBox* для вывода строки в диалоговое окно, *Selection.Typetext* для вывода строки в документ, *Application.Quit* для закрытия приложения.

Инструкции присваивания используются для записи данных в подходящие области памяти. Рассмотрим инструкции присваивания подробно.

Оператор присваивания

Оператор присваивания записывает подготовленные данные в область памяти, соответствующую переменной или свойству объекта. Для присваивания значения выражения переменной или свойству объекта применяется ключевое слово *Let*.

Синтаксис

```
[Let] <Имя переменной или свойства> = <Выражение>
```

Ключевое слово *Let* необязательно и чаще всего опускается. Оператор присваивания предписывает выполнить выражение, заданное в его правой части, и присвоить результат переменной, имя которой указано в левой части. В результате, например, действия следующей пары операторов

```
x = 2
x = x + 2
```

переменной *x* будет присвоено 4.

Для присваивания объектной переменной ссылки на объект применяется ключевое слово *Set*.

Синтаксис

```
Set <ОбъектнаяПеременная> = [New]
<Объект>|Nothing
```

Аргументы

New используется при создании нового экземпляра класса.

Nothing позволяет освободить все системные ресурсы и ресурсы памяти, выделенные для объекта, на который имелась ссылка (вольно говоря, удаляет объект из памяти).

В следующем примере инструкция *Set* присваивает переменной *Область* диапазон *A1:B3*:

```
Set Область = Range ("A1:B3")
```

Совместимость типов

Переменной заданного типа можно присваивать только такое значение, которое соответствует этому типу. Например, если была описана переменная *a* типа *Integer*, то этой переменной нельзя корректно присвоить значение переменной *b* типа *Single*, поскольку VBA не сможет правильно отобразить значение с плавающей точкой на множество целых чисел. Выполнение инструкции

```
a = b
```

приведёт к тому, что в переменной *a* будет сохранено округлённое до целого значение переменной *b*, что может привести к ошибке при дальнейшем использовании этого значения. Кроме того, в связи с тем, что в переменной типа *Single* может быть сохранена большая величина, чем диапазон допустимых значений типа *Integer*, выполнение этой инструкции может вызвать ошибку переполнения в процессе выполнения программы.

С другой стороны, обратное отображение может быть выполнено вполне корректно, и переменной типа *Single* всегда может быть присвоено значение переменной типа *Integer*.

```
b = a
```

всегда будет корректно выполняться. Так, разные типы могут быть совместимы, если учесть размер, занимаемый ими в памяти, диапазон значений и воспользоваться здравым смыслом.

В случае необходимости присвоить переменной одного типа значение другого типа, нужно воспользоваться соответствующей функцией преобразования типов. При этом такое преобразование не

спасает от ошибок переполнения. Функции преобразования типов будут рассмотрены ниже.

Форматирование строк программы

Каждая строка программы на VBA воспринимается как инструкция. Иногда бывает удобно разбить одну инструкцию на несколько строк или разместить в одной строке несколько инструкций.

Расположение символов (пробел) и (Знак подчеркивания) в конце строки обеспечивает то, что последующая строка является продолжением предыдущей. При этом надо помнить, что:

- нельзя разбивать переносом строковые константы;
- допустимо не более семи продолжений одной строки;
- строка не может состоять более чем из 1024 символов.

Например:

```
Dim a As Integer, b As Byte _
S As String
```

Использование знака двоеточия позволяет разместить несколько операторов в одной строке. Таким образом, следующие две конструкции эквивалентны:

```
x = x + 1
y = x + 2
```

```
x = x + 1:y = x + 2
```

Комментарии

Работая с программой, удобно использовать **комментарии** – фрагменты текста программы, не являющиеся программными кодами и игнорируемые транслятором. Комментарии выполняют две важные функции:

- делают программу легко читаемой, поясняя смысл программных кодов и алгоритма;
- временно отключают фрагменты программы при отладке.

В языке VBA существуют два способа ввода комментариев:

- применение апострофа `'`. Его можно ставить в любом месте строки. Все символы от апострофа до конца строки будут восприниматься компилятором как комментарий;

– применение зарезервированного слова *Rem*.

Пример использования комментариев в тексте программы:

```
Dim a As Integer   ' a — целая переменная
Dim b As String
Rem b — строковая переменная
```

2.3. Операции в VBA

Операция – способ обозначения выполняемого в составе инструкции действия над аргументами с получением результата.

Операция записывается по аналогии с записью алгебраического выражения. Аргументы операции называются **операндами**. По количеству операндов различают унарные (одноместные – с одним операндом), бинарные (двуместные – с двумя операндами) и так далее. В VBA присутствуют только одноместные и двуместные операции.

Действие обозначается специальным символом или ключевым словом. Для унарных операций в VBA используется префиксная нотация (символ действия перед операндом). Для бинарных операций используется инфиксная нотация (символ действия между операндами).

В одной инструкции может присутствовать несколько операций, которые выполняются последовательно в соответствии с правилами приоритета по аналогии с алгебраическими операциями.

Арифметические операции

Арифметические операции VBA сведены в таблицу 4.

Таблица.4. Арифметические операции.

Операция	Описание
`<Операнд1> + <Операнд2>`	Сложение
`<Операнд1> - <Операнд2>`	Вычитание
`- <Операнд>`	Перемена знака
`<Операнд1> * <Операнд2>`	Умножение
`<Операнд1> / <Операнд2>`	Деление
`<Операнд1> \ <Операнд2>`	Целочисленное деление
`<Операнд1> Mod <Операнд2>`	Остаток от деления по модулю
`<Операнд1> ^ <Операнд2>`	Возведение в степень

Операнды должны быть числовых подтипов, результаты операций также являются числами.

Строковые операции

В VBA присутствует одна встроенная строковая операция: конкатенация (слияние) строк.

Синтаксис

```
<Строка1> & <Строка2>
```

или

```
<Строка1> + <Строка2>
```

Результатом операции будет строка, образованная слиянием *Строки1* и *Строки2*.

Для сложения строк предпочтительнее, во избежание путаницы с арифметическим сложением, применять операцию со знаком `&`

Операции отношения

Операции отношения или операции сравнения сведены в таблицу 5.

Таблица 5. Операции отношения.

Операция	Описание
`<Операнд1> < <Операнд2>`	Меньше
`<Операнд1> > <Операнд2>`	Больше
`<Операнд1> <= <Операнд2>`	Меньше или равно
`<Операнд1> >= <Операнд2>`	Больше или равно
`<Операнд1> <> <Операнд2>`	Не равно
`<Операнд1> = <Операнд2>`	Равно
`<Операнд1> Is <Операнд2>`	Сравнение двух операндов, содержащих ссылки на объекты
`<Операнд1> Like <Операнд2>`	Сравнение двух строковых выражений

Оба операнда должны относиться к одному подтипу, результат операции будет булевского типа: *`True`*, если отношение выполняется, и *`False`*, если не выполняется.

Логические операции.

Операции данного типа перечислены в таблице 6.

Таблица 6. Логические операции.

Операция	Описание
<Операнд1> And <Операнд2>	Логическое умножение («И»)
<Операнд1> Or <Операнд2>	Логическое сложение («ИЛИ»)
<Операнд1> Xor <Операнд2>	Исключающее «ИЛИ»
Not <Операнд1>	Логическое отрицание («НЕ»)
<Операнд1> Imp <Операнд2>	Логическая импликация
<Операнд1> Equ <Операнд2>	Логическая эквивалентность

Операнды могут быть булевского или одного из числовых типов. Если операнды логических операций булевского типа, результат операции тоже будет булевского типа соответственно таблице истинности 7. Операция *Not* одноместная и применяется только к *Операнду1*, остальные операции – двуместные. В других случаях выполняется побитовая логическая операция над двоичными представлениями операндов.

Таблица 7. Таблица истинности

Операнд1	Операнд2	And	Or	Xor	Not	Imp	Equ
False	False	False	False	False	True	False	True
False	True	False	True	True	True	True	False
True	False	False	True	True	False	False	False
True	True	True	True	False	False	False	True

Чаще всего логические операции используются для объединения нескольких операций сравнения. Например, если имеются переменные *a, b,* и *c,* каждая из которых должна быть меньше двадцати, и в программе необходимо проверить этот факт, формулируется условие

```
(a<20) And (b<20) And (c<20)
```

которое выполняется (результат равен *True* – истина) только в случае, когда значение каждой переменной меньше двадцати. Если хотя бы одна переменная содержит значение большее или равное двадцати, условие не выполняется и результат сравнения будет равен *False*.

Приоритеты операций

VBA выполняет операции в соответствии с их приоритетами, что обеспечивает однозначность в трактовке значений выражений. Приоритеты выполнения операций приведены в таблице 8.

Таблица 8. Приоритеты операций.

Приоритет	Операция	Приоритет	Операция
1	Вызов функции и скобки	8	>, <, >=, <=, <>, =
2	^	9	Not
3	– (смена знака)	10	And
4	*, /	11	Or
5	\	12	Xor
6	Mod	13	Equ
7	+, -, &	14	Imp

Операции с одинаковым приоритетом выполняются слева направо.

2.4. Встроенные функции VBA

Функция определяет зависимость возвращаемого значения от одного или нескольких аргументов, называемых параметрами. Тип значения функции и типы аргументов совпадают с типами данных VBA и определены в описании функции.

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

- Математические функции
- Функции проверки типов
- Функции преобразования форматов
- Функции обработки строк
- Функции времени и даты

Ниже рассмотрены основные функции из этих категорий.

Математические функции

Функции данного вида перечислены в таблице 9.

Таблица 9. Встроенные математические функции VBA

Функция	Возвращаемое значение
`Abs(<число>)`	Модуль (абсолютная величина) числа
`Atn(<число>)`	Арктангенс
`Cos(<число>)`	Косинус угла, заданного в радианах.
`Exp(<число>)`	Экспонента, то есть результат возведения основания натурального логарифма в указанную степень
`Log(<число>)`	Натуральный логарифм
`Rnd(<число>)`	Случайное число из интервала [0,1). Если *число* меньше нуля, то *Rnd* возвращает каждый раз одно и то же число, используя аргумент в качестве опорного числа; если *число* больше нуля или аргумент опущен, то следующее случайное число в последовательности; если *число* равняется нулю, то случайное число, возвращенное при предыдущем вызове этой функции. Перед вызовом функции *Rnd* используйте инструкцию *Randomize* без аргумента
`Sgn(<число>)`	Знак числа. Возвращает -1, 0 или 1.
`Sin(<число>)`	Синус
`Sqr(<число>)`	Квадратный корень из числа
`Tan(<число>)`	Тангенс
`Fix(<число>)` `Int(<число>)`	Обе функции, *Int* и *Fix*, отбрасывают дробную часть числа и возвращают целое значение. Различие между функциями *Int* и *Fix* состоит в том, что для отрицательного значения аргумента функция *Int* возвращает ближайшее отрицательное целое число, меньшее либо равное указанному, а *Fix* — ближайшее отрицательное целое число, большее либо равное указанному

Функции проверки типов

Функции данного вида (Таблица 10) проверяют, является ли переменная выражением специфицированного типа.

Таблица 10. Функции проверки типов VBA

Функция	Специфицированный тип
`IsArray(<переменная>)`	Является ли переменная массивом
`IsDate(<переменная>)`	Является ли переменная датой
`IsEmpty(<переменная>)`	Была ли переменная описана инструкцией *Dim*
`IsError(<переменная>)`	Является ли переменная кодом ошибки
`IsMissing(<переменная>)`	Была ли передана переменная как необязательный параметр
`IsNull(<переменная>)`	Является ли переменная пустым значением (*Null*)
`IsNumeric(<переменная>)`	Является ли переменная числом
`IsObject(<переменная>)`	Является ли переменная объектом

Возвращаемое значение имеет булевский тип: *True*, если переменная имеет специфицированный тип, и *False*, если не является.

Функции преобразования форматов

Преобразование строки в число и обратно осуществляют следующими функциями (Таблица 11).

Таблица 11. Строковые функции преобразования форматов.

Функция	Возвращаемое значение
`Val(<строка>)`	Возвращает числа, содержащиеся в строке, как числовое значение соответствующего типа
`Str(<число>)`	Возвращает значение типа *variant (String)*, являющееся строковым представлением числа. В качестве десятичного разделителя функция *Str* использует точку. При наличии другого десятичного разделителя (например, запятой) для преобразования чисел в строки следует использовать функцию *cstr*, описанную ниже.

Кроме функций *val* и *str* в VBA имеются другие функции преобразования типов (Таблица 12).

Таблица 12. Функции преобразования типов.

Функция	Тип, в который преобразуется аргумент
CBool(<Выражение>)	Boolean
CByte(<Выражение>)	Byte
CCur(<Выражение>)	Currency
CDate(<Выражение>)	Date
CDbl(<Выражение>)	Double
CDec(<Выражение>)	Decimal
CInt(<Выражение>)	Integer
CLng(<Выражение>)	Long
CSng(<Выражение>)	Single
CVar(<Выражение>)	Variant
CStr(<Выражение>)	String

Чтобы представить числовое значение как дату, время, денежное значение или в специальном формате, следует использовать функцию *Format*.

Синтаксис:

```
Format(Выражение [, Формат [, ПервыйДеньНедели [,
ПерваяНеделяГода]]])
```

Возвращает значение типа *variant (string)*, содержащее выражение, отформатированное согласно инструкциям, заданным в описании формата.

Аргументы:

Выражение — любое допустимое выражение

Формат — любое допустимое именованное или определяемое пользователем выражение формата. Примером именованного формата является *Fixed* — формат действительного числа с двумя значащими цифрами после десятичной точки

ПервыйДеньНедели — постоянная, определяющая первый день недели

ПерваяНеделяГода — постоянная, определяющая первую неделю года

При построении пользовательского формата возможно использование специальных символов (Таблица 13).

Таблица 13. Символы для построения пользовательского формата.

Символ	Назначение
0	Резервирует позицию цифрового разряда. Отображает цифру или нуль. Если у числа, представленного аргументом, есть какая-нибудь цифра в той позиции разряда, где в строке формата находится *0*, функция отображает эту цифру аргумента, если нет — в этой позиции отображается нуль
#	Резервирует позицию цифрового разряда. Отображает цифру или ничего не отображает. Если у числа, представленного аргументом, есть какая-нибудь цифра в той позиции разряда, где в строке формата находится #, функция отображает эту цифру аргумента, если нет — в исходной позиции не отображается ничего. Действие данного символа аналогично действию *0*, за исключением того, что лидирующие нули не отображаются
.	Резервирует позицию десятичного разделителя. Указание точки в строке формата определяет, сколько разрядов необходимо отображать слева и справа от десятичной точки
%	Резервирует процентное отображение числа
,	Разделитель разряда сотен от тысяч
:	Разделитель часов, минут и секунд в категории форматов *Время (Time)*
/	Разделитель дня, месяца и года в категории форматов *Дата (Date)*
E+, E-, e+, e-	Разделитель мантиссы и порядка в экспоненциальном формате

Функции обработки строк

В VBA имеются следующие функции обработки строковых выражений (Таблица 14).

Таблица 14. Функции обработки строк

Функция	Результат	Синтаксис:
Asc	Возвращает ASCII-код начальной буквы строки.	Asc(<Строка>)
Chr	Преобразует ASCII-код в строку.	Chr(<Код>) Например, *Chr(13)* - переход на новую строку, *Chr(97)="a"*
Lease	Преобразует строку к нижнему регистру.	Lease(<Строка>)
Ucase	Преобразует строку к верхнему регистру.	Ucase(<Строка>)
Left	Возвращает подстроку, состоящую из заданного числа первых символов исходной строки.	Left(<string>, <length>) **Аргументы:** **length** – число символов **string** – исходная строка
Right	Возвращает строку, состоящую из заданного числа последних символов исходной строки.	Right(<string>, <length>) **Аргументы:** **length** – число символов **string** – исходная строка
Mid	Возвращает подстроку строки, содержащую указанное число символов.	Mid(<string>, <start> [, <length>]) **Аргументы:** **string** – строковое выражение, из которого извлекается подстрока **start** – позиция символа в строке *string*, с которого начинается нужная подстрока **length** – число возвращаемых символов подстроки.
Len	Возвращает число символов строки.	Len(<Строка>)

Таблица 14, продолжение

Функция	Результат	Синтаксис:
`LTrim`	Возвращает копию строки без пробелов в начале.	`LTrim(<Строка>)`
`RTrim`	Возвращает копию строки без пробелов в конце.	`RTrim(<Строка>)`
`Trim`	Возвращает копию строки без пробелов в начале и в конце	`Trim(<Строка>)`
`Space`	Возвращает строку, состоящую из указанного числа пробелов.	`Space(<Число>)`
`String`	Возвращает строку, состоящую из указанного числа повторений одного и того же символа.	`String(<number>, <character>)` **Аргументы:** **`number`** – число повторений **`character`** – повторяемый символ
`StrComp`	Возвращает результат сравнения двух строк.	`StrComp(<string1>, <string2> [, <compare>])` **Аргументы:** **`string1`** и **`string2`** – два любых строковых выражения **`compare`** – указывает способ сравнения строк. Допустимые значения: *0* (двоичное сравнение), *1* (посимвольное сравнение без учета регистра) Возвращаемые значения: Если **`string1 < string2`**, то **`-1`** Если **`string1 = string2`**, то **`0`** Если **`string1 > string2`**, то **`1`**

Таблица 14, окончание

Функция	Результат	Синтаксис:
`InStr`	Возвращает позицию первого вхождения одной строки внутри другой строки.	`InStr([start,] <string1>, <string2> [, <compare>])` **Аргументы:** **`start`** – числовое выражение, задающее позицию, с которой начинается каждый поиск. Если этот аргумент опущен, поиск начинается с первого символа строки **`string1`** – строковое выражение, в котором выполняется поиск **`string2`** – искомое строковое выражение **`compare`** – указывает способ сравнения строк. Допустимые значения: *0* (для двоичного сравнения), *1* (посимвольное сравнение без учета регистра)

Функции времени и даты

Функции обработки времени и даты перечислены в таблице 15.

Таблица 15. Функции времени и даты

Функция	Результат	Синтаксис
`Date`	Возвращает значение типа *`Variant (Date)`*, содержащее текущую системную дату	`Date()`
`Time`	Возвращает значение типа *`Variant (Date)`*, содержащее текущее время по системным часам компьютера	`Time()`
`Now`	Возвращает значение типа *`Variant (Date)`*, содержащее текущую дату и время по системному календарю и часам компьютера	`Now()`
`Timer`	Возвращает значение типа *`Single`*, представляющее число секунд, прошедших после полуночи	`Timer()`

Таблица 15, продолжение

Функция	Результат	Синтаксис
Hour Minute Second	Возвращают значения типа *variant (integer)*, содержащее целое число, которое представляет часы, минуты и секунды в значении времени	Hour(<время>) Minute(<время>) Second(<время>) **Аргументы:** **время** – значение времени или выражение, его определяющее В следующем примере переменной *Час* присваивается *16*, *Минута – 35*, *Секунда – 17:* Время = #4:35:17 PM# Час = Hour (Время) Минута = Minute (Время) Секунда = Second (Время)

Day
Month
Year
	Возвращает значение типа *Variant (Integer)*, содержащее целое число, которое представляет день, месяц, год в значении даты	Day(<дата>)
		Month(<дата>)
		Year(<дата>)
		Аргументы:
		дата – значение даты или выражение, ее определяющее
		В следующем примере переменной *день* присваивается *17*, *Месяц – May*, *Год – 1960:*
		ДеньРож = #May 17, 1960#
		День = Day (ДеньРож)
		Месяц = Month (ДеньРож)
		Год = Year (ДеньРож)

Таблица 15, продолжение

Функция	Результат	Синтаксис

`TimeValue`	Возвращает значение типа *Variant (Date)*, соответствующее времени, указанному в строке.	`TimeValue(<Строка>)`
		Аргументы:
		строка – значение типа *string*, обозначающее время.
		В данном примере переменной *Время* присваивается *#4:35:17 PM#:*
		`Время = TimeValue("4:35:17 PM")`

`TimeSerial`	Возвращает значение типа *Variant (Date)*, содержащее значение времени, соответствующее указанным часу, минуте и секунде.	`TimeSerial(<hour>, <minute>, <second>)`
		Аргументы:
		hour, **minute** и **second** – значения типа *Variant (Integer)*
		В данном примере переменной *Время* присваивается *#4:35:17 PM#:*
		`Время = TimeSerial(16, 35, 17)`

`DateSerial`	Возвращает значение типа *Variant (Date)*, соответствующее указанному году, месяцу и дню.	`DateSerial(<year>, <month>, <day>)`
		Аргументы:
		year, **month** и **day** – значения типа *Integer*
		В следующем примере переменной *Дата* присваивается *#May 17, 1960#*:
		`Дата = DateSerial(1960, 5, 17)`

Таблица 15, продолжение

Функция	Результат	Синтаксис
DateDiff	Возвращает значение типа *Variant (Long)*, указывающее число временных интервалов между двумя датами	DateDiff(<interval>, <date1>, <date2> [, <firstdayofweek> [, <firstweekofyear>]]) **Аргументы:** **Interval** – строковое выражение, указывающее тип временного интервала, который используется при вычислении разности между датами *date1* и *date2*. Допустимые значения: *"yyyy"* (год), *"q"* (квартал), *"m"* (месяц), *"y"* (день года), *"d"* (день месяца), *"w"* (день недели), *"ww"* (неделя), *"h"* (часы), *"m"* (минуты), *"s"* (секунды) **date1, date2** – значения типа *Variant (Date)*. Две даты, разность между которыми следует вычислить **firstdayofweek** – число, обозначающее первый день недели. По умолчанию подразумевается *1* (воскресенье). Допустимы значения: *0* (указанное в настройках системы), *2* (понедельник), *3* (вторник), *4* (среда), *5* (четверг), *6* (пятница) и *7* (суббота) **firstweekofyear** – число, обозначающее первую неделю года. По умолчанию подразумевается *1* (неделя, включающее первое января). Допустимы также значения: *0* (указанное в настройках системы), *2* (неделя, включающая как минимум 4 дня нового года), *3* (первая полная неделя года). В следующем примере переменной *ПМ* присваивается *465:* ПМ = DateDiff("m",#5/17/601,Now)

Таблица 15, продолжение

Функция	Результат	Синтаксис

DatePart	Возвращает значение типа *variant (Integer)*, содержащее указанный компонент даты.	DatePart(<interval>, <date> [, <firstdayofweek> [, <firstweekofyear>]])
		Аргументы:
		Interval – строковое выражение, указывающее тип временного интервала
		date – значение типа Variant (Date). Дата, для которой нужно определить интервал.
		firstdayofweek – число, обозначающее первый день недели.
		firstweekofyear – число, обозначающее первую неделю года.
		В примере переменной *ДеньГода* присваивается *138*:
		ДеньРож = #May 17, 1960#
		ДеньГода = DatePart("y", ДеньРож)

Weekday	Возвращает значение типа *Variant (integer)*, содержащее целое число, представляющее день недели.	Weekday(<date> [,<firstdayofweek>])
		Аргументы:
		date – значение типа Variant (Date). Дата, для которой нужно определить день недели.
		firstdayofweek – число, обозначающее первый день недели.
		В примере переменной *ДеньНед* присваивается *3*, то есть вторник:
		ДеньРож = #May 17, 1960#
		ДеньНед = Weekday (ДеньРож)

Таблица 15, окончание

Функция	Результат	Синтаксис
DateAdd	Возвращает значение типа *Variant (Date)*, содержащее дату, к которой добавлен указанный временной интервал.	DateAdd (<interval>, <number>, <date>) **interval** – строковое выражение, указывающее тип добавляемого временного интервала **number** – числовое выражение, указывающее число временных интервалов, которое следует добавить. Может быть положительным (для получения более поздних дат) или отрицательным (для более ранних). **date** – значение типа *Variant (Date)*, представляющее дату, к которой добавляется временной интервал В следующем примере переменной *Д* присваивается значение *#March 17, 1963#:* Д = DateAdd("m", 34, 05/17/60#)

Функции, возвращающие строки

Некоторые функции имеют по две версии, одна из которых возвращает тип данных *Variant*, а другая – тип данных *String*. Первая версия является более удобной, так как при этом для значений типа *Variant* преобразование типов данных выполняется автоматически. Вторая версия, возвращающая тип *String*, использует меньше памяти и может быть полезна в следующих случаях:

– Для экономии памяти, если в программе имеется очень много переменных;
– При выполнении прямой записи данных в файлы с произвольным доступом.

Перечисленные ниже функции (Таблица 16) возвращают значения типа *String*, если к их имени добавляется символ доллара (*$*).

Эти функции имеют такое же применение и синтаксис, как и их эквиваленты без символа доллара, возвращающие тип *Variant*.

Таблица 16. Функции, возвращающие строки

Chr$	CurDir$	Date$	Dir$	Error$	Format$	Input$
InputB$	LCase$	Left$	LTrim$	MidS	Right$	Rtrim$
Space$	Str$	String$	Time $	Trim$	Ucase$	

Встроенные диалоговые окна

В проектах VBA часто встречаются две разновидности диалоговых окон: окна сообщений и окна ввода. Они встроены в VBA, и если их возможностей достаточно, то можно обойтись без проектирования диалоговых окон. Окно сообщений (*MsgBox*) выводит простейшие сообщения для пользователя, а окно ввода (*InputBox*) обеспечивает ввод информации.

Синтаксис

```
InputBox(prompt[, title][, default][, xpos][,
ypos][, helpfile, context])
```

Возвращает введённую строку (тип *Variant (String)*).

Аргументы:

prompt – строковое выражение, отображаемое как сообщение в диалоговом окне. Строковое значение *prompt* может содержать несколько строк. Для разделения строк допускается использование символа возврата каретки (*Chr(13)*), символа перевода строки (*Chr (10)*) или комбинацию этих символов (*Chr(13) & Chr(10)*)

title – строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку заголовка помещается имя приложения

default – строковое выражение, отображаемое в поле ввода как используемое по умолчанию, если пользователь не введет другую строку. Если этот аргумент опущен, поле ввода изображается пустым

xpos – числовое выражение, задающее расстояние по горизонтали между левой границей диалогового окна и левым краем экрана.

Если этот аргумент опущен, диалоговое окно выравнивается по центру экрана по горизонтали

ypos – числовое выражение, задающее расстояние по вертикали между верхней границей диалогового окна и верхним краем экрана. Если этот аргумент опущен, диалоговое окно помещается по вертикали примерно на одну треть высоты экрана

helpfile – строковое выражение, определяющее имя файла справки, содержащего справочные сведения о данном диалоговом окне. Если этот аргумент указан, необходимо наличие также аргумента *context*

context – числовое выражение, определяющее номер соответствующего раздела справочной системы. Если этот аргумент указан, необходимо наличие также аргумента *helpfile*

Синтаксис

```
MsgBox(prompt[, buttons][, title][, helpfile,
context])
```

Возвращает код нажатой кнопки (Тип *Integer*).

Аргументы:

prompt – строковое выражение, отображаемое как сообщение в диалоговом окне

buttons – числовое выражение, представляющее сумму значений, которые указывают число и тип отображаемых кнопок, тип используемого значка и основную кнопку окна сообщения. Значение по умолчанию этого аргумента равняется *0*.

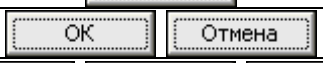
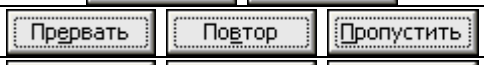
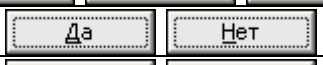
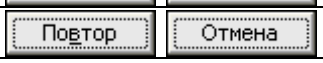
title – строковое выражение, отображаемое в строке, заголовка диалогового окна. Если этот аргумент опущен, в строку заголовка помещается имя приложения

helpfile – строковое выражение, определяющее имя файла справки, содержащего справочные сведения о данном диалоговом окне. Если этот аргумент указан, необходимо указать также аргумент *context.*

context – числовое выражение, определяющее номер соответствующего раздела справочной системы. Если этот аргумент указан, необходимо указать также аргумент *helpfile.*

Значения аргумента *buttons* процедуры *MsgBox*, определяющие отображаемые кнопки в диалоговом окне, представлены в таблице (Таблица 17)

Таблица 17. Возможные значения аргумента *Buttons* для кнопок.

Константа	Значение	Отображаются кнопки
vbOKOnly	0	
vbOKCancel	1	
vbAbortRetrylgnore	2	
vbYesNoCancel	3	
vbYesNo	4	
vbRetryCancel	5	

Значения аргумента *buttons* процедуры *MsgBox*, определяющие отображаемые информационные значки в диалоговом окне представлены в таблице (Таблица 18)

Таблица 18. Возможные значения аргумента *Buttons* для значков

Константа	Значение	Значок
vbCritical	16	
vbQuestion	32	
vbExclamation	48	
vbInformation	64	

Значения аргумента *buttons* процедуры *MsgBox*, определяющие основную кнопку в диалоговом окне представлены в таблице (Таблица **19**)

Таблица 19. Значения аргумента *Buttons* для кнопки по умолчанию

Константа	Значение	Кнопка
VbDefaultButton1	0	1
VbDefaultButton2	256	2
VbDefaultButton3	512	3
VbDefaultButton4	768	4

При написании программ с откликом, в зависимости от того, какая кнопка диалогового окна нажата, вместо возвращаемых значений удобнее использовать константы VBA, которые делают код про-

граммы удобочитаемым и, к тому же, их легко запомнить. Эти константы перечислены в таблице (Таблица 20)

Таблица 20. Константы - возвращаемые значения *MsgBox*.

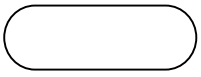
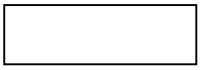
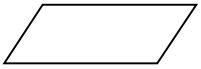
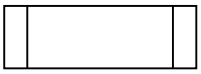
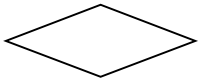
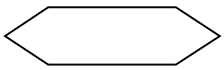
Константа.	Значение	Нажатая кнопка
vbOK	1	OK
vbCancel	2	Отмена (Cancel)
vbAbort	3	Прервать (Abort)
vbRetry	4	Повторить (Retry)
vbIgnore	5	Пропустить (Ignore)
vbYes	6	Да (Yes)
vbNo	7	Нет (No)

2.5. Схема алгоритма

Схема алгоритма – точное наглядное графическое изображение последовательности действий.

Каждый блок схемы алгоритма имеет взаимно однозначно соответствующую конструкцию алгоритмического языка (табл.).

Таблица. Базовые блоки схемы алгоритма:

Вид	Значение
	Терминатор, обозначает начало и окончание алгоритма.
	Процесс – выполнение какого-либо действия.
	Ввод или вывод данных.
	Ссылка на текущей странице.
	Ссылка на другую страницу.
	Типовой процесс (подпрограмма).
	Условие.
	Подготовка цикла.

Блоки 7 и 8 при организации алгоритмов линейной структуры не используются.

Лабораторная работа №2

Цель работы: Научиться создавать простейшие программы на VBA.

Задание

1. Разработать алгоритм и программу на VBA по варианту.
2. Ввести программу в редактор VBA и добиться её выполнения.
3. Ответить на контрольные вопросы.
4. Подготовить отчёт.

Содержание отчёта

1. Титульный лист.
2. Индивидуальное задание.
3. Схема алгоритма.
4. Текст программы с комментариями.
5. Ответы на контрольные вопросы.

Индивидуальные задания

1. Ввести два числа и вывести их сумму, разность, произведение и частное.
2. Ввести числа x и y. Вывести x в степени y и корень степени y из x.
3. Ввести два числа. Вывести результат их целочисленного деления и остаток от деления.
4. Вывести пять случайных чисел и их сумму.
5. Вывести случайное число в заданном диапазоне.
6. Ввести строку и вывести заданное число её последних символов, дополнив её пробелами слева до длины введённой строки.
7. Ввести последовательно ФИО и вывести инициалы.
8. Ввести строку. Вывести позицию заданной буквы и окончание строки, начиная с этой буквы.
9. Ввести последовательно три слова и вывести их попарно.
10. По заданной дате рождения определить возраст.
11. Ввести длины сторон треугольника и вывести его периметр, площадь и их отношение.

12. Ввести длины сторон треугольника и вывести радиус описанной окружности.
13. Ввести длины сторон треугольника и вывести радиус вписанной окружности.
14. Ввести длины катетов и вывести длину гипотенузы и углы прямоугольного треугольника.
15. Ввести длины сторон прямоугольника и вывести его периметр, площадь и их отношение.
16. Ввести радиус круга. Найти площадь круга, длину окружности и их отношение.
17. Ввести сторону правильного треугольника и вывести периметр его и эквивалентного по площади квадрата.
18. Ввести длину стороны квадрата. Вывести периметр квадрата, радиус эквивалентного по площади круга и длину окружности.
19. Ввести радиус сферы и вычислить её объём, площадь поверхности и их отношение.
20. Ввести радиус и высоту цилиндра, вычислить его объём, площадь поверхности и их отношение.
21. Ввести радиус и высоту цилиндра и вывести длину ребра и площадь поверхности куба эквивалентного объёма.
22. Ввести радиус и высоту цилиндра и вывести радиус шара эквивалентного объёма.
23. Ввести трёхзначное число и вывести его цифры по одной.
24. Ввести последовательно четыре цифры и преобразовать их в соответствующее целое четырёхзначное число.
25. Ввести вещественную и мнимую часть комплексного числа. Вывести число в тригонометрической форме.
26. Ввести модуль и аргумент комплексного числа и вывести его вещественную и мнимую части.
27. Два комплексных числа задать вещественными и мнимыми частями. Найти сумму и произведение этих чисел.
28. Два комплексных числа задать вещественными и мнимыми частями. Найти разность и частное этих чисел.
29. Комплексное число задать вещественной и мнимой частями. Возвести это число в степень n.
30. Комплексное число задать вещественной и мнимой частями. Вычислить его квадратный корень.

Контрольные вопросы

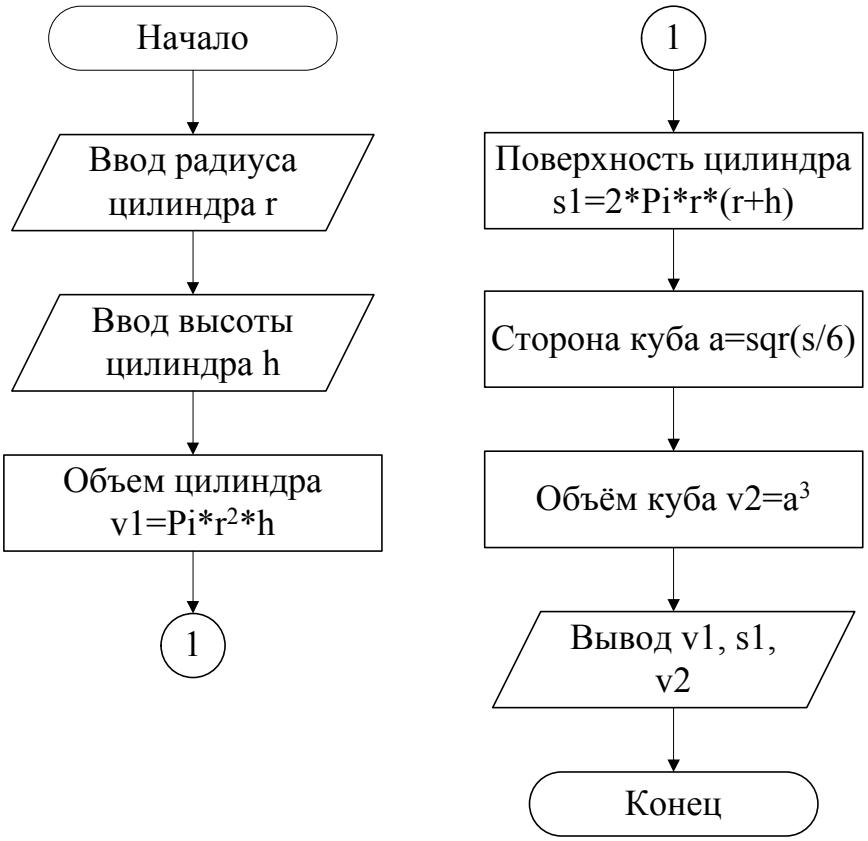
1. Что такое переменные и зачем они применяются?
2. Что такое тип данных?
3. Почему все переменные лучше заранее описывать?
4. Зачем нужны константы?
5. Приведите тринадцать примеров недопустимых имён (по одному на каждое ограничение).
6. Какие проблемы могут возникнуть, если переменной одного типа присвоить значение другого типа?
7. Зачем нужны комментарии?
8. Что такое операция?
9. По какому признаку операции в VBA делятся на типы?
10. Какие проблемы возникли бы при отсутствии приоритетов операций?
11. Что такое функция и зачем она нужна?
12. Как определить тип аргументов и тип возвращаемого значения функции?
13. Зачем нужны функции, возвращающие строки?
14. Каким образом можно ввести данные в программу в процессе её выполнения?
15. Какие способы вывода данных, используются в VBA?

Пример выполнения работы

Задание

По высоте и радиусу цилиндра вычислить объём цилиндра, площадь поверхности цилиндра и объём куба, площадь поверхности которого равна площади поверхности цилиндра.

Схема алгоритма



Текст программы

```
Sub Example1() 'Начало программы
  Dim r As Single 'Здесь будет радиус цилиндра
  Dim h As Single 'Здесь будет высота цилиндра
  Dim v1 As Single 'Здесь будет объём цилиндра
  Dim s1 As Single 'Здесь будет площадь
  Dim a As Single 'Здесь будет сторона куба
  Dim v2 As Single 'Здесь будет объём куба
  Const Pi = 3.1415926 'Здесь будет число Пи
    'Ввод радиуса и высоты цилиндра
  r = InputBox("Введите радиус цилиндра")
  h = InputBox("Введите высоту цилиндра")
  v1 = Pi*r^2*h 'Вычисление объёма цилиндра
  s1 = 2*Pi*r*(r+h) 'Расчёт площади поверхности
  a = Sqr(s1/6) 'Вычисление стороны куба
  v2 = a^3 'Вычисление объёма куба
  MsgBox ("Объём цилиндра = "+Str(v1)+Chr(13)+ _
```

```
    "Площадь поверхности цилиндра = "+Str(s1)+ _
    Chr(13)+"Объём куба с такой поверхностью = " _
    +Str(v2)) 'Вывод в 3 строки
End Sub
```

3. Реализация базовой алгоритмической структуры «ветвление»

Решение значительной части задач требует возможности изменения порядка выполнения инструкций в зависимости от исходных данных. Для обеспечения этой возможности используется базовая алгоритмическая структура «ветвление», позволяющая проверить условие относительно некоторого набора данных и, в зависимости от результатов проверки, назначить выполнение тех или иных инструкций. На языке VBA «ветвление» реализовано с помощью трёх операторов ветвления и трёх функций ветвления.

3.1. Операторы ветвления

Операторы ветвления представлены классическим условным оператором *If Then Else*, оператором последовательной проверки *If Then ElseIf* и оператором выбора *Select Case*.

Оператор If Then Else

Условный оператор *If Then Else* служит для организации базовой алгоритмической структуры «ветвление» (Рис.21).

Синтаксис

```
If <Условие> Then <ИнструкцияДа> [Else
<ИнструкцияНет>]
```

Если условие принимает значение *True*, выполняется *ИнструкцияДа*, идущая после слова *Then*. Если условие не выполняется (принимает значение *False*), выполняется *ИнструкцияНет*, идущая после слова *Else*. Схема алгоритма изображена на Рис.21.

Часть оператора, начинающаяся с *Else*, не является обязательной. Если она отсутствует, говорят, что оператор записан в сокращённой форме, иначе речь идёт о полной форме. Схема алгоритма, соответствующая сокращённой форме условного оператора изображена на Рис.22.

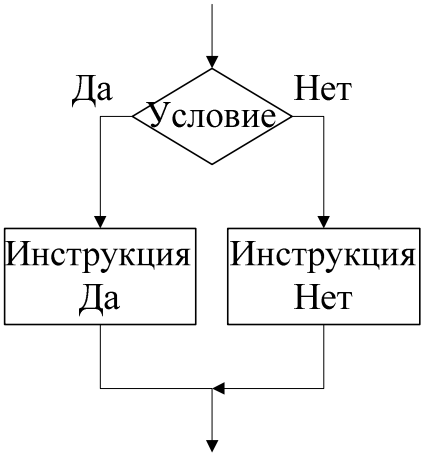


Рис.21. Базовая алгоритмическая структура «Ветвле-

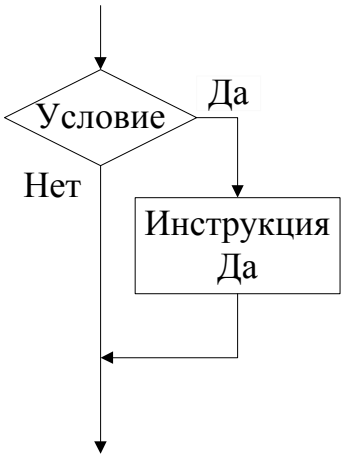


Рис.22. Схема алгоритма сокращённой формы If Then Else

Приведенная конструкция позволяет выполнить только по одной инструкции в случае выполнения или невыполнения условия. Если одна из ветвей должна содержать несколько инструкций, целесообразно применение блока `If`.

В следующем примере выполняется вычисление натурального логарифма. Как известно, логарифм можно вычислить только для положительного значения аргумента. Попытка вычислить логарифм неположительного числа, приведёт к аварийному завершению программы. Схема алгоритма представлена на Рис.23

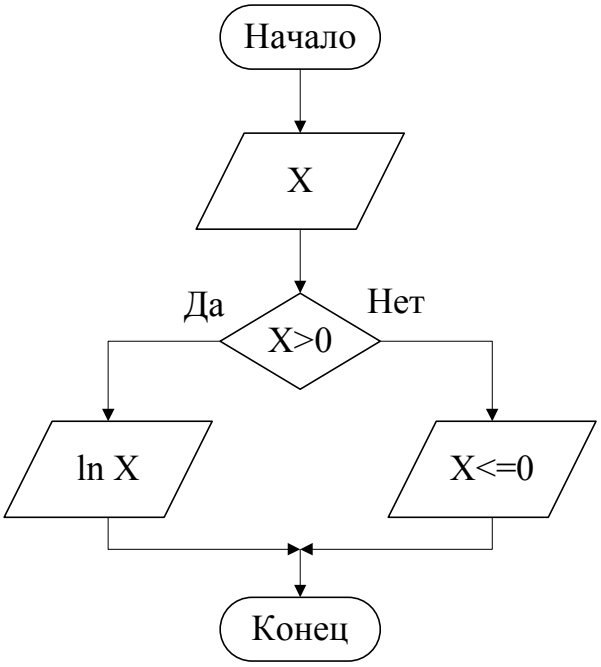


Рис.23. Алгоритм вычисления натурального логарифма

Текст программы

```
Sub ln()
Dim x As Single
x = Val(InputBox("Введите x"))
If x>0 Then _
  MsgBox("Ln "&Str(x)&" = "&Str(Log(x))) _
Else MsgBox ("Аргумент должен быть больше нуля")
End Sub
```

Блок If

Блок *If* выполняет функции условного оператора, но позволяет реализовать по несколько операторов в каждой ветви алгоритма.

Синтаксис

```
If <Условие> Then
<Инструкции>
[Else
<Инструкции>]
End If
```

Как и любой блок в VBA, блок *If* заканчивается ключевым словом *End* и названием блока. Ветвь *Else* не обязательна.

Пример

Вычислить площадь треугольника, заданного длинами сторон.

Как известно, чтобы вычислить площадь треугольника по длине сторон, необходимо убедиться, что треугольник существует. Иначе при вычислении площади по формуле Герона квадратный корень будет извлекаться из отрицательного числа, что приведёт к аварийному завершению программы.

Схема алгоритма приведена на (Рис.24)

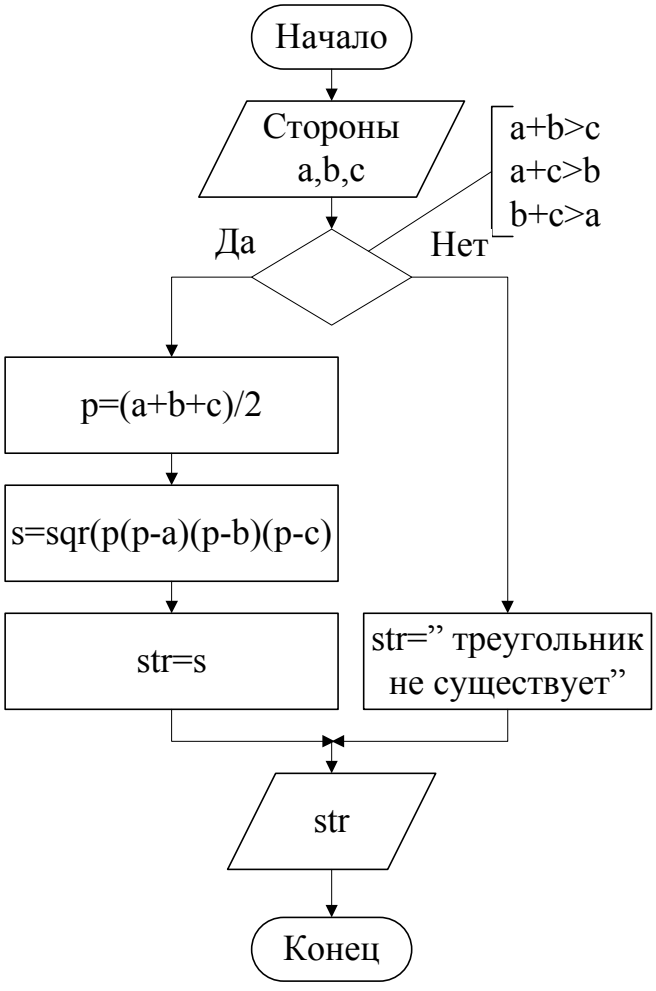


Рис.24. Алгоритм вычисления площади треугольника

Текст программы

```
Sub Triangle()
Dim a, b, c, p, s As Single, outstr As String
```

```
a = Val(InputBox("Введите длину стороны 1"))
b = Val(InputBox("Введите длину стороны 2"))
c = Val(InputBox("Введите длину стороны 3"))
If ((a+b)>c)And((a+c)>b)And((b+c)>a) Then
  p = (a+b+c)/2
  s = Sqr(p*(p-a)*(p-b)*(p-c))
  outstr = "Площадь треугольника равна " & Str(s)
Else
  outstr = "Треугольник не существует"
End If
MsgBox (outstr)
End Sub
```

Оператор If Then ElseIf

Рассмотрим задачу следующего содержания: по введённым длинам сторон определить, является ли треугольник равнобедренным, равносторонним, ни тем, ни другим, или вообще не существует.

Алгоритм задачи приведен на рис. 25

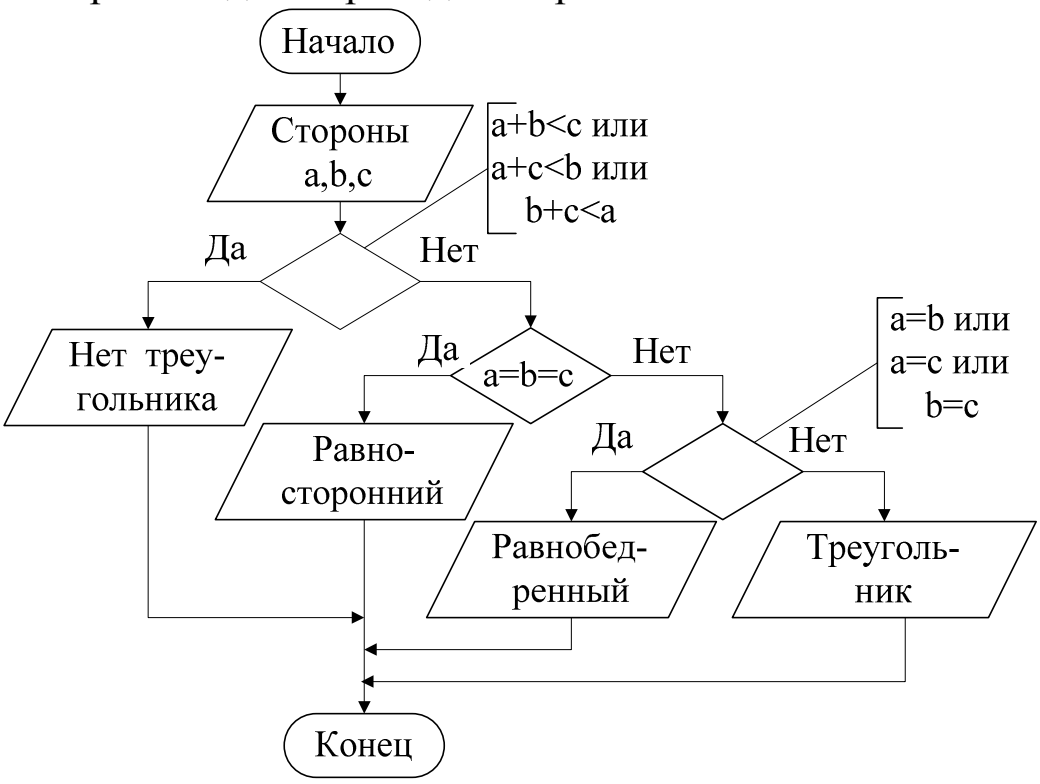


Рис.25. Алгоритм определения вида треугольника.

Для решения этой задачи требуется последовательная проверка трёх условий: в случае невыполнения первого проверяется второе, в случае невыполнения второго проверяется третье. В схему этого алгоритма идеально укладывается блок операторов *If Then ElseIf*.

Синтаксис

```
If <Условие 1> Then
<Инструкции 1>
[ElseIf <Условие 2> Then
<Инструкции 2>
...
[ElseIf <Условие n> Then
<Инструкции n>
[Else
<Инструкции>]
End If
```

Схема алгоритма

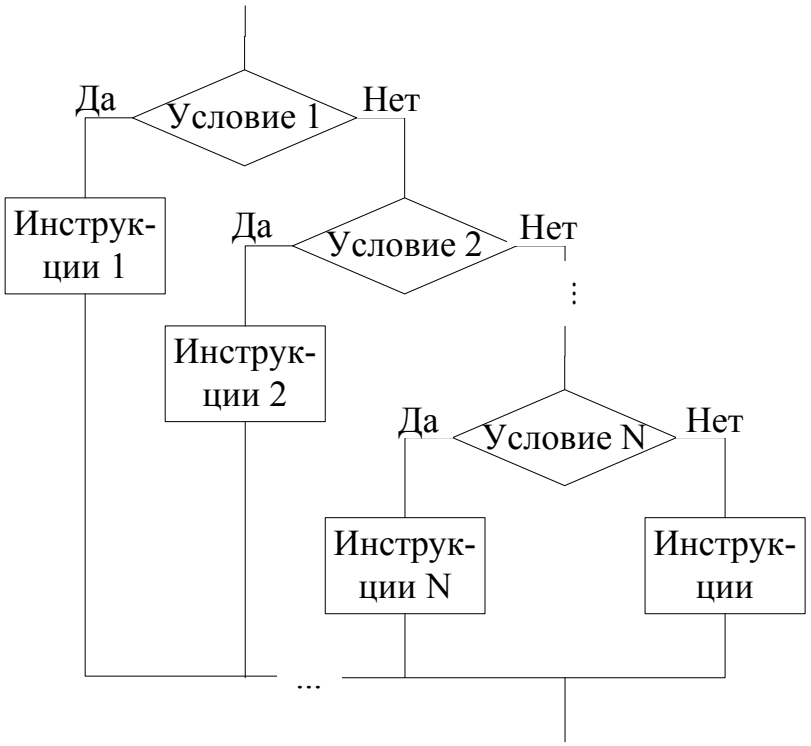


Рис.26. Схема алгоритма оператора *If Then ElseIf*.

Текст программы

```
Sub Triangles()
Dim a, b, c As Single
a = Val(InputBox("Введите сторону 1"))
b = Val(InputBox("Введите сторону 2"))
c = Val(InputBox("Введите сторону 3"))
If ((a+b)<=c)Or((a+c)<=b)Or((b+c)<=a) Then
  MsgBox ("Треугольника не существует")
ElseIf ((a=b)And(b=c)) Then
  MsgBox ("Треугольник равносторонний")
ElseIf ((a=b)Or(b=c)Or(a=c)) Then
  MsgBox ("Треугольник равнобедренный")
Else
  MsgBox ("Треугольник обыкновенный")
End If
End Sub
```

Оператор выбора Select Case

В случае, когда проверяемое выражение может удовлетворять одному из нескольких условий, и в зависимости от того, какому условию оно удовлетворяет, должна выполняться соответствующая ветвь алгоритма, применяется оператор выбора *Select Case*.

Синтаксис

```
Select Case <выражение>
Case <СписокВыражений-1>:
<инструкции- 1>
...
[Case <СписокВыражений-n>:
<инструкции-n>]
[Case Else: <инструкции else>]
End Select
```

Аргументы

Выражение – проверяемое выражение, в зависимости от значения которого выполняется та или иная ветвь алгоритма

СписокВыражений – одно или несколько (через запятую) условий типа список, диапазон (*50 To 100*) или отношение (*IS>10*), которым должно удовлетворять выражение.

инструкции – одна или несколько инструкций, выполняемых в том случае, если выражение удовлетворяет любому компоненту условия *СписокВыражений*

инструкции else (необязательная часть) – одна или несколько инструкций, выполняемых в том случае, если выражение не совпадает ни с одним из предложений *Case*

Схема алгоритма

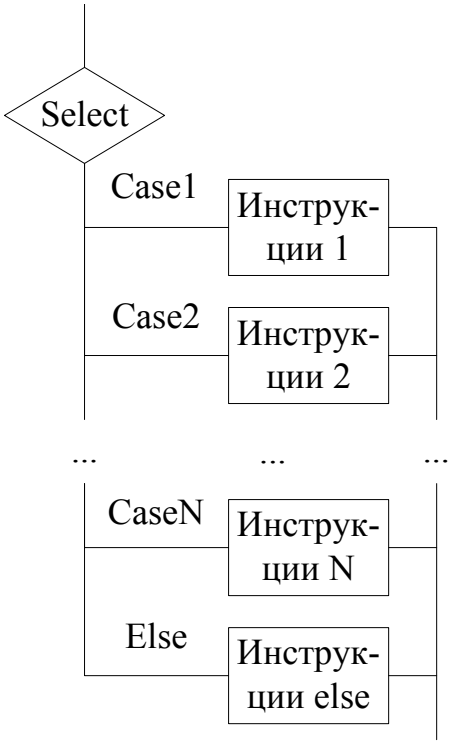


Рис.27. Схема алгоритма оператора *Select Case*

Пример

В зависимости от введённого времени выдать одно из сообщений: Утро, День, Вечер, Ночь, Полночь, Полдень.

Схема алгоритма программы приведена на рисунке 28

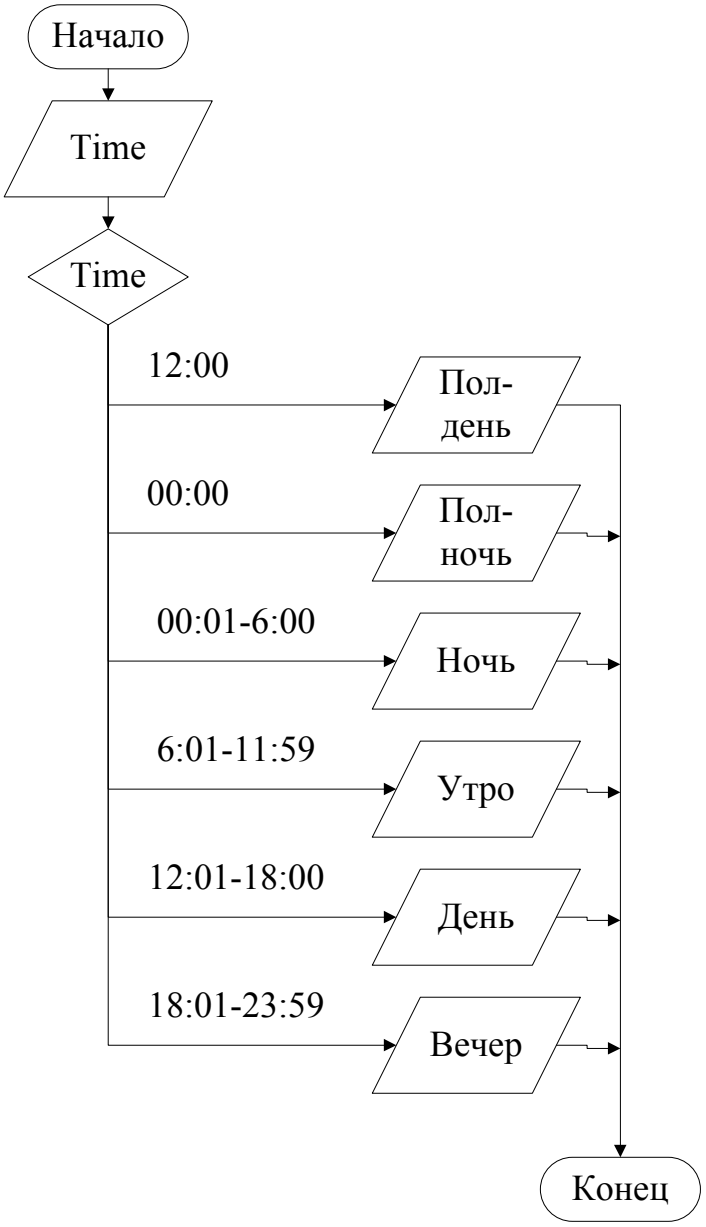


Рис.28. Схема алгоритма программы определения времени суток

Текст программы

```
Sub Time()
Dim T As Date
T = TimeValue(InputBox("Введите время чч:мм"))
Select Case T
Case #12:00:00 PM#:
  MsgBox ("Полдень")
Case #12:00:00 AM#:
```

```
  MsgBox ("Полночь")
Case #12:00:01 AM# To #6:00:00 AM#:
  MsgBox ("Ночь")
Case #6:00:01 AM# To #11:59:59 AM#:
  MsgBox ("Утро")
Case #12:00:01 PM# To #6:00:00 PM#:
  MsgBox ("День")
Case Else: MsgBox ("Вечер")
End Select
End Sub
```

Оператор безусловного перехода GoTo

Оператор безусловного перехода позволяет изменить порядок выполнения инструкций путём передачи управления определённой произвольно расположенной строке. Часто применяется совместно с условным оператором, так как для изменения порядка выполнения инструкций нужна причина, которую можно выявить проверкой условия. Применение оператора безусловного перехода не рекомендуется в связи с тем, что при его использовании алгоритм программы может стать неструктурированным. Вследствие этого программа будет работать неэффективно (оператор безусловного перехода выполняется очень медленно). Кроме того, затрудняется её чтение и поиск ошибок: при использовании нескольких меток понять алгоритм работы программы заметно сложнее, а искать ошибки проще в структурированных алгоритмах, так как в них алгоритм можно разбить на блоки, представляющие собой базовые алгоритмические структуры.

Чаще всего этот оператор используется, когда программа на VBA создаётся автоматически. Кроме того, в редких частных оправданных случаях применение оператора безусловного перехода может существенно упростить написание программы.

Синтаксис

```
<метка>: <Инструкция>
GoTo <метка>
```

Задает безусловный переход на указанную строку с *инструкцией* внутри процедуры. Обязательный аргумент *метка* может быть

любой меткой строки, соответствующей правилам именования VBA, или номером строки.

3.2. Функции выбора

Функции выбора позволяют реализовать базовую алгоритмическую структуру «ветвление» в виде функциональной зависимости. В ряде случаев применение функций выбора вместо соответствующих операторов приводит к упрощению записи и облегчению понимания программы за счёт компактности и возможности немедленно использовать возвращаемое значение.

Функция IIf

Функция *IIf* возвращает одну из двух альтернатив.

Синтаксис

```
IIf(<expr>, <truepart>, <falsepart>)
```

Аргументы

expr – проверяемое выражение

truepart — значение или выражение, возвращаемое, если *expr* имеет значение *True*

falsepart — значение или выражение, возвращаемое, если *expr* имеет значение *False*

Пример

Если значение переменной *Балл* равно *5*, то переменной *Оценка* присваивается строковая константа *"Отлично"*. В противном случае ей присваивается значение *"Не отлично"*:

```
Оценка = IIf(Балл = 5, "Отлично", "Не отлично")
```

Функция Choose

Функция *Choose* возвращает значение, выбранное из списка аргументов.

Синтаксис

```
Choose (индекс, вариант-1[, вариант-2, ... [,
вариант-n] ])
```

Аргументы

индекс — числовое выражение или поле, значением которого является число, лежащее между 1 и числом элементов в списке

вариант — выражение типа *Variant,* содержащее один из элементов списка

Действие функции *Choose:* если *индекс* равняется *1* , возвращается первый элемент списка, если *индекс* равняется *2*, возвращается второй элемент списка и т. д. Функцию *choose* можно использовать для выбора одного из возможных значений, представленных в виде списка.

Пример

Если аргумент *Выбор* принимает значения 1, 2 или 3, то переменной *РезультатВыбор* присваивается значение *"один", "два"* или *"три"* соответственно.

```
РезультатВыбор = Choose(Выбор, "один", "два",
"три")
```

Функция Switch

Функция *Switch* возвращает значение, соответствующее первому истинному выражению в списке.

Синтаксис

```
Switch ( выражение- 1, значение-1, выражение-2,
значение-2 ... [, выражение-n, значение-n] ] )
```

Пример

Переменной *OutStr* в зависимости от содержимого переменной *S* присваивается словесное описание оценки.

```
OutStr = Switch(S = 0, "Не аттестован", S = 1, _
  "Плохо", S = 2, "Неудовлетворительно", S = 3, _
```

```
"Удовлетворительно", S = 4, "Хорошо", S = 5, _
"Отлично", ((S>5)Or(S<0)), "Вообще не оценка")
```

Лабораторная работа №3

Цель работы: Получить навыки разработки программ с использованием разветвляющихся структур.

Задание

1. Разработать алгоритм и программу на VBA по варианту.
2. Ввести программу в редактор VBA и добиться её выполнения.
3. Ответить на контрольные вопросы.
4. Подготовить отчёт.

Содержание отчёта

1. Титульный лист.
2. Индивидуальное задание.
3. Схема алгоритма.
4. Текст программы с комментариями.
5. Ответы на контрольные вопросы.

Индивидуальные задания

1. По заданным коэффициентам a, b, c найти действительные корни уравнения $ax^4+bx^2+c=0$ или выдать сообщение об отсутствии действительных корней.
2. По введенным числам a_1,b_1,a_2,b_2 найти решение системы уравнений $\begin{cases} a_1x+b_1y=1 \\ a_2x+b_2y=1 \end{cases}$, если оно единственно.
3. По введенным числам a_1,b_1,a_2,b_2 найти решение системы неравенств $\begin{cases} a_1x+b_1y<1 \\ a_2x+b_2y<1 \end{cases}$.
4. Определить, является ли введённое целое число палиндромом.
5. По введённому номеру трамвайного билета определить, является ли он счастливым.
6. По заданным координатам полей шахматной доски определить, может ли конь перейти с одного поля на другое.
7. Определить, является ли введённое целое число полным квадратом, полным кубом, четвёртой или пятой степенью.

8. По заданным длинам сторон треугольника определить, является ли треугольник остроугольным, прямоугольным или тупоугольным.
9. Задан угол и прилежащие стороны параллелепипеда. Определить, является ли он ромбом, квадратом, прямоугольником или обыкновенным параллелепипедом.
10. Задана точка своими координатами на плоскости. Определить, к какой координатной четверти, или на какой оси расположена точка.
11. Окружность задана координатами центра и радиусом. По введённым координатам точки определить, попадает ли она в окружность, лежит на окружности или лежит вне окружности.
12. Две прямые описываются уравнениями $y=a_1x+b_1$, $y=a_2x+b_2$. Определить координаты точки пересечения прямых, либо сообщить, что эти прямые совпадают, не пересекаются или не существуют.
13. Окружность задана координатами центра и радиусом. Прямая описывается уравнением $y=ax+b$. Определить количество точек пересечения прямой и окружности.
14. Заданы стороны квадрата, правильного треугольника и радиус круга. Вывести названия тел в порядке возрастания площадей, периметров и их отношения.
15. Заданы радиус сферы, радиус и высота цилиндра и сторона куба. Вывести названия тел в порядке возрастания объёмов, площадей поверхностей и их отношения.
16. По введённой дате определить число дней в году и число дней в месяце.
17. По введённой дате определите животное восточного гороскопа.
18. По введённой дате определите знак зодиака
19. По введённому числу выведите его характеристики: знак, чётность, количество разрядов, тип данных переменной, в которой число может быть сохранено и т.д.
20. По введённому символу определить его характеристики: буква, регистр, язык, цифра, знак и т.д.
21. По введённой букве определить её характеристики: большая или маленькая, гласная или согласная, звонкая или глухая и т.д.

22. По введённой длине волны светового излучения определить цвет.
23. По введённой частоте электромагнитного излучения определить вид излучения.
24. По введённому напряжению и виду тока определить степень опасности для человека для случаев переменного и постоянного токов.
25. С вероятностью 0,5 вывести сообщение о проигрыше, с вероятностью 0,3 о ничьей, с вероятностью 0,2 о выигрыше.
26. Написать программу, которая последовательно выводит окна `MsgBox` с разными наборами кнопок. В каждом последующем окне отображать названия выбранных ранее кнопок.
27. Определить, является ли введённая фраза названием одной из изучаемых дисциплин.
28. Определить, является ли введённая фраза предложением, содержащим, по крайней мере, два слова.
29. Скидка в магазине осуществляется по серебряной (5%), золотой (10%) и платиновой (15%) дисконтным картам. Кроме того, скидка зависит от итоговой суммы покупки – 5% если сумма больше 999 руб. По введённой сумме стоимости товаров и виду дисконтной карты определить, сколько должен заплатить покупатель.
30. По введённым оценкам за сессию определить размер стипендии.

Контрольные вопросы

1. В каких случаях допускается применение оператора безусловного перехода?
2. Для чего служит базовая алгоритмическая структура «Ветвление»?
3. Чем блок `If` отличается от оператора `If`?
4. В чём преимущество блока `If Then ElseIf` перед блоком `If`?
5. В каких случаях целесообразно использование оператора выбора?
6. Каким операторам соответствует каждая из функций выбора?
7. Нарисуйте схемы алгоритмов каждой из функций выбора?

8. В каких случаях целесообразно использование функций выбора вместо соответствующих операторов?

4. Реализация базовой алгоритмической структуры «Цикл»

Алгоритмическая структура «Цикл» организует многократное выполнение одного и того же действия или последовательности действий. Многократно повторяемые действия составляют **тело цикла**. Очередное повторение называется **итерацией**. Прекращение повторений и выход из цикла осуществляется при выполнении определённого *условия*.

Различают циклы *с предусловием, с постусловием* и *с параметром*. В VBA реализованы все типы циклов.

4.1. Организация циклов

Для организации цикла следует определить *тело цикла*, *условие*, при выполнении которого выполняется итерация цикла, механизм *модификации* условия при каждой итерации в составе тела цикла, чтобы выполнение цикла завершилось в нужный момент, и механизм *инициализации* условия для обеспечения нормального входа в процесс итераций. В зависимости от задачи, определяющей характер *условия*, его *инициализации* и *модификации*, выбирается *тип цикла*.

Оператор For – Next

Цикл *For – Next* относится к циклам *с параметром*. Количество повторений зависит от параметра оператора цикла.

Синтаксис

```
For <Счетчик> = <Начало> To <Конец> [Step <Шаг>]
[Инструкции]
[Exit For]
[Инструкции]
Next <Счетчик>
```

Повторяет выполнение группы инструкций, пока значение переменной *Счетчик* изменяется от начального до конечного с указанным шагом. Если ключевое слово *Step* отсутствует, шаг полагается равным 1. Досрочный выход из цикла осуществляется с помощью инструкции *Exit For*. Алгоритм цикла *For – Next* приведён на Рис.29. Для обозначения циклов с модификаторами применяется также сокращённая запись алгоритма Рис.30.

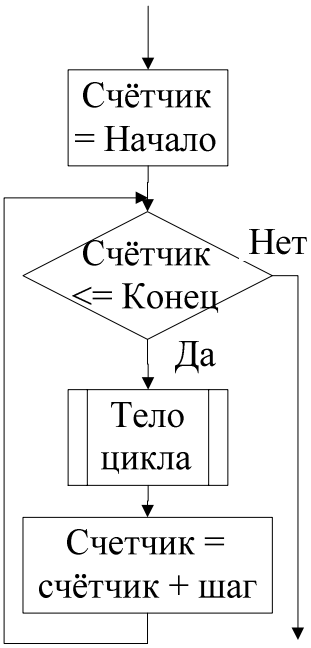


Рис.29. Цикл For -

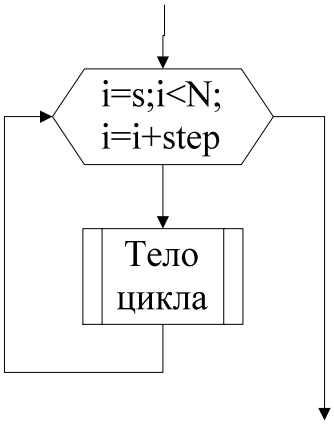


Рис.30. Цикл с параметром.

Цикл *For – Next* предназначен для использования в тех случаях, когда число повторений известно до начала выполнения цикла.

Пример

Заполнить массив из 20 элементов случайными числами и вывести их.

Схема алгоритма

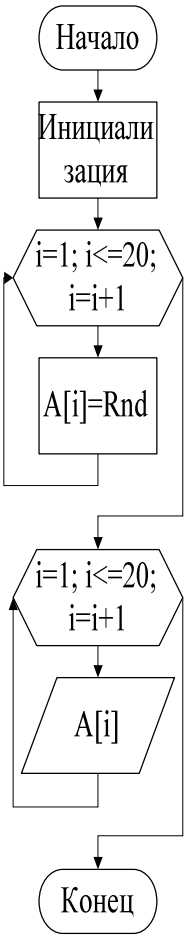


Рис.31. Схема алгоритма генерации и вывода массива из 20 случайных чисел.

Текст программы

```
Sub RandArr () ‘процедура
‘опишем счётчик (целое число) и массив
Dim i As Integer, A(20) As Single
Randomize ‘инициализация ГСЧ
For i = 0 To 19 ‘запускаем цикл от 0 до 19
‘записываем в i-й элемент массива случайное число
  A (i) = Rnd()
Next I ‘заканчиваем цикл
For i = 0 To 19 ‘ запускаем цикл от 0 до 19
‘в документ выводим i-й элемент и знак абзаца
```

```
  Selection.TypeText(Str(A(i))+Chr(13))
Next I 'заканчиваем цикл
End Sub 'заканчиваем процедуру
```

Оператор For Each – Next

Цикл *For Each - Next* так же относится к циклам с параметром.

Синтаксис

```
For Each <Элемент> In <Группа>
[Инструкции]
[Exit For]
[Инструкции]
Next <Элемент>
```

Повторяет выполнение группы инструкций для каждого элемента массива или семейства. Альтернативный способ выхода из цикла предоставляет инструкция *Exit For*. Схема алгоритма приведена на Рис.32.

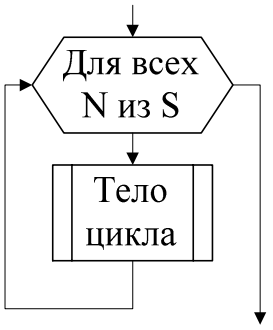


Рис.32. Цикл
For Each - Next

Пример

Для каждого абзаца текущего документа установить отступ слева со сдвигом на миллиметр относительно предыдущего.

Схема алгоритма

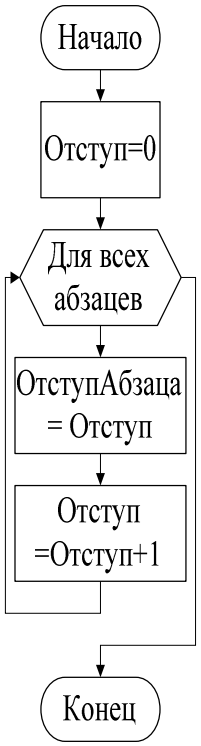


Рис.33. Алгоритм изменений отступов абзацев.

Текст программы

```
Sub ForEach() 'Заголовок процедуры
'Описание переменной – величины сдвига
Dim i As Single
i = 0 'Начальный сдвиг равен нулю
'Для всех абзацев текущего документа:
For Each Paragraph In ActiveDocument.Paragraphs
'Отступ слева устанавливаем в i мм.
'Для этого преобразуем миллиметры в пункты.
  Paragraph.LeftIndent = MillimetersToPoints(i)
  i = i+1 'Увеличиваем величину отступа на 1 мм.
Next Paragraph 'Переходим к следующему абзацу.
End Sub 'Конец процедуры
```

Оператор Do Until – Loop

Цикл *Do Until – Loop* относится к циклам с предусловием. Это означает, что условие выхода из цикла проверяется перед выполнением операторов, составляющих тело цикла. Следовательно, если условие выхода выполняется до начала цикла, инструкции цикла не будут выполнены ни разу.

Синтаксис

```
Do Until <Условие>
[Инструкции]
[Exit Do]
[Инструкции]
Loop
```

Повторяет выполнение набора инструкций, пока условие не примет значение *True*. Сначала проверяется условие, а потом выполняется инструкция. Альтернативный способ выхода из цикла предоставляет инструкция *Exit Do*.

Схема алгоритма приведена на Рис. 34.

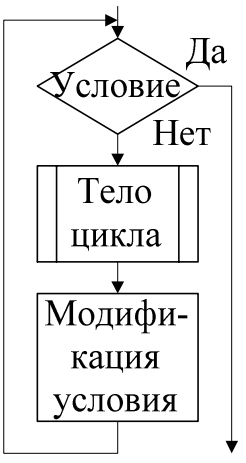


Рис. 34. Цикл
Do Until - Loop

Оператор Do — Loop Until

Цикл *Do – Loop Until* относится к циклам с постусловием. Это означает, что условие выхода из цикла проверяется после выполнения операторов, составляющих тело цикла. Следовательно, инструкции цикла будут выполнены хотя бы один раз.

Синтаксис

```
Do
[Инструкции]
[Exit Do]
[Инструкции]
Loop Until <Условие>
```

Повторяет выполнение набора инструкций, пока условие не примет значение *True*. Условие проверяется после выполнения инструкции, по крайней мере, один раз. Альтернативный способ выхода из цикла предоставляет инструкция *Exit Do*.

Схема алгоритма приведена на Рис.35.

Рис.35.
Цикл Do - Loop Until

Пример

Ввести число от 0 до 20.

Схема алгоритма

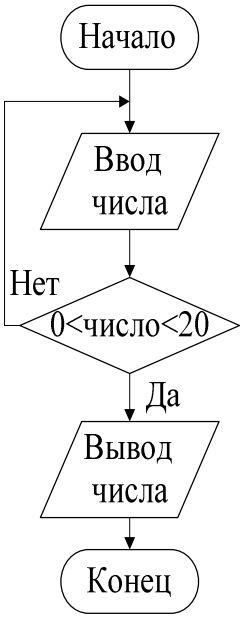


Рис.36. Схема алгоритма ввода числа больше 0 и меньше 20

Текст программы

```
Sub InputNum1() 'Начало процедуры
'Переменная, в которую вводим число
Dim Num As Integer
Do 'Начало цикла
'Вводим число
  Num = InputBox("Введите число от 0 до 20")
'Если 0<число<20 конец цикла.
Loop Until (Num>0)And(Num<20)
MsgBox (Num) 'Выводим число
End Sub Конец процедуры
```

Оператор Do While — Loop

Цикл *Do While – Loop* относится к циклам с предусловием. В отличие от цикла *Do Until – Loop* выход из цикла происходит, когда

условие, стоящее в заголовке, не выполняется. То есть это условие является условием продолжения работы цикла.

Синтаксис

```
Do While <Условие>
[Инструкции]
[Exit Do]
[Инструкции]
Loop
```

Повторяет выполнение набора инструкций, пока условие имеет значение *True*. Условие проверяется после выполнения инструкции, по крайней мере, один раз. Альтернативный способ выхода из цикла предоставляет инструкция *Exit Do*.

Схема алгоритма изображена на Рис. 37.

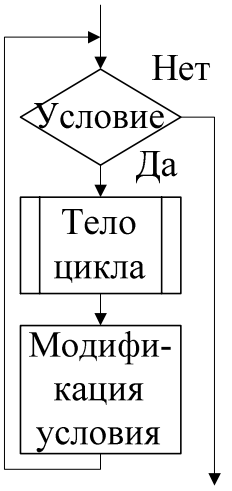


Рис. 37. Цикл Do While - Loop

Пример

Ввести число от 0 до 20. В случае ошибки выводить замечание.

Схема алгоритма

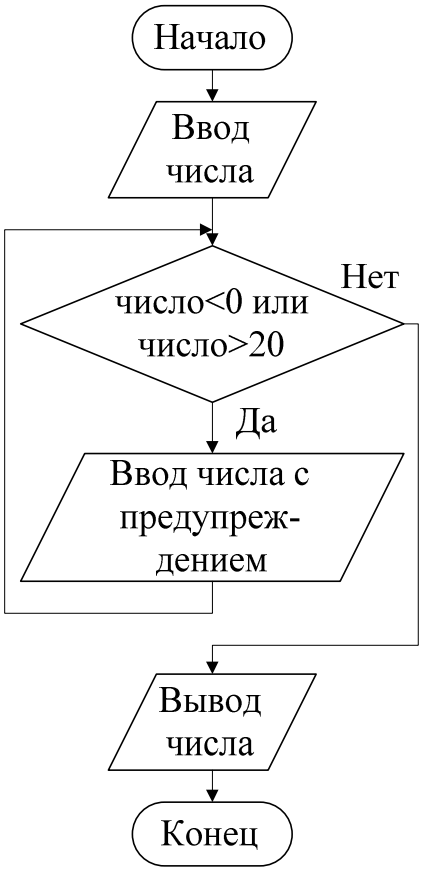


Рис.38. Алгоритм ввода числа с выводом замечания в случае ошибки

Текст программы

```
Sub InputNum2() 'Начало процедуры
'Переменная, в которую вводим число
Dim Num As Integer
'Ввод числа
Num = InputBox("Введите число от 0 до 20")
'Если выполняется – цикл
Do While (Num<=0)Or(Num>=20)
'Ввод числа, при этом выдаётся замечание
  Num=InputBox("Будте внимательны! От 0 до 20!")
Loop 'Конец цикла
MsgBox (Num) 'Вывод числа
End Sub 'Конец процедуры
```

Оператор Do — Loop While

Цикл *Do – Loop While* относится к циклам с постусловием. В отличие от цикла *Do – Loop Until* выход из цикла происходит, когда условие, стоящее в заголовке, не выполняется. То есть это условие является условием продолжения работы цикла.

Синтаксис

```
Do
[Инструкции]
[Exit Do]
[Инструкции]
Loop While <Условие>
```

Повторяет выполнение набора инструкций, пока условие имеет значение *True*. Сначала проверяется условие, а потом выполняется инструкция. Альтернативный способ выхода из цикла предоставляет инструкция *Exit Do*.

Схема алгоритма изображена на Рис. 39.

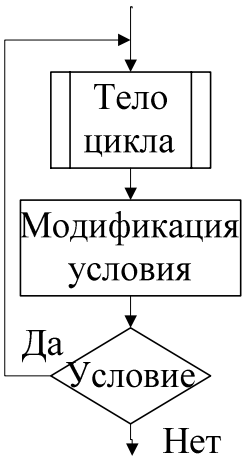


Рис. 39.
Цикл Do -
Loop While.

Оператор While — Wend

Цикл *While - Wend* является реализацией классического цикла с предусловием, в котором заданное условие является условием продолжения работы цикла. Фактически этот цикл дублирует цикл *Do While - Loop*, при этом обладает меньшими возможностями.

Синтаксис

```
While Условие
[Инструкции]
Wend
```

Выполняет последовательность инструкций, пока заданное условие имеет значение *`True`*.

Схема алгоритма соответствует схеме алгоритма цикла *Do While - Loop* (Рис. 37)

4.2. Массивы

Массив – упорядоченная совокупность переменных одного типа, объединённых общим именем. Доступ к каждому элементу массива осуществляется по его порядковому номеру – индексу.

Массив относится к структурированному типу данных.

Массивы применяются для хранения данных одного типа, логически связанных в некоторую совокупность, например, коэффициентов ряда, результатов измерений, зарплаты сотрудников и т.п.

В связи с тем, что элементов массива может быть много, учитывая, что при вводе или обработке массива с каждым элементом выполняются, как правило, однотипные действия, для работы с массивами чаще всего применяются алгоритмы циклической структуры.

Нумерация элементов массива осуществляется начиная с 0 (по умолчанию) или с 1 (После применения инструкции *`Option Base = 1`*). Кроме того, при описании массива можно указать любое начальное значение индекса.

```
   Dim R(10) As Integer ‘описание массива с именем «R» из 11 целых чисел. Номера элементов изменяются от 0 до 10.

R(1) =2 ‘присваивает второму элементу массива значение 2

A=R(10) ‘присваивает значение последнего элемента массива переменной «A».

Dim B(1 To 10, 1 To 16) As Single ‘описание двумерного массива вещественных чисел размерности 10x16, номера элементов начинаются с единицы.
```

Динамические массивы

Если при описании массива указана размерность, память под элементы массива отводится перед началом выполнения программы. В случае, когда размер массива заранее неизвестен, имеется возможность создать такой массив, размер которого задаётся и может изменяться в процессе выполнения программы. Такой тип массива называется динамическим. Работа с динамическими массивами требует известной аккуратности, так как по ошибке или недосмотру можно задать размер массива, на который у операционной системы может не хватить свободной памяти. В результате это может привести не только к аварийному завершению программы, но и к сбою в операционной системы и её аварийному завершению.

```
Dim D() As Single. ‘Описание динамического массива
```

Для определения размерности массива обязательно использование функции *ReDim*.

Синтаксис

```
ReDim [Preserve] <ИмяПеременной>(<Индексы>)[ As <Тип>][,<ИмяПеременной>(<Индексы>)[ As <Тип>]...]
```

Аргументы

Preserve – ключевое слово, используемое для сохранения данных в существующем массиве при изменении значения последней размерности

ИмяПеременной – имя переменной, удовлетворяющее стандартным правилам именования переменных

Индексы – Размерности переменной массива; допускается описание до 60 размерностей. Аргумент индексы использует следующий синтаксис: [<Нижний> To] <Верхний>[, [<Нижний> To] <Верхний>]. Если нижний индекс не задан явно, нижняя граница массива определяется инструкцией *Option Base.* Если отсутствует инструкция *Option Base,* нижняя граница массива равняется нулю

Тип – Тип данных массива

Пример

```
Redim D(1 To 5) 'Массив из 5 элементов.
```

Функция Array

Для создания массива, значения элементов которого известны, можно применять функцию *Array*.

Синтаксис

```
Array (<СписокАргументов>)
```

Возвращаемое значение

Возвращает массив типа *Variant*.

Аргументы

СписокАргументов – разделенный запятыми список значений, присваиваемых элементам массива.

Пример

Описание массива «День», элементам которого присвоены сокращённые названия дней недели:

```
   Dim День As Variant
День = Array ("Пн", "Вт", "Ср", "Чт", "Пт")
```

Определение границ массива

Для определения номеров первого и последнего элементов массива используются функции *LBound* и *UBound*.

Синтаксис

```
LBound (<ИмяМассива> [, <Размерность>])
UBound (<ИмяМассива> [, <Размерность>])
```

Возвращаемое значение

Минимальное и максимальное допустимое значение указанной размерности массива.

Аргументы

ИмяМассива — имя переменной массива

Размерность — целое число, указывающее размерность, нижнюю или верхнюю границу которой возвращает функция. Для первой размерности следует указать 1, для второй 2 и т. д. Если аргумент Размерность опущен, подразумевается значение 1

Ликвидация массива

Функция *Erase* служит для очистки или повторной инициализации массива.

Синтаксис

```
Erase <СписокМассивов>
```

Повторно инициализирует элементы массивов фиксированной длины и освобождает память, отведенную для динамического массива.

Аргументы

`СписокМассивов` представляет имена одной или нескольких очищаемых переменных массивов, разделенных запятой. Инструкция *`Erase`* устанавливает элементы массивов фиксированной длины следующим образом: массив чисел или строк фиксированной длины (присваивает каждому элементу значение 0), массив строк переменной длины (присваивает каждому элементу значение пустой строки), массив типа *`Variant`* (присваивает каждому элементу значение *`Empty`*). *`Erase`* освобождает память, используемую динамическими массивами. Перед тем как из программы вновь появится возможность сослаться на динамический массив, необходимо переопределить размерности переменной массива с помощью инструкции *`ReDim`*.

Лабораторная работа №4

Цель работы: получить навыки разработки программ с использованием циклических структур.

Задание

1. Разработать алгоритм и программу на VBA по варианту.
2. Ввести программу в редактор VBA и добиться её выполнения.
3. Ответить на контрольные вопросы.
4. Подготовить отчёт.

Содержание отчёта

1. Титульный лист.
2. Индивидуальное задание.
3. Схема алгоритма.
4. Текст программы с комментариями.
5. Ответы на контрольные вопросы.

Индивидуальные задания

1. Сформируйте одномерный массив длиной N<=20, состоящий из целых чисел. Подсчитайте количество минимальных элементов.
2. Сформируйте одномерный массив длиной N<=20, состоящий из целых чисел. Определите, на каких позициях находятся максимальные элементы.
3. Сформируйте одномерный массив длиной N<=20, состоящий из целых чисел. Поменяйте первый минимальный элемент и последний максимальный элемент;
4. Одномерный массив длиной N<=25 заполните целыми числами из диапазона [x1..x2]. Определить позиции и количество элементов, значения которых лежат в диапазоне [y1..y2].
5. Одномерный массив заполните числами из диапазона [–x..x]. Переместите отрицательные элементы массива в конец, сдвинув остальные элементы влево.
6. Сформируйте массив целых чисел и определите количество и позиции четных, нечётных и нулевых элементов.
7. Сформируйте массив чисел из диапазона [-x, x] и определите суммы положительных и отрицательных чисел, не превышающих по модулю заданного значения.
8. Сформируйте массив чисел из диапазона [-x, x] и определите максимальное отрицательное и минимальное положительное число.
9. Сформируйте массив чисел из диапазона [-x, x]. Определите максимальное количество подряд идущих положительных элементов последовательности, не прерываемых ни нулями, ни отрицательными элементами. Напечатайте найденный фрагмент.
10. Сформируйте массив целых чисел и определите максимальное количество подряд идущих одинаковых элементов.
11. Сформируйте массив целых чисел и определите максимальное расстояние между парой одинаковых чисел.
12. Сформируйте массив целых чисел и определите количество совокупностей подряд идущих одинаковых элементов.
13. Сформируйте массив чисел и определите, является ли он упорядоченным по неубыванию.

14. Сформируйте массив чисел и вычислите среднее арифметическое, среднее геометрическое и среднее квадратическое.
15. Сформируйте массив целых чисел и выведите числа, присутствующие в массиве и частоты их появления.
16. Сформируйте массив чисел и выведите массив, состоящий из разностей между соседними элементами.
17. Сформируйте массив целых чисел и определите количество противоположных по знаку одинаковых по модулю чисел.
18. Сформируйте массив чисел и поменяйте местами элементы, отстоящие друг от друга на заданное число позиций.
19. Сформируйте массив чисел и определите позиции элементов, величина которых больше среднего значения.
20. Сформируйте массив чисел и отсортируйте его по возрастанию.
21. Сформируйте массив дат и выведите даты, приходящиеся на заданное число.
22. Выведите таблицы значений тригонометрических функций с заданным шагом.
23. Переведите заданное десятичное число в двоичную систему счисления.
24. Сформируйте таблицу номеров «счастливых» билетов, подсчитайте их количество и вероятность получить «счастливый» билет.
25. Вычислите число Эйлера с заданной точностью, используя ряд.
26. Введите строку и выведите отдельные слова в алфавитном порядке.
27. Введите целое число и выведите массив, состоящий из всех простых сомножителей заданного числа.
28. Введите строку и вычислите среднюю длину слов.
29. Введите строку и выведите слова, длина которых больше заданной.
30. Ввести строку и осуществить замену одного заданного слова на другое.

Контрольные вопросы

1. Что такое циклы и для чего они нужны в программе?

2. В чём различия циклов с параметром, с предусловием и с постусловием?
3. Какие циклы с параметром вы знаете? В каких случаях их следует применять?
4. Какие циклы с предусловием вы знаете? В каких случаях их следует применять?
5. Какие циклы с постусловием вы знаете? В каких случаях их следует применять?
6. В чём преимущества и недостатки цикла «While - Wend» по сравнению с циклом «Do While - Loop»?
7. Что такое и зачем нужны массивы?
8. Почему применение динамических массивов может негативно повлиять на стабильность работы операционной системы?
9. Что такое размерность массива? Какое количество размерностей массива допустимо?
10. С чего начинается нумерация элементов массива?
11. Почему массивы целесообразно обрабатывать, используя алгоритм циклической структуры? Поясните на примере.

Пример выполнения работы

Задание

Сформируйте массив целых чисел из диапазона [-x, x], x<=100, длиной N<=20 и определите произведение ненулевых элементов.

Схема алгоритма приведена на рис. 40.

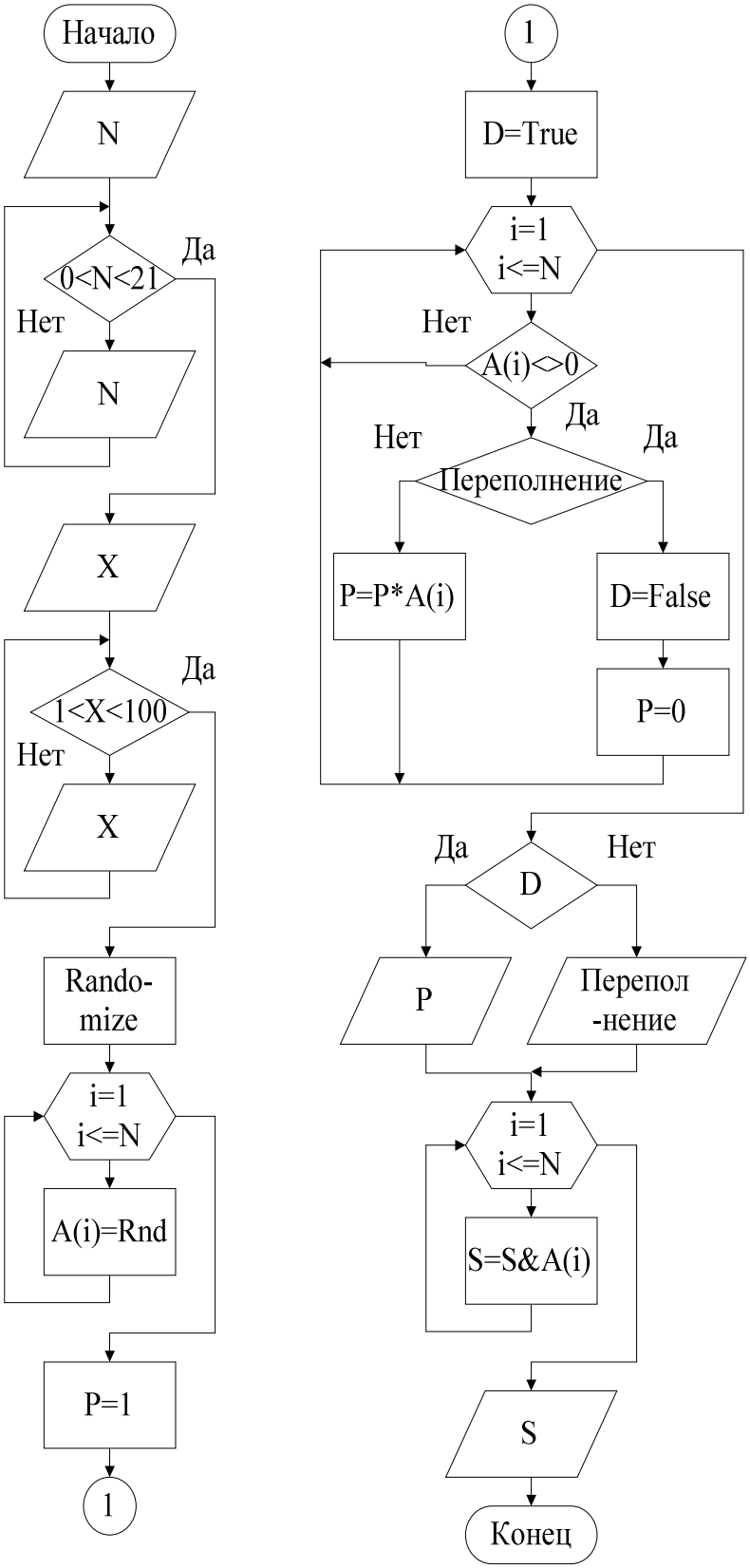


Рис.40. Алгоритм программы вычисления произведения ненулевых элементов массива.

Текст программы

```
Sub Example()
Dim A(1 To 20), x, i, N As Integer, P As Long
Dim D As Boolean, S As String
N = InputBox("Введите длину массива от 1 до 20")
Do While (N < 1) Or (N > 20)
  N = InputBox("Будте внимательны! От 1 до 20!")
Loop
x = InputBox("Введите диапазон чисел (до 100)")
Do While (Abs(x) < 1) Or (Abs(x) > 100)
  x = InputBox("Будте внимательны! От 1 до 100!")
Loop
Randomize
For i = 1 To N
'Целые случайные числа из [-x,x]
  A(i) = Int(Rnd * 2 * x - x)
Next i
P = 1 'Произведение начинаем вычислять с 1
D = True 'Не будет переполнения - останется True
For i = 1 To N 'Цикл для всех элементов
  If (A(i) <> 0) Then 'Если элемент ненулевой
'Если тип Long не переполнится, перемножаем.
      If Abs(2147483647 / A(i)) > Abs(P) Then
          P = P * A(i) '
        Else 'Иначе
          D = False 'Если переменная переполнится
          P = 0 'произведение не вычислить...
      End If
  End If
Next i
If D Then MsgBox("Произведение = " & Str(P)) _
Else MsgBox ("Произведение слишком велико!")
'Формируем строку из массива
For i = 1 To N
  S = S & Str(A(i)) & " "
Next i
```

```
MsgBox ("Массив: " & S) 'Выводим строку
End Sub
```

5. Структурное программирование на VBA

В главе излагаются основы структурного программирования с использованием процедур и функций. Также рассматриваются приёмы работы с Host приложениями, использования структурированных данных и базовых алгоритмических конструкций.

VBA относится к языкам структурного программирования за счёт реализации базовых алгоритмических конструкций и подпрограмм в соответствии с принципами структурного программирования.

5.1. Структурное программирование

Структурное программирование – методология разработки программ, определяющая состав программы в виде иерархической структуры, основными элементами которой являются базовые алгоритмические конструкции.

К базовым алгоритмическим конструкциям относятся «Следование», «Ветвление» и «Цикл». В рамки методологии не вписывается оператор безусловного перехода GoTo, поэтому его использование при написании структурированной программы запрещено. Теорема Бёма-Джакопини доказывает, что любой алгоритм можно привести к структурированному виду. Замечено [1] что структурированные алгоритмы легче поддаются анализу, проще для понимания и лучше приспособлены для модифицирования, так как каждая базовая алгоритмическая структура имеет один вход и один выход и её можно рассматривать изолированно, не вдаваясь в подробности функционирования остальных частей программы.

С точки зрения математического анализа структурное программирование базируется на теории рекурсивных функций, программа рассматривается как частично-рекурсивный оператор над подпрограммами и операциями. Также, структурное программирование базируется на теории доказательств, прежде всего на естественном выводе в рамках интуитивистской логики. В этом смысле, структура программы соответствует структуре простейшего математического рассуждения, лишённого сложных лемм и абстрактных понятий [2].

При создании алгоритма это значит, что каждая реализуемая задача разбивается на последовательность логически и функционально завершённых подзадач, а информация – на достаточно независимые структуры данных, среди которых выделяются и обособляются данные, локальные для каждой подзадачи. Для каждой из подзадач определяется формализованный набор входных и выходных данных. Каждая подзадача может быть в свою очередь разбита по тому же принципу. Разбиение происходит до тех пор, пока это выглядит целесообразно и соответствует здравому смыслу, а алгоритм не будет содержать только те команды, которые могут быть выполнены исполнителем. На каждом этапе используются только базовые алгоритмические конструкции. Такой принцип разбиения задачи на подзадачи называется структурным программированием сверху вниз или нисходящим программированием.

Часто в алгоритмах на разных этапах решения задачи могут быть выделены последовательности действий, выполняющие одну и ту же подзадачу. Такие последовательности можно обнаружить, рассматривая предельно подробное описание алгоритма программы, либо предположить их существование исходя из условий задачи. Эти последовательности действий называются «типовой процесс» и оформляются как подзадача, которая описывается однократно, и в дальнейшем многократно может быть использована. Такой принцип разбиения задачи на подзадачи называется структурным программированием снизу вверх или восходящим программированием.

Для решения масштабной задачи в первую очередь целесообразно использовать нисходящее программирование. Для применения восходящего программирования необходимо либо уже иметь готовый алгоритм, который для большой задачи будет громоздким и, как следствие, трудным для анализа, либо базировать разработку тех или иных подзадач на гипотезах относительно их целесообразности в составе решения задачи. Гипотезы в конечном итоге могут оказаться частично или полностью неверными, что приведёт к необходимости переделывать решение подзадач заново. В то же время, после завершения нисходящего программирования создаются возможности для применения восходящего программирования, так что эти подходы взаимно дополняют друг друга.

5.2. Реализация подпрограмм на VBA.

Средствами записи подзадач на язык программирования VBA являются подпрограммы, которые записываются однократно и в дальнейшем их выполнение вызывается из любой точки программы, где это необходимо. После того, как подпрограмма закончила своё выполнение, программа продолжает выполнение с той позиции, из которой была вызвана подпрограмма.

В VBA имеется три типа подпрограмм: процедуры, функции и подпрограммы *GoSub – Return*. Каждый из типов имеет свои области применения и отличается особенностями описания и вызова.

Описание подпрограмм GoSub - Return

В VBA от первоначальных версий BASIC сохранилась конструкция подпрограммы *GoSub – Return*, которая в настоящее время используется редко. Для полноты изложения кратко приведены правила использования этой инструкции.

Синтаксис вызова и описания подпрограммы:

```
GoSub <Строка>
[Инструкции программы]
...
<Строка>
[Инструкции подпрограммы]
Return
```

Строка - метка или номер строки. В качестве метки строки может быть любая удовлетворяющая правилам именования комбинация символов, заканчивающаяся двоеточием.

Допускается использование инструкций `GoSub` и `Return` в любом месте процедуры, но `GoSub` и соответствующая инструкция `Return` должны находиться в одной процедуре. Подпрограмма может содержать несколько инструкций `Return`. Первая обнаруженная инструкция `Return` осуществляет переход обратно к инструкции, следующей за инструкцией `GoSub`, вызвавшей подпрограмму.

Описание процедур и функций.

Процедура – поименованная изолированная часть программы.

Функция – процедура, определяющая зависимость возвращаемого значения от значений аргументов.

При использовании процедур и функций, VBA в соответствии с условиями структурного программирования особым образом организовывает работу с переменными и их значениями, которые применяются во всей программе, внутри процедуры, передаются в процедуру и возвращаются из процедуры в процессе её вызова.

Функция на VBA является разновидностью процедуры, отличающейся только тем, что когда требуется использовать возвращаемое значение, функция может применяться в правой части выражения, в то время как значения, возвращаемые процедурой, могут храниться только в переменных, указанных в списке фактических параметров и использоваться только в следующих за вызовом инструкциях. В то же время, если не требуется использовать возвращаемое значение функции, она может применяться в стиле процедуры.

Области видимости имен

Область видимости имени задает участки программы, в которых поименованный объект может быть использован. В VBA имеется три уровня области видимости:

1. Объекты уровня процедуры (локальные) используются только в процедуре или функции, в теле которой они описаны.
2. Объекты уровня модуля (глобальные) описываются в модуле вне процедур и функций и используются только в модуле, в теле которого они описаны.
3. Объекты уровня проекта (общие) используются во всех модулях данного проекта. Описываются в теле модуля вне процедур и функций при помощи инструкции Public.

Время жизни переменной

Локальная переменная сохраняет свое значение, пока выполняется процедура, в которой эта переменная описана. При завершении процедуры значение переменной теряется, и при повторном запуске процедуры переменную надо заново инициализировать.

Локальные статические переменные, описанные при помощи инструкции *Static*, сохраняют свое значение при выходе из процедуры, но пока работает программа. При этом видны они будут только после входа в процедуру.

Глобальные и общие переменные сохраняют свои значения в течение всего времени работы программы.

Параметры

В процедурах, задача которых состоит в преобразовании данных программы, используется формализованное описание входных и выходных данных, называемых параметрами или аргументами. В пределах процедуры с ними обращаются как с проинициализированными локальными личными переменными.

При вызове тем или иным способом указывается набор передаваемых в процедуру значений, который называется списком фактических параметров.

Параметры передаются в процедуру одним из двух способов: по ссылке или по значению. Если параметр передаётся по значению, в процедуру передаётся только значение аргумента, создаётся локальная переменная для хранения копии этого значения и при изменении значения локальной переменной содержимое внешней переменной не изменяется. Если параметр передаётся по ссылке, в процедуру передаётся адрес ячейки памяти, в которой хранится значение параметра. Вследствие этого при использовании в качестве фактического параметра переменной любые изменения параметра внутри процедуры изменяют значения переданных в процедуру внешних переменных.

Синтаксис описания процедуры:

```
[Private | Public] [Static] Sub <Имя>
([СписокАргументов])
[Инструкции]
[Exit Sub]
[Инструкции]
End Sub
```

Аргументы

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. По умолчанию используется *Private*.

Static – Указывает, что локальные переменные процедуры сохраняются в промежутках времени между вызовами этой процедуры.

Имя процедуры *Sub* должно удовлетворять стандартным правилам именования в VBA.

СписокАргументов – Список переменных, воспринимающих аргументы, которые передаются в процедуру *Sub* при ее вызове. Имена переменных разделяются запятой.

Инструкции – Группа инструкций, выполняемых в процедуре.

Exit Sub — Инструкция, приводящая к немедленному выходу из процедуры *Sub*.

End Sub — конец процедуры.

Синтаксис элемента *СписокАргументов:*

```
[Optional] [ByVal | ByRef] [ParamArray]
<ИмяПеременной>[()][As <Тип>] [= поУмолчанию]
```

Аргументы

Optional — Ключевое слово, указывающее, что аргумент не является обязательным. При использовании этого элемента все последующие аргументы, которые содержатся в списке *СписокАргументов*, также должны быть необязательными, и описаны с помощью ключевого слова *Optional*. Все аргументы, описанные как *Optional*, должны иметь тип *Variant*. Не допускается использование ключевого слова *Optional* для любого из аргументов, если используется ключевое слово *ParamArray*.

ByVal — Указывает, что этот аргумент передается по значению.

ByRef — Указывает, что этот аргумент передается по ссылке. Описание *ByRef* используется в VBA по умолчанию.

ParamArray — Используется только в качестве последнего элемента в списке *СписокАргументов* для указания, что конечным аргументом является *Optional* массив значений типа *Variant*. Позволяет задавать произвольное количество аргументов. Не может быть использовано со словами *ByVal, ByRef* или *Optional.*

ИмяПеременной — Имя переменной, удовлетворяющее стандартным правилам именования переменных.

Тип — Тип данных аргумента, переданного в процедуру; поддерживаются переменные типов *Byte, Boolean, Integer, Long, Currency, Single, Double, Date, String* (только строки переменной длины), *Object, Variant.* Если отсутствует ключевое слово *Optional,* могут быть также указаны определяемый пользователем тип или объектный тип.

поУмолчанию — Любая константа или выражение, дающее константу. Используется только вместе с параметром *Optional.* Если указан тип *Object,* единственным значением по умолчанию может быть значение *Nothing.*

Синтаксис описания функции:

```
[Public|Private][Static]Function<Имя>([СписокАргументов])[As<Тип>]
[Инструкции]
[Имя = Выражение]
[Exit Function]
[Инструкции]
[Имя = Выражение]
End Function
```

Синтаксис инструкции *Function* содержит те же ключевые слова, что и *Sub.*

Для возврата значения из функции следует в теле функции присвоить значение имени функции. Любое число таких инструкций

присвоения может находиться в любом месте функции, возвращено будет последнее присвоенное значение.

Вызов процедур и функций

Вызов процедур и функций подразумевает передачу управления подпрограмме, передачу данных в виде фактических параметров в качестве аргументов и возврат из подпрограммы в точку вызова.

Синтаксис вызова процедуры:

```
[Call] <ИмяПроцедуры>
(<СписокФактическихПараметров>)
```

Значения ключевых слов

ИмяПроцедуры – имя вызываемой процедуры.

СписокФактическихПараметров - список аргументов, передаваемых процедуре. Он должен соответствовать списку, заданному в описании процедуры по количеству и типу элементов. Элементы списка разделяются запятой. Необязательные *(Optional)* параметры можно пропускать, отделяя их позиции запятыми.

Если требуется использовать несколько процедур из разных модулей с одинаковыми названиями, при их вызове после имени процедуры через точку надо указывать имя модуля, на котором они расположены:

```
ИмяМодуля.ИмяПроцедуры.
```

Команду *Call* можно опускать, тогда VBA требует, чтобы список фактических параметров не заключается в скобки.

Язык VBA позволяет вводить фактические параметры через имена аргументов в любом порядке. При этом после имени аргумента ставятся двоеточие и знак равенства, после которого помещается значение аргумента (фактический параметр).

Функция при вызове возвращает значение результата в соответствии с описанным типом таким образом, что может вызываться в правой части оператора присваивания или в составе выражения.

Если нужно, чтобы функция только выполнила свои действия, и нет необходимости в возвращаемом значении, *функция* может быть вызвана в стиле *процедуры*. В этом случае, инструкция вызова функ-

ции, подобно вызову процедуры, состоит из имени функции и списка фактических параметров, который не заключается в скобки.

Основные способы передачи параметров в процедуры.

Функция:

```
'целочисленная функция с именем "sq" и аргументом
'"x" целого типа, передаваемым по значению:
Function sq (ByVal x As Integer) As Integer
  sq = x ^ 2 'вычисление возвращаемого значения
End Function 'конец функции
```

Процедура:

```
'процедура с именем "OutS", и целочисленными
'аргументами: "a", передаваемым по значению,
'"s" – по ссылке.
Sub OutS (ByVal a As Integer, ByRef s As Integer)
  s = a ^ 2 'вычисление результата
End Sub 'конец процедуры
```

Примеры передачи параметров:

```
Sub Main () 'главная процедура модуля
'x, y — переменные, используемые в качестве
'фактических параметров
  Dim x, y As Integer
'Вызов процедуры с конкретными числами как
'фактическими параметрами
  OutS 0, y
  MsgBox CStr(y) 'вывод результата
  x = 1 'присвоение значения переменной x
'вызов процедуры с использованием переменных в
'качестве фактических параметров
  OutS x, y
  MsgBox CStr(y) 'вывод результата
  x = 2 'присвоение значения переменной x
  Call OutS(x, y) 'вызов процедуры командой call
  MsgBox CStr(y) 'вывод результата
  y=sq(3) 'вызов функции
  MsgBox CStr(y) 'вывод результата
```

```
'вызов процедуры с использованием значения
'функции в качестве фактического параметра
  Call OutS(sq(2),y)
  MsgBox CStr(y) 'вывод результата
'вызов процедуры с указанием имён фактических
'параметров – допустимо в любом порядке для этого
'способа
  OutS s:=y, a:=5
  MsgBox CStr(y)'вывод результата
'использование выражения в качестве фактического
'параметра функции
  y=sq(x+sq(2))
  MsgBox CStr(y) 'вывод результата
End Sub 'Конец процедуры
```

Передача массива как параметра.

В качестве формальных параметров могут использоваться массивы. Массив может передаваться только по ссылке *(ByRef)*. При описании массивов в качестве параметров размер массива не должен указываться, и массив в заголовке функции описывается подобно динамическому. Предполагается, что при вызовах процедура получает массив, фактический размер которого в каждом случае определён вне процедуры и в разных случаях может различаться. В этих условиях необходимо использовать специальные приёмы для того чтобы каждый раз процедура правильно обрабатывала все элементы фактически переданного массива при любом их количестве. Кроме того, что количество элементов может быть передано в процедуру с помощью дополнительных параметров, применяются определение позиции первого и последнего элементов массива с помощью функций *LBound* и *UBound*, а также использование цикла *For – Each – Next* для обработки элементов массива. Для элементов массива в этом цикле должна использоваться переменная типа *Variant*. Оба эти способа рассмотрим на примере функции, вычисляющей сумму элементов массива.

Описание функции вычисления суммы элементов массива с определением позиций первого и последнего элементов:

```
'целочисленная функция с аргументом “Mas” типа
'“массив целых чисел”, передаваемым по значению.
Function SumMas1(Mas() As Integer) As Integer
  Dim I As Integer 'индекс элемента
  SumMas1 = 0 'Начальное значение суммы
'Цикл от первого до последнего элемента массива
  For i = LBound(Mas) To UBound(Mas)
  'Добавляется значение элемента в сумму
    SumMas1=SumMas1+Mas(i)
  Next I 'Переход к следующему элементу
End Function 'Конец функции
```

Описание функции вычисления суммы элементов массива с использованием цикла *For – Each – Next:*

```
'целочисленная функция с аргументом “Mas” типа
'“массив целых чисел”, передаваемым по значению.
Function SumMas2(Mas() As Integer) As Integer
  Dim M As Variant 'Элемент массива
  SumMas2 = 0 'Начальное значение суммы
'Цикл для всех элементов массива
  For each M in Mas
  'Добавляется значение элемента массива в сумму
    SumMas2=SumMas2+M
  Next M 'Переход к следующему элементу
End Function 'Конец функции
```

Вызов функций вычисления суммы

```
Sub sss() 'Процедура без параметров
'Массив в 20 целых чисел, индекс
  Dim M(1 To 20)As Integer, i As integer
  For i = 1 To 20 'Цикл от 1 до 20
    M(i) = Rnd * 100 'случайное число от 1 до 100
  Next i 'Переход к следующему элементу
'Вычисление суммы элементов массива с помощью
'функции SumMas1
  MsgBox (SumMas1(M))
```

```
'Вычисление суммы элементов массива с помощью
'функции SumMas2
  MsgBox (SumMas2(M))
End Sub 'Конец процедуры
```

Использование необязательных параметров

Использование необязательных параметров рассмотрим на примере функции *TriSide*, позволяющей найти длину недостающей стороны прямоугольного треугольника, где переменные *A* и *B* отведены под длины катетов, а переменная *C* – под гипотенузу. 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.

Описание функции *TriSide* с необязательными параметрами:

```
'вещественная функция с необязательными
'параметрами “a”, “b”, и “c” типа variant
Function TriSide(Optional a, Optional b, _
Optional c) As Single
'если присутствуют параметры “a” и “b”, вычисляем
'длину гипотенузы
If Not(IsMissing(a)) And Not(IsMissing(b)) Then _
TriSide = Sqr(a ^ 2 + b ^ 2)
'если присутствуют параметры “a” и “c” вычисляем
'длину катета
If Not(IsMissing(a)) And Not(IsMissing(c)) Then _
TriSide = Sqr(c ^ 2 - a ^ 2)
'если присутствуют параметры “b” и “c” вычисляем
'длину катета
If Not(IsMissing(b)) And Not(IsMissing(c)) Then _
TriSide = Sqr(c ^ 2 - b ^ 2)
End Function   'конец функции
```

Вызов функции *TriSide* с необязательными параметрами

```
Sub Triangle() 'процедура без параметров
  x = Val(InputBox("Введите катет 1"))
  y = Val(InputBox("Введите катет 2"))
```

```
'вызов функции TriSide с передачей первых двух
'фактических параметров
  MsgBox (Str(TriSide(x,y)))
  x = Val(InputBox("Введите катет 1"))
  y = Val(InputBox("Введите гипотенузу"))
'вызов функции TriSide с передачей первого и
'третьего фактических параметров.
  MsgBox (Str(TriSide(x,,y)))
  x = Val(InputBox("Введите катет 2"))
  y = Val(InputBox("Введите гипотенузу"))
'вызов функции TriSide с передачей двух последних
'фактических параметров.
  MsgBox (Str(TriSide(,x,y)))
End Sub
```

Для необязательного параметра можно определить значение по умолчанию. В следующем примере, если значение параметра *Str* не передано в процедуру, то ему присваивается указанное по умолчанию значение «Техническая пауза».

Описание функции с использованием значения по умолчанию для необязательного параметра:

```
'Процедура с необязательным строковым параметром
'Str, для которого определено значение по
'умолчанию
Sub TecMsg(Optional Str As String = _
“Техническая пауза”)
  MsgBox Str 'Вывод строки
End Sub 'Конец процедуры
```

Вызов функции с использованием значения по умолчанию

```
Sub TestTec() 'Процедура без параметров
'Вызов TecMsg без параметров, будет выведено
'значение по умолчанию
  TecMsg
'Вызов TecMsg с фактическим параметром
  TecMsg("Остановка")
End Sub 'Конец процедуры
```

Использование неопределенного количества параметров

Как правило, количество передаваемых параметров в процедуру совпадает с количеством определенных у этой процедуры параметров. Ключевое слово *ParamArray* предоставляет возможность ввода в процедуру произвольного, заранее не указанного числа параметров (например, как это происходит при использовании функции рабочего листа СУММ (SUM) в MS Excel). В качестве примера приведем функцию *СУММПОЛ*, которая выполняет то же действие, что и функция рабочего листа СУММ.

Описание функции с неопределённым числом параметров:

```
'функция типа Variant с неопределённым
'количеством параметров в виде массива
Function СУММПОЛ(ParamArray Массив())
'описание переменных
  Dim S As Variant, a As Variant, b As Variant
  S = 0 'инициализация суммы
'цикл для каждого элемента массива
  For Each a In Массив
    If IsNumeric(a) Then 'если он содержит число,
      S = S + a 'добавить его в сумму
  'иначе если он содержит массив
    ElseIf IsArray(a) Then
      For Each b In a 'каждый элемент массива
      'если число, добавить в сумму
        If IsNumeric(b) Then S=S+b
      Next b 'перейти к следующему элементу
    End If 'конец ветвления
  Next a 'перейти к следующему параметру
  СУММПОЛ = S 'возвращаемое значение равно сумме
End Function 'конец функции
```

Рекурсивные процедуры и функции

В VBA возможно создание рекурсивных процедур и функций, то есть, вызывающих самих себя. Стандартным примером рекурсии является вычисление факториала. Факториал натурального числа n вычисляется как произведение первых n натуральных чисел, причём

факториал нуля определён равным единице. Для этой функции имеет место соотношение: n!=n(n–1)!. Для вычисления (n-1)! каждый раз можно применить то же соотношение, пока не получим 0!. Основываясь на данном соотношении, приводимая ниже рекурсивная функция вычисляет значение факториала.

Описание рекурсивной функции для вычисления факториала:

```
'Целая функция и целый аргумент
Function Fact(n As Byte) As Long
  If n<2 Then 'Если значение аргумента меньше 2
    Fact = 1 'Значение функции равно 1
  Else 'Иначе
  'Значение функции вычисляется как n, умноженное
  'на значение функции от n-1
    Fact=Fact(n-1)*n
  End If 'Конец ветвления
End Function 'Конец функции
```

Другим примером применения рекурсивных функций является поиск наибольшего общего делителя (НОД) двух целых чисел по алгоритму Евклида. НОД двух целых чисел — это наибольшее целое, на которое делятся оба числа. Например, НОД(10, 14) = 2 и НОД (15, 31) = 1. Алгоритм Евклида состоит в следующем: Если a делится на b, то НОД(a, b) = b, иначе – НОД(a, b) = НОД(b, a Mod b).

Описание рекурсивной функции поиска НОД по алгоритму Евклида:

```
'Функция с двумя целыми параметрами
Function НОД(a As Long, b As Long) As Long
  If a Mod b = 0 Then 'Если a делится на b
    НОД = b 'НОД=b
  Else 'иначе
  'Рекурсивный вызов с новыми данными
    НОД=НОД(b,a Mod b)
  End If 'Конец ветвления
End Function 'конец функции
```

Цепочка рекурсивных вызовов не должна быть бесконечной. В тело рекурсивной функции необходимо включать условие выхода из рекурсии в том или ином виде. При вычислении факториала выход из

рекурсии происходит, когда значение аргумента меньше двух. Так как в цепочке вызовов значение аргумента каждый раз уменьшается на единицу, рано или поздно условие выхода выполнится. При вычислении наибольшего общего делителя остаток от деления в цепочке вызовов будет уменьшаться и рано или поздно достигнет нуля.

В процессе работы рекурсивных функций в оперативной памяти хранятся данные на всю глубину рекурсии, неаккуратное использование может привести к переполнению. Применение рекурсии почти всегда не совместимо с применением циклов, поэтому для реализации повторения однотипных действий следует применять либо то, либо другое, но никак не вместе [4].

В ряде случаев рекурсия существенно менее эффективна, чем цикл. Пример – вычисление значений ряда Фибоначчи, в котором каждое последующее число вычисляется как сумма двух предыдущих, начиная с единицы.

Описание функции вычисления числа Фибоначчи с помощью рекурсии:

```
Function FibR(a As Integer) As Long
If a <= 1 Then FibR = a Else FibR = FibR(a - 1) +
FibR(a - 2)
End Function
```

Описание функции вычисление числа Фибоначчи с помощью цикла:

```
Function FibC(a As Integer) As Long
  Dim i As Integer, f1 As Long, f2 As Long, _
  t As Long
  f1 = 1 'значение числа
  f2 = 0 'значение предыдущего числа
  'вычисляем с первого до предпоследнего
  For i=1 To a-1
    t = f1 'сохранение предыдущего числа
    'следующее равно текущее + предыдущее
    f1 = f1 + f2
    f2 = t 'предыдущее – то, что было текущим
  Next I '
  FibC = f1 '
End Function
```

В сравнении с циклической реализацией, рекурсивная функция содержит меньше строк, не содержит вспомогательных переменных, и, в целом, кажется более элегантной. Между тем, каждый вызов рекурсивной функции порождает два потока вызовов, один из которых на следующем уровне будет вычислять то же самое значение, что и другой на текущем. С увеличением номера элемента ряда количество лишних вызовов растёт в геометрической прогрессии, соответственно растут время выполнения и затраты памяти. Функция, реализованная с помощью цикла, кроме вспомогательных переменных никаких дополнительных затрат ресурсов не влечёт [4]. Следующий пример позволяет сравнить время вычисления числа Фибоначчи разными способами, или вычислительную мощность компьютеров.

Вызов и определение времени работы функций вычисления числа Фибоначчи с использованием рекурсии и цикла

```
Sub TestFib()
  Dim a As Integer, t As Single, f As Long, _
  d As Single
  a = Val(InputBox("a")) 'Номер числа Фибоначчи
  t = Timer 'Время перед началом вычисления
  f = FibR(a) 'Вычисление с помощью рекурсии
  d = Timer - t 'Время вычисления
  Selection.TypeText "Рекурсия:"+Str(a)+ _
  "-е число Фибоначчи =" + Str(f)+ _
  " вычислено за "+Str(d)+" с."+Chr(13)
  t = Timer 'Время перед началом вычисления
  f = FibC(a) 'Вычисление с помощью цикла
  d = Timer - t 'Время вычисления
  Selection.TypeText "Цикл:"+Str(a)+ _
  " -е число Фибоначчи =" + Str(f)+ _
  " вычислено за "+Str(d)+" с."+Chr(13)
End Sub
```

Результаты работы:

Рекурсия: 20-е число Фибоначчи = 6765 вычислено за 0 с.
Цикл: 20-е число Фибоначчи = 6765 вычислено за 0 с.
Рекурсия: 30-е число Фибоначчи = 832040 вычислено за 0.343 с.
Цикл: 30-е число Фибоначчи = 832040 вычислено за 0 с.
Рекурсия: 40-е число Фибоначчи = 102334155 вычислено за 43 с.

Цикл: 40-е число Фибоначчи = 102334155 вычислено за 0 с.

Использование функций пользователя в Excel

Вызов функции, определённой пользователем в модуле проекта на VBA, в ячейке таблицы осуществляется по имени так же, как и любой другой встроенной функции Excel. В ячейке отобразится возвращаемое значение функции. Использование процедур в ячейках таблицы в Excel не предусмотрено.

Значение функции будет автоматически пересчитываться при каждом изменении данных листа, если после заголовка указать инструкцию *Application.Volatile.*

Описание функции отображения последней изменённой ячейки:

```
Function Ad() As String
'Перерасчёт при изменении данных
  Application.Volatile
'Адрес текущего выделения
  Ad = Selection.Address
End Function
```

Запись в ячейке =Ad() приведёт к отображению адреса этой ячейки. Эта функция будет отображать последнюю изменённую ячейку или диапазон.

Аргументы функции пользователя при вызове её из ячейки таблицы передаются по правилам, определённым для функций Excel. Список аргументов разделяется точками с запятой, в качестве аргументов используются константы и формулы в явном виде или в виде ссылки на ячейку или диапазон, типы значений аргументов в последовательности, образующей список, должны соответствовать типам, определённым при описании функции. Для передачи произвольного числа аргументов при описании списка аргументов функции используется ключевое слово *ParamArray,* как это было показано выше на примере функции *СУММПОЛ.*

5.3. Обработка двумерных массивов.

Задачи обработки двумерных массивов в практике программирования встречаются сравнительно часто. Это связано с тем, что во многих случаях информацию для обработки целесообразно предста-

вить в виде таблицы. При решении вычислительных задач двумерный массив представляет матрицу, одномерный массив – вектор. Обрабатываются массивы поэлементно. Элемент двумерного массива задаётся двумя индексами, интерпретируемыми как номер строки и номер столбца. Чтобы обработать все элементы двумерного массива, в цикле перебирают номера всех строк, а для каждого номера строки в цикле, перебирают номера всех элементов строк, то есть номера столбцов. Конструкцию из этой пары циклов часто называют «цикл в цикле» или вложенные циклы (Рисунок .**44**).

Транспонирование матрицы.

Необходимость транспонирования матрицы возникает во многих вычислительных задачах, связанных с матричной арифметикой (оптимизация, поиск псевдорешения и т.д.), при преобразовании представления двумерных таблиц в статистике для соответствия формату обработки готовыми программными средствами или удобства отображения и во многих других ситуациях.

Постановка задачи

Исходными данными являются матрица и её размеры. Результатом должна быть транспонированная матрица (Рисунок .**41**).

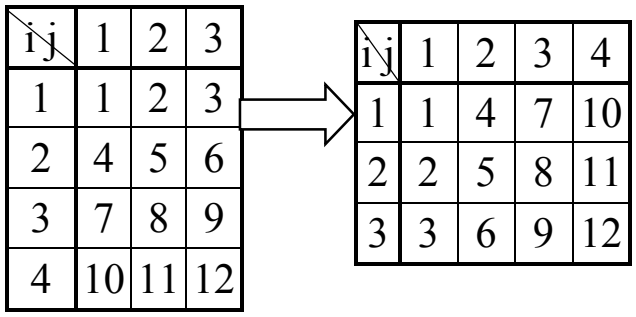


Рисунок .41. Транспонирование матрицы.

Исходная матрица считывается из левого верхнего угла текущего листа MS Excel. Результат выводится на тот же лист на строку ниже конца исходной матрицы. Формат документа MS Excel 2003 накладывает ограничения на количество столбцов: от 1 до 255.

Информационная структура задачи

Входные данные:
m – число строк матрицы – целое число от 1 до 255;
n – число столбцов матрицы – целое число от 1 до 255;
A – двумерный массив с матрицей – 255x255 произвольного типа.

Выходные данные:
B – двумерный массив с транспонированной матрицей – 255x255 произвольного типа.

Разработка алгоритма программы транспонирования матрицы

Для получения результата программа должна выполнять следующую последовательность действий: Ввод размеров матрицы, ввод матрицы, транспонирование матрицы, вывод данных. В соответствии с принципом нисходящего структурного программирования каждое из этих действий описывается в виде подпрограммы.

Первая и вторая процедуры обеспечивают подсказку и корректный ввод пользователем размеров матрицы. Подсказка и ограничения задаются входными параметрами. Возвращаемое значение – число в заданных границах. Так как процедура возвращает одно значение, которое удобно записывать в правой части оператора присваивания, её целесообразно описать в формате функции. В процедуре ввода массива: входные параметры задают размеры массива и область ввода, возвращаемые данные – массив исходных значений. В процедуре транспонирования: входные параметры – исходный массив и его размеры, возвращаемые данные – транспонированный массив. В процедуре вывода массива: входные параметры – транспонированный массив, его размеры и координаты начала вывода.

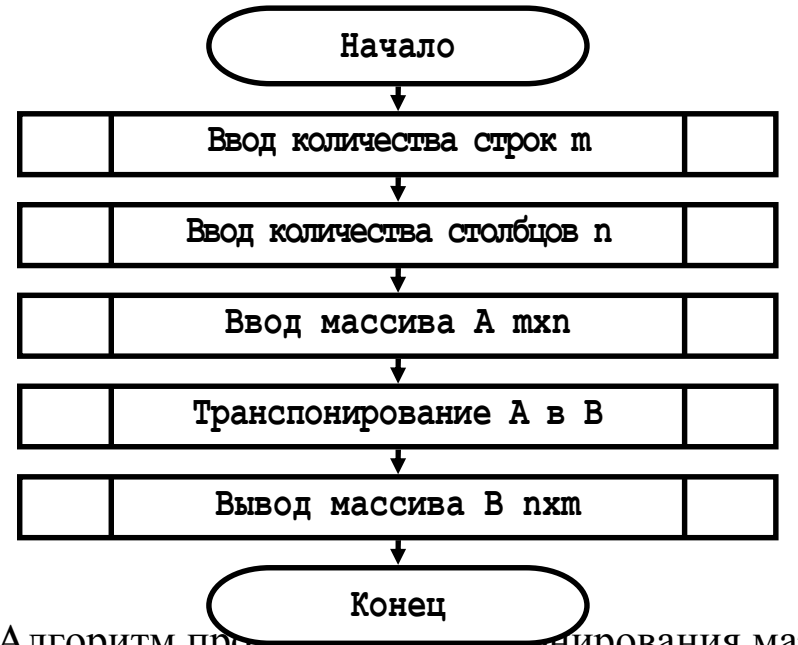


Рисунок .42. Алгоритм программы транспонирования матрицы.

Разработка текста программы транспонирования матрицы

```
Sub Trans()
Dim A(1 To 255,1 To 255),B(1 To 255,1 To 255), _
m As Long,n As Long
  m=inputvalue("Введите число строк",1,255)
  n=inputvalue("Введите число столбцов",1,255)
'Ввод массива A начиная с ячейки 1,1
  InputMas A,m,n,1,1
'Транспонирование массива A в B
  Transpose A,B,m,n
'Вывод массива B nxm с ячейки m+2,1
  OutputMas B,n,m,m+2,1
End Sub
```

Процедура ввода целочисленного значения пользователя из заданного диапазона с выдачей заданной подсказки

Ввод осуществляется через диалоговое окно и повторяется, пока введённое число не окажется в пределах заданного диапазона.

Информационная структура процедуры:

Входные данные:

p – подсказка – строка;
a – нижняя граница диапазона возможных значений – целое число.
b – верхняя граница диапазона возможных значений – целое число.

Локальные данные:

v – введённое пользователем значение, подлежащее проверке на соответствие заданному диапазону – целое число.

Выходные данные

InputValue – прошедшее проверку значение v – целое число.

Алгоритм

Так как значение должно быть введено хотя бы один раз, и ввод должен повторяться до тех пор, пока значение не окажется в пределах заданного диапазона, применяется цикл с постусловием.

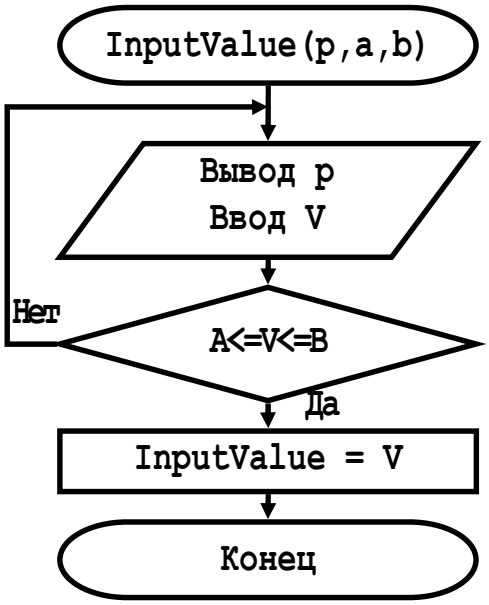


Рисунок .43. Схема алгоритма ввода значения пользователя из заданного диапазона

Исходный текст

```
Function InputValue(P As String,A As Long, _
B As Long)As Long
  Dim V As Long
  Do
  'Ввод с преобразованием в тип Long
    V=CLng(InputBox(p))
'Пока значение не попадёт в [A;B]
```

```
  Loop Until A<=V And V<=B
  InputValue = V 'Возвращаемое значение
End Function
```

Процедура ввода массива заданного размера из заданной позиции текущего листа электронной таблицы

Элемент двумерного массива задан номерами строки и столбца.

Информационная структура

Входные данные

m – число строк матрицы – целое число >0;
n – число столбцов матрицы – целое число >0;
r – начальная строка ввода на листе Excel – целое число >0;
c – начальный столбец ввода на листе Excel – целое число >0;

Локальные данные

i – номер текущей строки массива – целое число от 1 до m;
j – номер текущего столбца массива – целое число от 1 до n;

Выходные данные

A – двумерный массив с матрицей – mxn произвольного типа.

Алгоритм

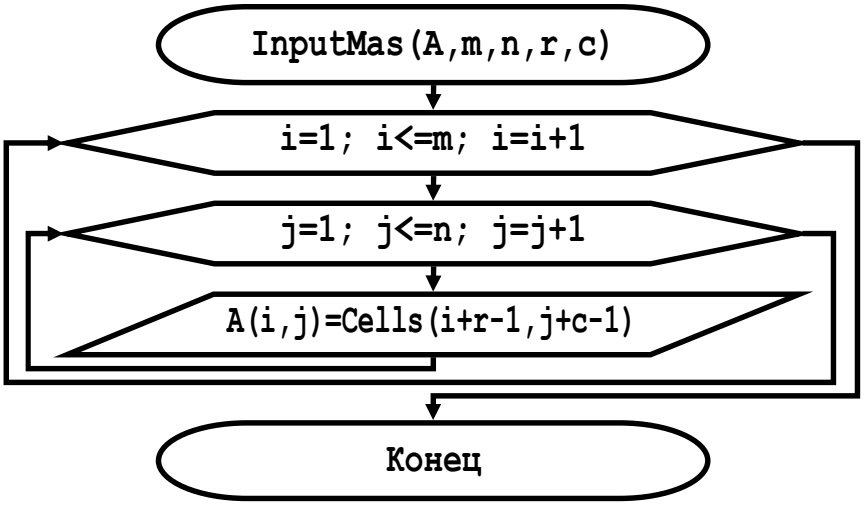


Рисунок .44. Схема алгоритма процедуры ввода массива

Исходный текст

```
Sub InputMas(A(), m As Long, n As Long, _
r As Long, c As Long)
  Dim i As Long, j As Long
```

```
  For i = 1 To m 'строки i от 1 до m
    For j = 1 To n 'столбцы j от 1 до n
    'Ввод данных ячейки в массив
      A(i,j)=Cells(i+r-1,j+c-1)
    Next j 'конец цикла по j
  Next i 'конец цикла по i
End Sub
```

Процедура транспонирования массива заданного размера

Информационная структура

Входные данные

m – число строк матрицы – целое число >0;

n – число столбцов матрицы – целое число >0;

A – двумерный массив с матрицей – mxn произвольного типа.

Локальные данные

i – номер текущей строки массива – целое число от 1 до m;

j – номер текущего столбца массива – целое число от 1 до n;

Выходные данные

B – двумерный массив с матрицей – nxm произвольного типа.

Алгоритм

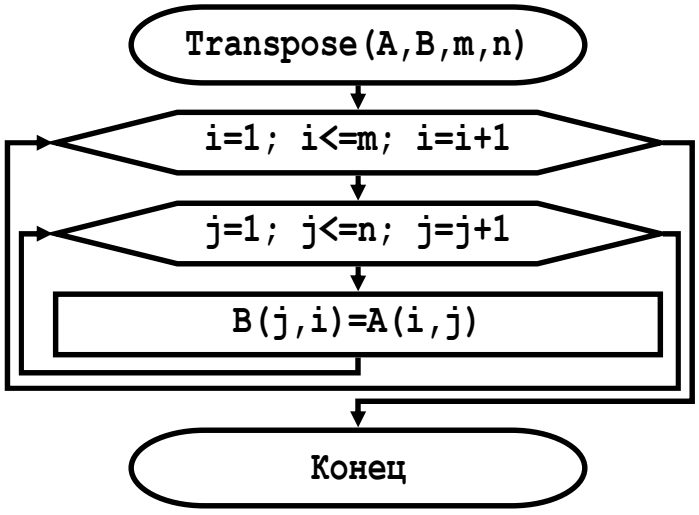


Рисунок .45. Схема алгоритма транспонирования матрицы

Исходный текст

```
Sub Transpose(A(),B(),m As Long, n As Long)
Dim i As Long, j As Long
```

```
  For i = 1 To m 'Строки массива A
    For j = 1 To n 'Столбцы массива A
      B(j, i) = A(i, j) 'транспонирование
    Next j 'Следующий столбец
  Next i 'Следующая строка
End Sub
```

Процедура вывода массива заданного размера в заданную позицию листа электронной таблицы.

Информационная структура

Входные данные

m – число строк матрицы – целое число >0;
n – число столбцов матрицы – целое число >0;
r – начальная строка вывода на листе Excel – целое число >0;
c – начальный столбец вывода на листе Excel – целое число >0;
A – двумерный массив с матрицей – mxn произвольного типа.

Локальные данные

i – номер текущей строки массива – целое число от 1 до m;
j – номер текущего столбца массива – целое число от 1 до n;

Выходные данные

отсутствуют.

Алгоритм

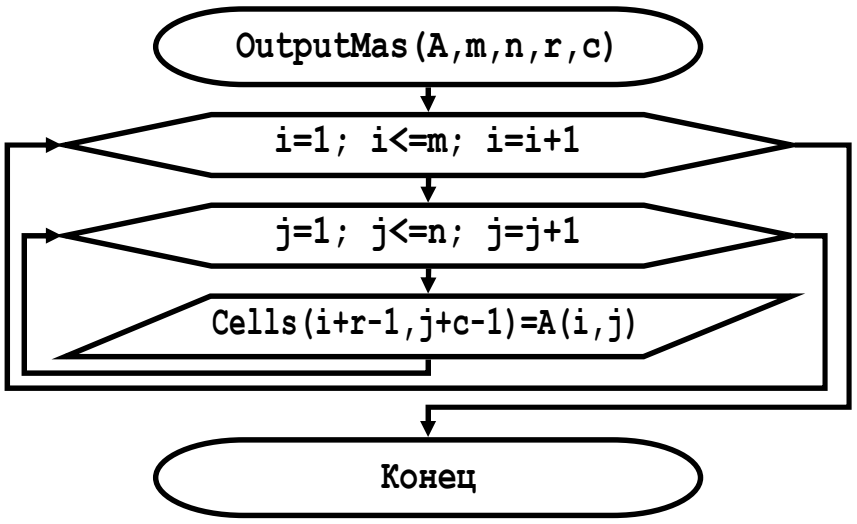


Рисунок .46. Алгоритм процедуры вывода двумерного массива

Исходный текст

```
Sub OutputMas(A(), m As Long, n As Long, _
r As Long, c As Long)
  Dim i As Long, j As Long
  For i = 1 To m 'строкам i от 1 до m
    For j = 1 To n 'столбцы j от 1 до n
    'Вывод на текущий лист
      Cells(i+r-1,j+c-1)=A(i,j)
    Next j 'конец цикла по j
  Next i 'конец цикла по i
End Sub
```

Работа с диагоналями

Обработка диагоналей матрицы и областей, ограниченных ими, встречается на практике во многих численных алгоритмах, связанных с матричной арифметикой (обращение матриц, решение систем уравнений и т.д.), в задачах статистической обработки данных, и ряде других случаев. Наличие диагоналей подразумевает, что матрица квадратная. Диагональ из левого верхнего в правый нижний угол матрицы принято называть главной, другую диагональ – побочной.

Постановка задачи

Задание: вычислить суммы элементов над главной, над побочной, под главной, под побочной диагоналями и на диагоналях.

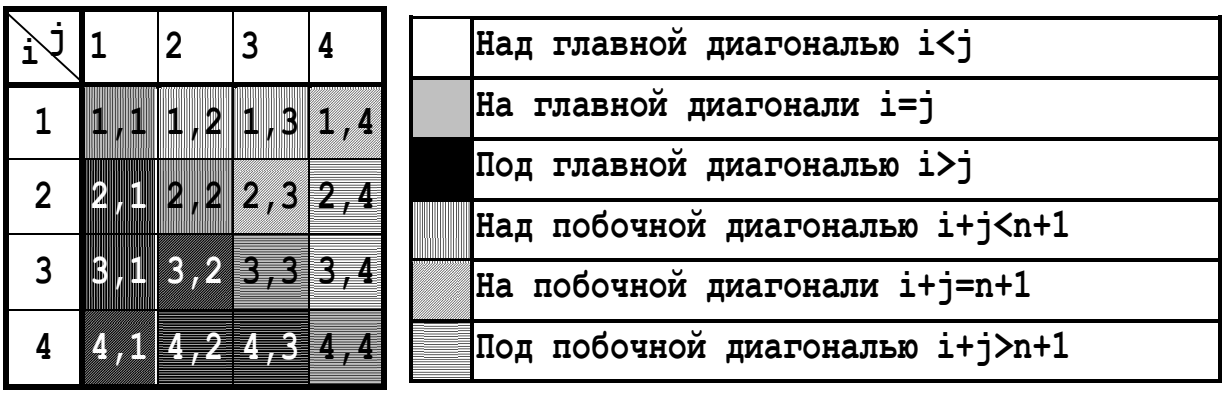


Рисунок.47. Диагонали квадратной матрицы

Исходная матрица считывается из левого верхнего угла текущего листа MS Excel. Результат выводится на тот же лист на строку ни-

же конца исходной матрицы. MS Excel 2003 накладывает ограничения на количество столбцов: от 1 до 255.

Информационная структура задачи

Входные данные:

n – число строк и столбцов матрицы – целое число от 1 до 255;

A – двумерный массив с матрицей – 255x255 чисел.

Выходные данные:

S – одномерный массив сумм – 6 чисел.

Разработка алгоритма программы

Для получения результата программа должна выполнять следующую последовательность действий: Ввод размера матрицы, ввод матрицы, вычисление сумм, вывод данных. В соответствии с принципом нисходящего структурного программирования представим эти действия на схеме алгоритма как последовательность подпрограмм.

Первая процедура обеспечивает подсказку и корректный ввод пользователем размера матрицы. Подсказка и ограничения задаются входными параметрами. Возвращаемое значение – число в заданных границах. В процедуре ввода массива: входные параметры задают размер массива и область ввода, возвращаемые данные – массив исходных значений. В процедуре вычисления сумм: входные параметры – исходный массив и его размер, возвращаемые данные – одномерный массив с результатами. В процедуре вывода массива: входные параметры – одномерный массив, его размеры и координаты начала вывода. Массив выводится в строку.

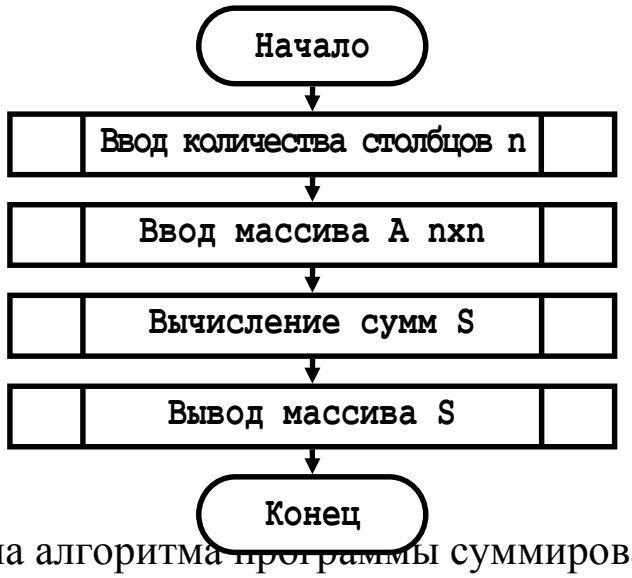


Рисунок .48. Схема алгоритма программы суммирования.

Разработка исходного текста программы

Для ввода количества столбцов используется функция *Inputvalue* из предыдущего примера. Для ввода массива используется процедура *InputMas* из предыдущего примера, в качестве фактических параметров указывается одинаковое количество строк и столбцов.

```
Sub DiagS()
  Dim A(1 To 255, 1 To 255), S(1 To 6), n As Long
  n = inputvalue("Введите количество
столбцов",1,255)
  InputMas A,n,n,1,1  'Ввод массива A nxn
  SumDiag A,S,n           'Вычисление сумм S
  OutputRow S,6,n+2,1 'Вывод массива S
End Sub
```

Процедура вычисления сумм

Информационная структура

Входные данные

n – число столбцов матрицы – целое число >0;

A – двумерный массив с матрицей – числового типа.

Локальные данные

i – номер текущей строки массива – целое число от 1 до n;

j – номер текущего столбца массива –целое число от 1 до n;

Выходные данные

S – одномерный массив с суммами – 6 элементов числового типа.

Алгоритм

При разработке алгоритма учитывается, что каждая из диагоналей определяет три непересекающиеся области матрицы, при этом области, определяемые разными диагоналями, пересекаются.

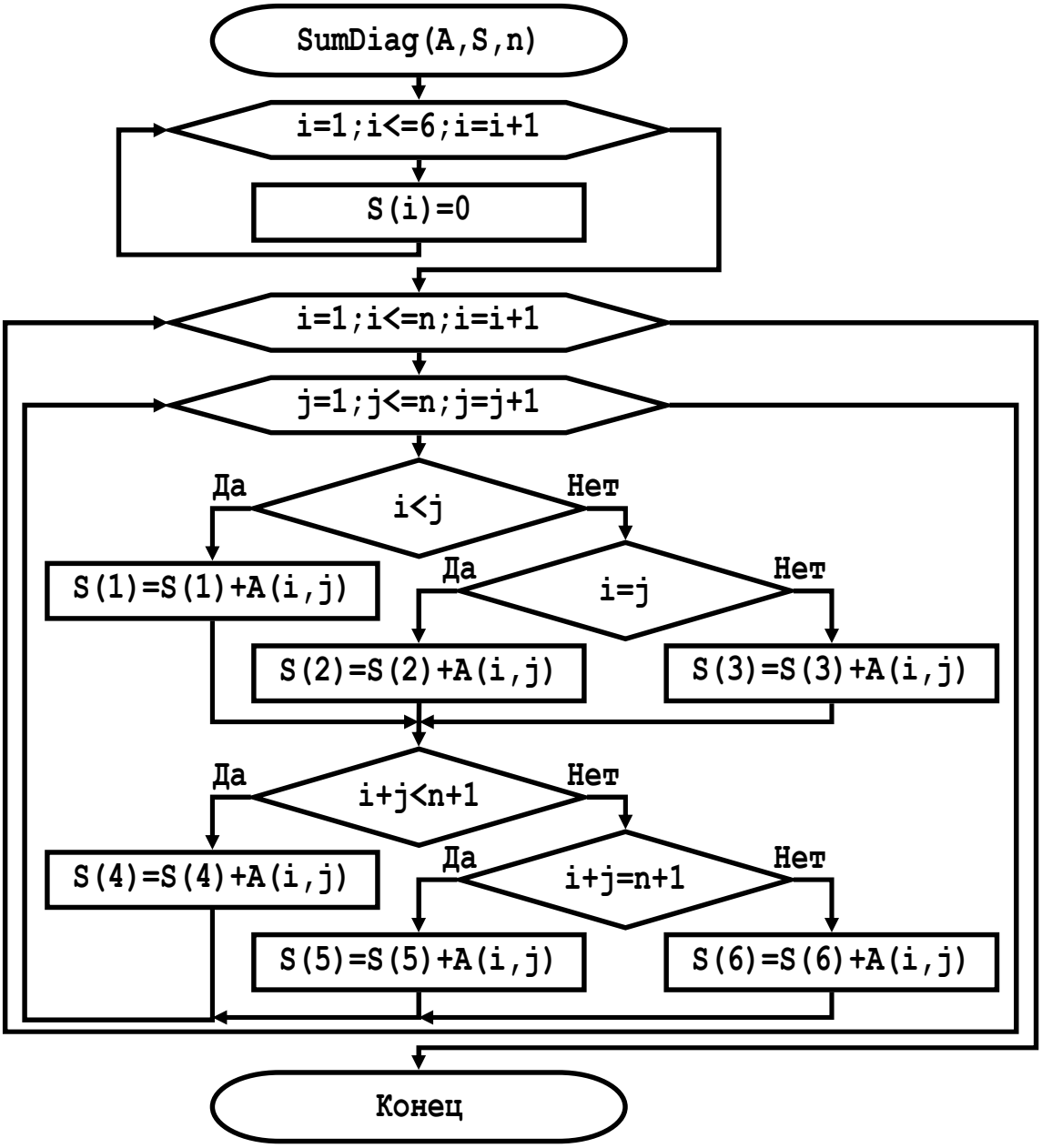


Рисунок .49 Схема алгоритма процедуры суммирования элементов определённых диагоналями областей матрицы

Исходный текст:

```
Sub SumDiag(A(), S(), n As Long)
  Dim i As Long, j As Long
  For i = 1 To 6 'Инициализация массива сумм
    s(i) = 0
  Next i
  For i = 1 To n 'Строки
    For j = 1 To n 'Столбцы
      If (i < j) Then 'Над главной диагональю
        s(1)=s(1)+A(i,j) 'сумма 1
      ElseIf (i=j) Then 'На главной
        s(2)=s(2)+A(i,j) 'сумма 2
      Else 'Иначе – под главной
        s(3)=s(3)+A(i,j) 'сумма 3
    'Позиции относительно главной рассмотрены
      End If
      If (i+j)<(n+1) Then 'Над побочной
       s(4)=s(4)+A(i,j) 'сумма 4
      ElseIf (i+j)=(n+1) Then 'На побочной
        s(5)=s(5)+A(i,j) 'сумма 5
      Else 'Иначе – под побочной
        s(6)=s(6)+A(i,j) 'сумма 6
    'Позиции относительно побочной рассмотрены
      End If
    Next j 'Следующий элемент строки
  Next i 'Следующая строка
End Sub
```

Процедура вывода строки

Информационная структура

Входные данные

n – длина строки – целое число >0;

A – одномерный массив – произвольного типа.

r – строка вывода на листе Excel – целое число >0;

c – начальный столбец вывода на листе Excel – целое число >0;

Локальные данные

i – номер текущей строки массива – параметр цикла – целое число от 1 до n;

Выходные данные
отсутствуют.

Алгоритм

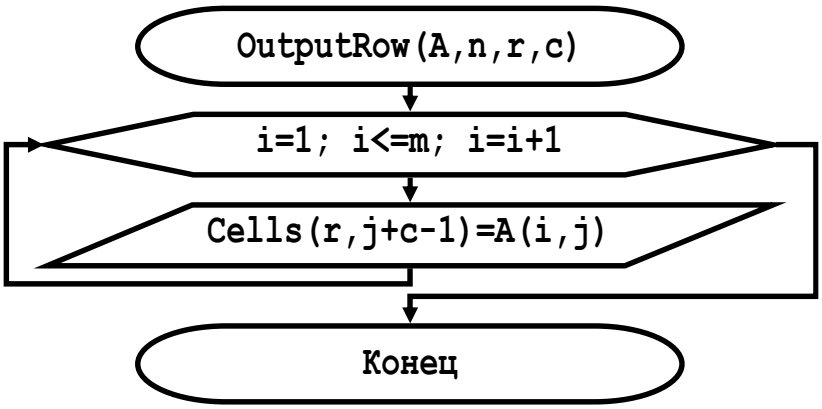


Рисунок .50. Схема алгоритма процедуры вывода строки.

Исходный текст

```
Sub OutputRow(A(), n As Long, r As Long, _
c As Long)
  Dim i As Long
  For i = 1 To n 'элементы от 1 до n
  'Вывод элемента массива на текущий лист
    Cells(r,i+c-1)=A(i)
  Next i 'конец цикла по i
End Sub
```

Сортировка

Задачи сортировки данных применяются во многих численных алгоритмах (для минимизации погрешностей, выигрыша в скорости, защиты от ошибок переполнения и деления на ноль и т.д.), в статистической обработке, для удобства представления данных и во многих других приложениях [7].

Постановка задачи

Задание: Отсортировать матрицу по возрастанию минимальных значений строк (Рисунок.**51**).

Исходная матрица считывается из левого верхнего угла текущего листа MS Excel. Результат выводится на тот же лист на строку ниже конца исходной матрицы. MS Excel 2003 накладывает ограничения на количество строк и столбцов: от 1 до 255.

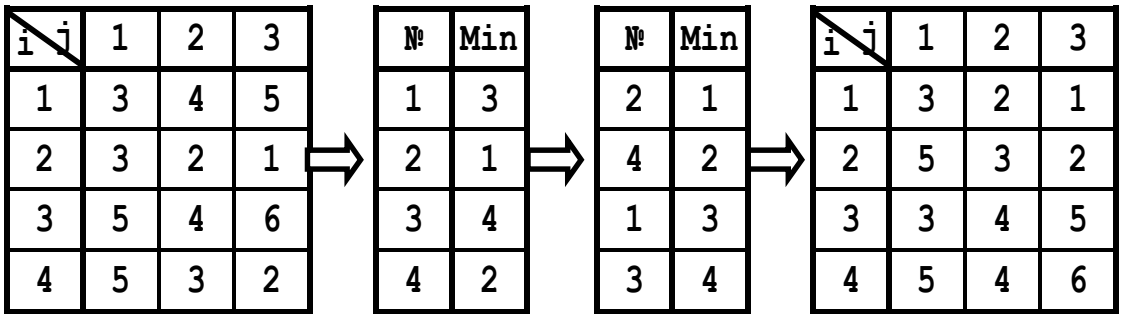


Рисунок.51. Сортировка матрицы: исходная матрица, массив минимальных значений строк, порядок строк, отсортированных по возрастанию минимальных значений, матрица, отсортированная по возрастанию минимальных значений строк.

Информационная структура задачи

В соответствии со схемой (Рисунок.**51**), в процессе обработки вводится промежуточный массив минимальных значений строк и массив номеров строк, по которому происходит сортировка матрицы.

Входные данные:

m – число строк матрицы – целое число от 1 до 255;

n – число столбцов матрицы – целое число от 1 до 255;

A – двумерный массив с матрицей – 255x255 числового типа.

Промежуточные данные:

Mins – одномерный массив минимумов строк – 255 чисел.

Nums – одномерный массив номеров строк – 255 целых чисел.

Выходные данные:

B – двумерный массив с упорядоченной матрицей – 255x255 чисел.

Разработка алгоритма программы сортировки матрицы

Для получения результата программа должна выполнять следующую последовательность действий: Ввод размеров матрицы, ввод матрицы, определение минимальных значений строк, сортировка номеров строк в порядке возрастания минимальных значений, сортировка матрицы, вывод данных. В соответствии с принципом нисхо-

дящего структурного программирования представим эти действия на схеме алгоритма в виде последовательности подпрограмм.

Первая и вторая процедуры обеспечивают подсказку и корректный ввод пользователем размеров матрицы. Подсказка и ограничения задаются входными параметрами. Возвращаемое значение – число в заданных границах. В процедуре ввода массива: входные параметры задают размеры массива и область ввода, возвращаемые данные – массив исходных значений. В процедуре определения минимальных значений строк: входные параметры – исходный массив и его размеры, возвращаемые данные – массив минимальных значений строк. В процедуре сортировки номеров строк: входные данные – массив минимальных значений строк и его размеры, выходные - массив номеров строк, отсортированный в порядке возрастания минимальных значений. В процедуре сортировки матрицы: входные параметры – исходный массив, его размеры, отсортированный массив номеров строк, возвращаемые данные – отсортированный массив. В процедуре вывода массива: входные параметры – отсортированный массив, его размеры и координаты начала вывода.

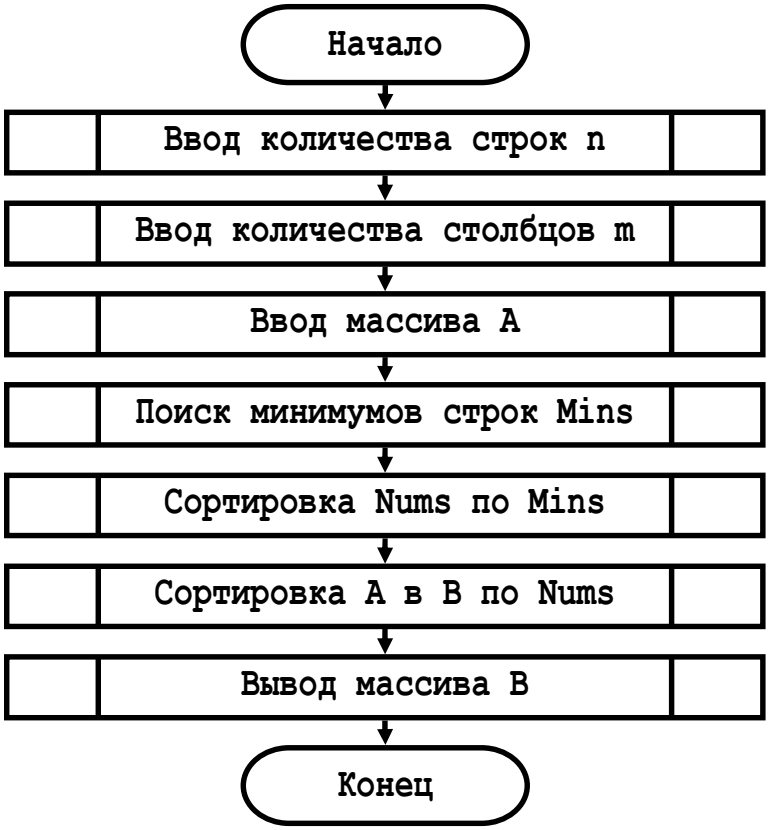


Рисунок .52. Схема алгоритма сортировки матрицы по возрастанию минимальных значений строк

Разработка исходного текста программы

```
Sub SortM ()
Dim A(1 To 255, 1 To 255), B(1 To 255, 1 To 255)
Dim Mins(1 To 255)
Dim m As Long, n As Long, Nums(1 To 255) As Long
  m=inputvalue("Введите число строк",1,255)
  n=inputvalue("Введите число столбцов",1,255)
'Ввод массива A начиная с ячейки 1,1
  InputMas A,m,n,1,1
'Поиск минимальных элементов строк
  FindMins A,m,n,Mins
'Сортировка номеров по Mins
  SortByMins m,Mins,Nums
'Сортировка массива A в B по Nums
  SortByNums A,m,n,Nums,B
'Вывод массива B mxn с ячейки m+2,1
  OutputMas B,m,n,m+2,1
End Sub
```

Все процедуры, кроме *FindMins*, *SortByMins* и *SortByNums* созданы выше при разработке программ транспонирования и обработки диагоналей.

Процедура поиска минимальных значений строк

Информационная структура

Входные данные

m – число строк матрицы – целое число >0;
n – число столбцов матрицы – целое число >0;
A – двумерный массив с матрицей – числового типа.

Локальные данные

i – номер текущей строки массива – целое число от 1 до m;
j – номер текущего столбца массива – целое число от 1 до n;

Выходные данные

Mins – одномерный массив минимальных значений – числового типа.

Алгоритм

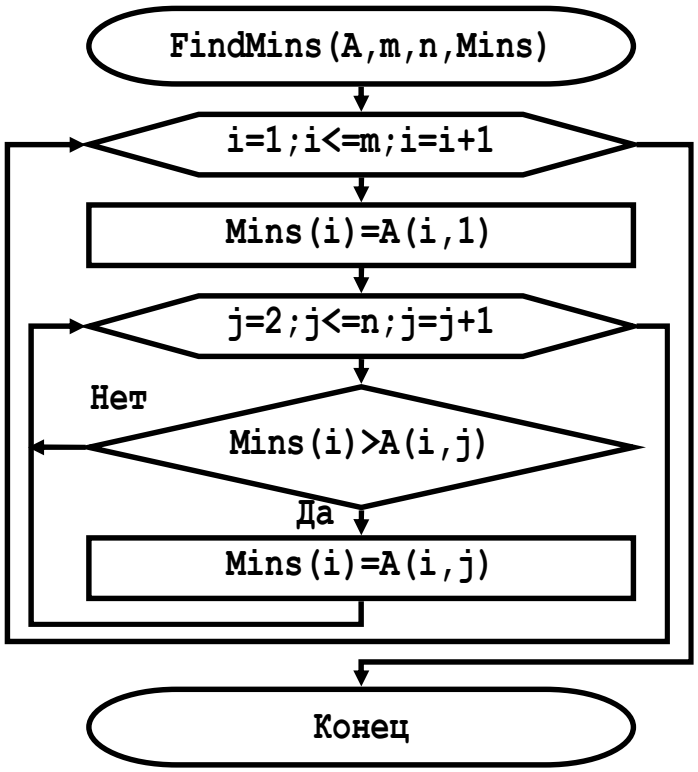


Рисунок .53. Схема алгоритма поиска минимумов строк

Исходный текст

```
Sub FindMins(A(),m As Long, n As Long, Mins())
  Dim i As Long, j As Long
  For i=1 To m 'Для всех строк
  'Первый элемент считаем минимальным
    Mins(i)=A(i,1)
  'Для всех остальных элементов строки
    For j=2 To n
    'Если встретился меньше минимального -
    'запоминаем новый
      If Mins(i)>A(i,j) Then Mins(i)=A(i,j)
    Next j 'Следующий элемент строки
  Next i 'Следующая строка
End Sub
```

Процедура сортировки номеров строк

Информационная структура

Входные данные

m – число строк матрицы – целое число >0;

Mins – одномерный массив минимумов строк –числового типа.

Локальные данные

i – номер текущей строки массива – целое число от 1 до m;

j – позиция пузырька при сортировке - целое число от 1 до m;

x – промежуточное минимальное значения при сортировке – число.

y – промежуточный номер строки при сортировке – целое число.

Выходные данные

Nums – упорядоченный одномерный массив номеров строк – целые числа.

Mins – упорядоченный одномерный массив минимумов – числа.

Алгоритм

Перед сортировкой массив Nums должен быть заполнен целыми числами от 1 до m, обозначающими начальный порядок строк исходной матрицы. Сортировка выполняется методом пузырька.

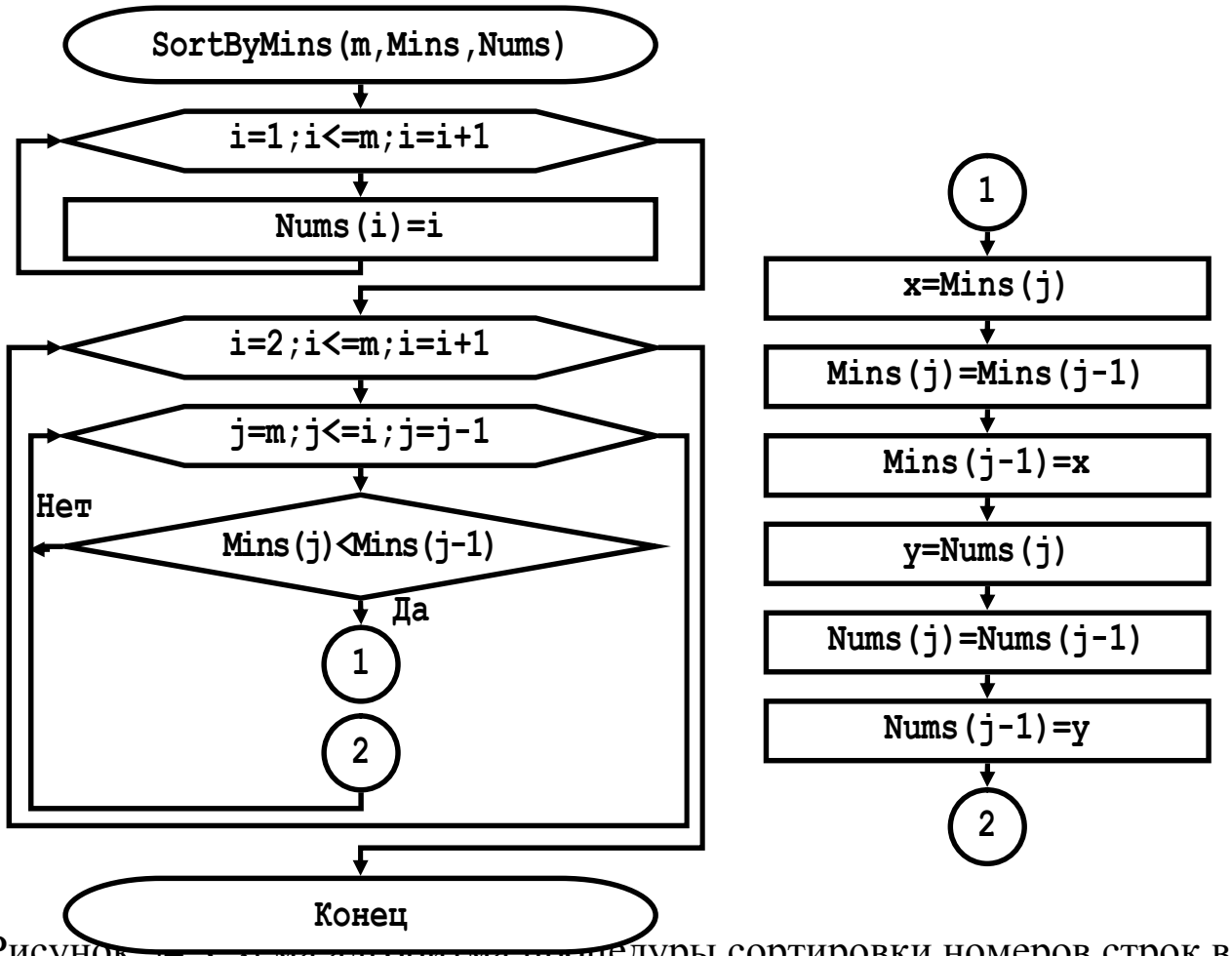


Рисунок.54. Схема алгоритма процедуры сортировки номеров строк в порядке возрастания минимальных значений.

Участок программы 1-2 включает в себя двукратное повторение однотипной последовательности действий, меняющих местами значения переменных. Реализуя принцип восходящего программирования, заменяем однотипные последовательности подпрограммами. В этом случае локальные переменные x и y не используются.

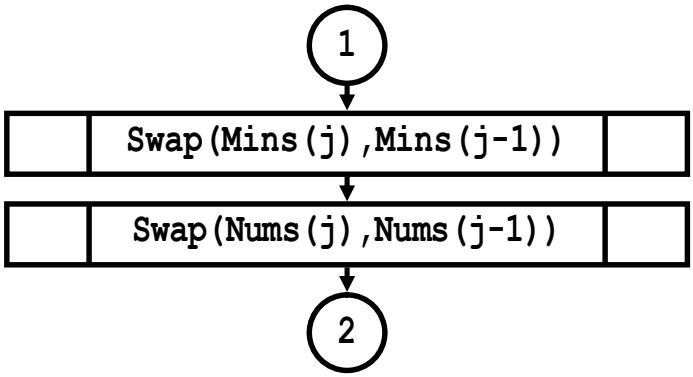


Рисунок.55. Вызов подпрограмм, меняющих значения аргументов, из процедуры сортировки номеров строк.

Исходный текст

```
Sub SortByMins(m As Long, Mins(), Nums() As Long)
  Dim i As Long, j As Long
  For i=1 To m 'Для всех строк
    Nums(i)=i 'Записываем в Nums их номера
  Next i
  For i=2 To m 'Сортировка методом «пузырька»
  'Для всех пар с последней по i-ю
    For j=m To i Step -1
    'Если пара не отсортирована
      If Mins(j)<Mins(j-1) Then
      'Меняем местами минимумы
        Swap Mins(j),Mins(j-1)
      'и номера строк, соответственно
        Swap Nums(j),Nums(j-1)
      End If
    Next j 'следующая пара
  Next i 'поиск следующего минимума
End Sub
```

Процедура замены значений переменных

Информационная структура

Входные данные

a, b – значения, которые надо поменять местами.

Локальные данные

c – промежуточное значение для замены.

Выходные данные

a, b – значения поменянные местами.

Алгоритм

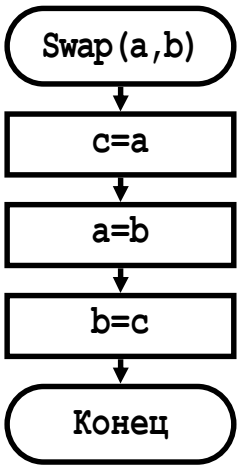


Рисунок.56. Схема алгоритма подпрограммы замены значений переменных.

Исходный текст

```
Sub Swap (a,b)
  Dim c
  c=a
  a=b
  b=c
End Sub
```

Процедура сортировки матрицы

В массив с результатом требуется записать строки исходного массива, порядок которых определяется на предыдущем этапе.

Информационная структура

Входные данные

m – число строк матрицы – целое число >0;

n – число столбцов матрицы – целое число >0;

A – двумерный массив с исходной матрицей – числового типа.

Nums – одномерный массив упорядоченных номеров строк – целые числа.

Локальные данные

i – номер текущей строки массива – целое число от 1 до m;

j – номер текущего столбца массива – целое число от 1 до n;

Выходные данные

B – двумерный массив с отсортированной матрицей – числа.

Алгоритм

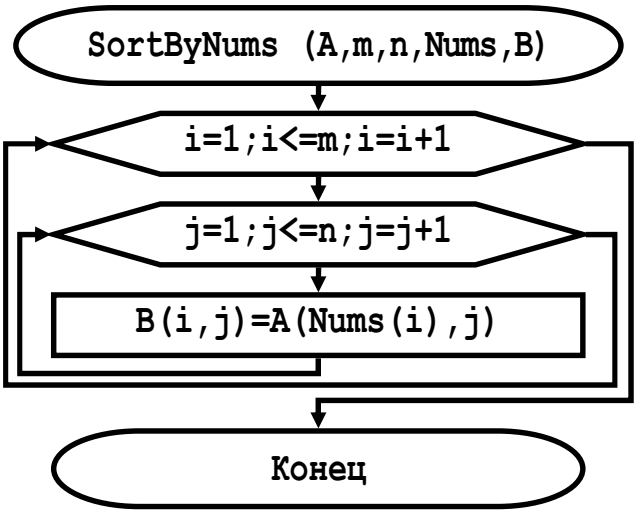


Рисунок .57. Схема алгоритма процедуры сортировки матрицы.

Исходный текст

```
Sub SortByNums(A(),m As Long, n As Long, Nums() _
As Long,B())
  Dim i As Long, j As Long
  For i=1 To m 'Для всех строк
    For j=1 To n 'Для всех элементов строки
    'Записываем строки в порядке Nums
      B(i,j)=A(Nums(i),j)
    Next j 'Следующий элемент строки
  Next i 'Следующий столбец
End Sub
```

Решение системы линейных алгебраических уравнений

Необходимость решения систем линейных уравнений (СЛАУ) возникает как в соответствующих задачах для непосредственного поиска результата, так и в составе более объёмных вычислительных задач, таких как интерполяция, решение систем дифференциальных уравнений, метод конечных элементов и многих других.

Постановка задачи

Задание: получить численное решение СЛАУ.

СЛАУ представляется в матричном виде: AX=B. Исходными данными являются количество уравнений, главная матрица A и вектор свободных членов B. Результатом должен быть вектор решений X или сообщение о неопределённости или несовместности системы. Тип результата определяется в процессе решения и кодируется числом. Исходные данные считываются с текущего листа MS Excel, где расположим их следующим образом: в левой верхней ячейке – размер, начиная со следующей строки и первого столбца – матрица A, через столбец справа от неё – столбец B. Результат выведем в виде строки снизу от матрицы A.

Информационная структура задачи

Входные данные:

n – количество уравнений – целое число от 1 до 255;

A – двумерный массив с матрицей – 255x255 вещественных чисел.

B – одномерный массив с вектором свободных членов – 255 вещественных чисел.

Промежуточные данные:

ECode – код ошибки для хранения информации о том, что система имеет решение, несовместна или неопределённа – целое число.

Выходные данные:

X – одномерный массив с вектором решений – 255 вещественных чисел.

Разработка алгоритма программы решения СЛАУ

Для получения результата программа должна выполнять следующую последовательность действий: Ввод размеров матрицы, ввод матрицы A, ввод вектора свободных членов B, решение системы, вывод результата: X или сообщения в зависимости от значения ECode. Матрица должна быть квадратной.

Для решения используем метод Гаусса – универсальный, точный, хорошо исследованный и простой для понимания. СЛАУ имеет единственное решение, если ранг основной матрицы совпадает с рангом расширенной, иначе система является неопределённой и имеет

бесконечное множество решений или несовместной и не имеет решений в действительных числах. В процессе решения определяется либо решение системы, либо причина того, почему решение не может быть получено. В соответствии с принципом нисходящего структурного программирования представим эти действия на схеме алгоритма в виде последовательности подпрограмм.

В процедуре ввода главной матрицы: входные параметры задают размеры матрицы и область ввода, возвращаемые данные – главная матрица вещественного типа двойной точности.

В процедуре ввода вектора свободных членов: входные параметры задают размеры вектора и область ввода, возвращаемые данные – вектор свободных членов вещественного типа двойной точности.

В процедуре решения СЛАУ: входные параметры – главная матрица, вектор свободных членов и их размер, возвращаемые данные – вектор решения и код ошибки. Определим следующие коды ошибок: 0 – нет ошибок, 1 – система неопределённа, 2 – система несовместна. Если код ошибки не равен нулю, вектор решения не определён.

В процедуре вывода вектора решения входными параметрами являются вектор решения, размеры вектора и координаты области вывода.

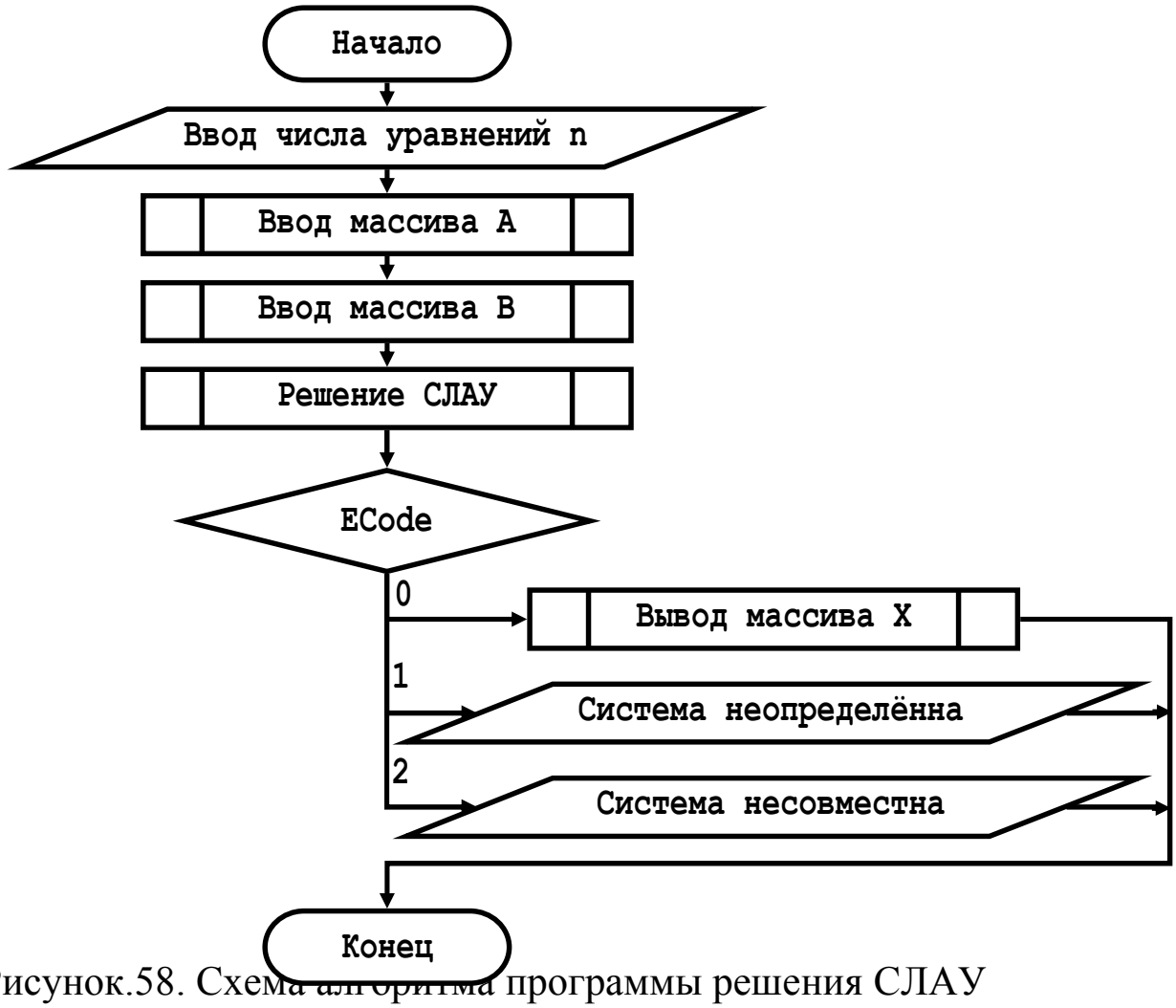


Рисунок.58. Схема алгоритма программы решения СЛАУ

Разработка исходного текста программы

```
Sub SolveSystem()
  Dim A(1 To 255, 1 To 255) As Double
  Dim B(1 To 255) As Double X(1 To 255) As Double
  Dim n As Long, ECode As Byte
'Ввод количества уравнений
  n = CLng(Cells(1, 1))
  InputDMas A,n,n,2,1 'Ввод главной матрицы
'Ввод вектора свободных членов
  InputDCol B,n,2,n+2
  GaussSolve A,B,X,n,ECode 'Решение СЛАУ
  Select Case ECode 'Анализ кода ошибки
    Case 0: 'Нет ошибки
      OutputDRow X,n,n+3,1 'Вывод вектора решения
```

```
    Case 1: 'Система неопределённа
      Cells(n + 3, 1) = "Система неопределённа"
    Case 2: 'Система несовместна
      Cells(n + 3, 1) = "Система несовместна"
  End Select
End Sub
```

Процедура ввода главной матрицы

Процедура *InputDMas* отличается от описанной ранее *InputMas* тем, что в *InputDMas* определён тип элементов массива – *Double*.

Информационная структура

Входные и локальные данные совпадают с данными *InputMas*.

Выходные данные

A – двумерный массив с матрицей – mxn типа Double.

Алгоритм

Алгоритм процедуры InputDMas совпадает с алгоритмом InputMas (Рисунок .**44**).

Исходный текст

```
Sub InputDMas(A() As Double, m As Long, _
n As Long, r As Long, c As Long)
  Dim i As Long, j As Long
  For i = 1 To m 'строки i от 1 до m
    For j = 1 To n 'столбцы j от 1 до n
    'ввод с конвертацией в Double
     A(i,j)=CDbl(Cells(i+r-1,j+c-1))
    Next j 'конец цикла по j
  Next i 'конец цикла по i
End Sub
```

Процедура ввода вектора свободных членов.

Вектор свободных членов формируется в виде столбца.

Информационная структура

Входные данные

n – количество уравнений – целое число >0;
r – начальная строка ввода на листе Excel – целое число >0;
c – начальный столбец ввода на листе Excel – целое число >0;

Локальные данные

i – номер элемента вектора – целое число от 1 до n;

Выходные данные

A – одномерный массив с вектором – n чисел типа Double.

Алгоритм

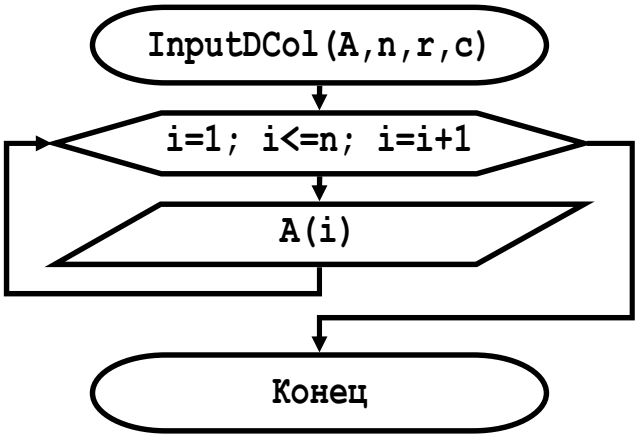


Рисунок.59. Схема алгоритма ввода вектора свободных членов

Исходный текст

```
Sub InputDCol(A() As Double, n As Long, _
r As Long, c As Long)
  Dim i As Long
  For i = 1 To n 'по строкам i от 1 до n
  'ввод с преобразованием в Double
    A(i)=CDbl(Cells(i+r-1,c))
  Next i 'конец цикла по i
End Sub
```

Процедура решения СЛАУ.

Решение СЛАУ методом Гаусса выполняется в два этапа: прямой ход, состоящий в преобразовании главной матрицы в верхнетреугольную, и обратный ход, на котором вычисляется результат.

В целях минимизации погрешностей, в прямой ход включают выбор главного элемента, состоящий в том, что среди элементов

матрицы выбирают максимальный по модулю и перемещают его на главную диагональ в первую строку перестановкой строк и столбцов. Вместе с перестановкой строк главной матрицы переставляются элементы вектора свободных членов, а при перестановке столбцов главной матрицы изменяется порядок элементов вектора решений. Для учёта изменения порядка элементов вектора решений вводится массив `XOrder`, элементы которого представляют собой исходными порядковые номера элементов вектора решений и переставляются вместе с перестановкой столбцов при выборе главного элемента в процессе прямого хода. На первом этапе также выясняется, имеет ли система единственное решение, или она является неопределённой либо несовместной. По результатам этой проверки переменная с кодом ошибки принимает одно из обозначенных выше значений.

Информационная структура

Входные данные

n – количество уравнений – целое число >0;

A – двумерный массив с матрицей – nxn типа Double;

B – одномерный массив с вектором – n чисел типа Double;

Локальные данные

i – номер элемента вектора решений – целое число от 1 до n;

XOrder – одномерный массив с исходными порядковыми номерами вектора решений – n целых чисел;

Выходные данные

X – одномерный массив с вектором решений – n чисел типа Double, определён при ECode=0;

ECode – код ошибки для хранения информации о том, что система имеет решение, несовместна или неопределённа – целое число.

Алгоритм

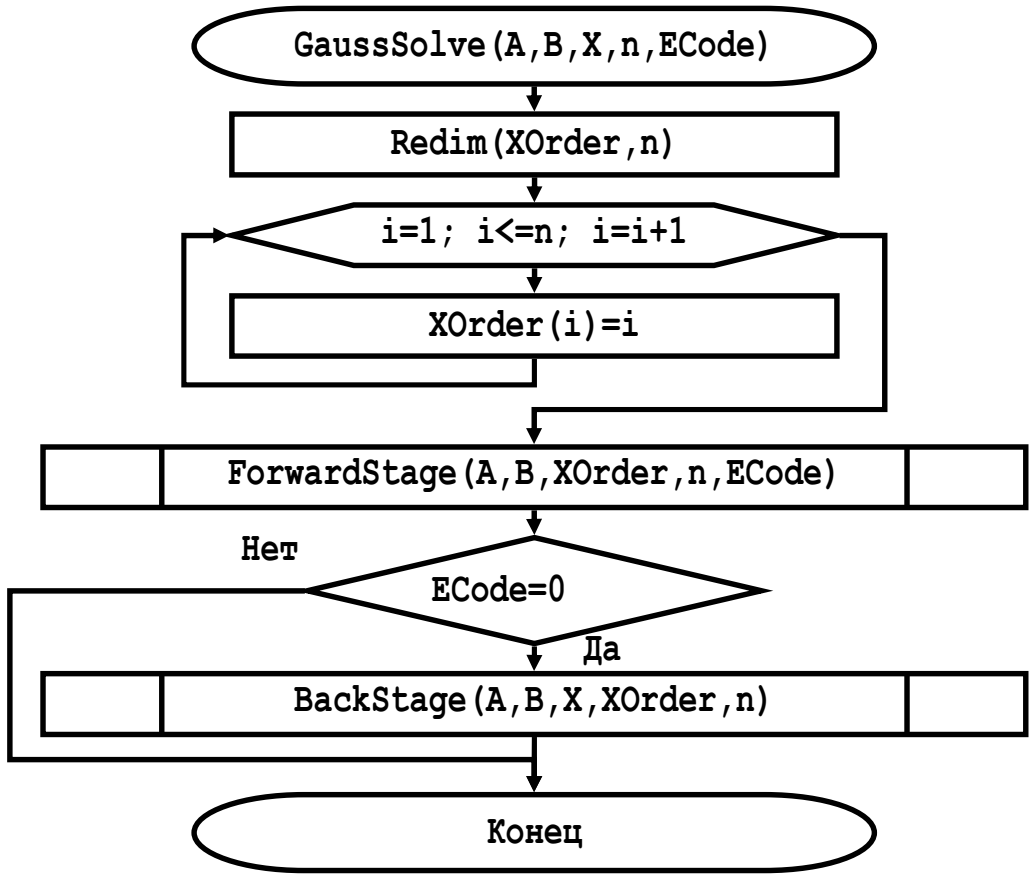


Рисунок .60. Схема алгоритма процедуры решения СЛАУ методом Гаусса

Исходный текст

```
Sub GaussSolve(A() As Double, B() As Double, _
X() As Double, n As Long, ECode As Byte)
  Dim XOrder() As Long, i As Long
  ReDim XOrder(1 To n) 'размер локального массива
  For i = 1 To n 'по строкам i от 1 до n
    XOrder(i) = i 'исходный порядок
  Next i 'следующий элемент
  ForwardStage A, B, XOrder, n, ECode 'Прямой ход
  'Обратный ход
  If ECode = 0 Then BackStage A, B, X, XOrder, n
End Sub
```

Процедура прямого хода.

В процессе прямого хода главную матрицу приводят к верхнетреугольной форме, обнуляя значения столбцов под главной диагональю. Столбцы обрабатываются последовательно, начиная с первого. Для каждого из столбцов выполняется поиск и перемещение главного элемента столбца на главную диагональ. Поиск осуществляется в части матрицы, не содержащей уже обработанных столбцов и соответствующих им строк с главными элементами. Для обнуления элементов столбца под главной диагональю, получившуюся после перестановки строку с главным элементом столбца вычитают из остальных строк, предварительно помножив на величину, равную отношению элемента обнуляемого столбца каждой из остальных строк (если он уже не нулевой) к главному элементу столбца.

Если система имеет единственное решение, в каждом столбце главный элемент не будет равен нулю. Если у одного из столбцов главный элемент получился равным нулю, то если соответствующий элемент вектора свободных членов тоже равен нулю, система неопределённа, иначе – несовместна.

Информационная структура

Входные данные

n – количество уравнений – целое число >0;

A – двумерный массив с матрицей – nxn типа Double;

B – одномерный массив с вектором – n чисел типа Double;

XOrder – одномерный массив с исходными порядковыми номерами вектора решений – n целых чисел;

Локальные данные

i – номер текущего диагонального элемента массива, под которым будет обнуляться столбец – целое;

j – номер текущей строки массива, в которой обнуляется i-й элемент – целое;

k – номер элемента строки, i-й столбец которой обнуляется – целое;

C – коэффициент, при умножении на который элемент i-го столбца j-й строки будет равен текущему диагональному - типа Double;

Выходные данные

ECode – код ошибки – целое число.

A – двумерный массив с верхнетреугольной матрицей – nxn типа Double, определён при ECode=0;
B – одномерный массив с преобразованным вектором свободных членов – n чисел типа Double, определён при ECode=0;
XOrder – одномерный массив с порядковыми номерами вектора решений – n целых чисел, определён при ECode=0;

Алгоритм

Для поиска главного элемента вызываем специальную процедуру. Если выяснится, что система не имеет единственного решения, то для выхода из цикла параметру присваиваем конечное значение.

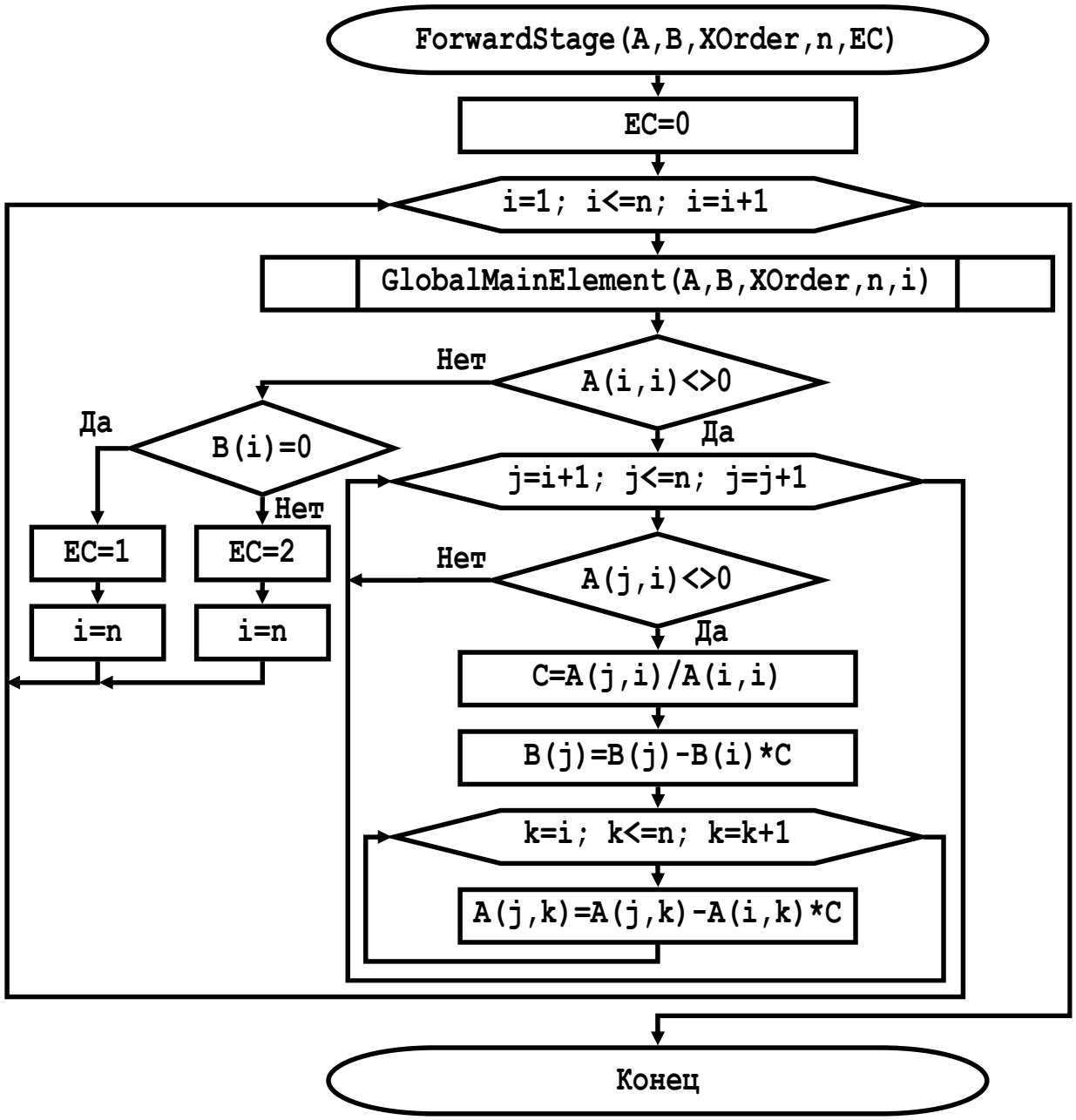


Рисунок.61. Схема алгоритма процедуры прямого хода метода Гаусса

Исходный текст

```
Sub ForwardStage(A() As Double, B() As Double, _
XOrder() As Long, n As Long, EC As Byte)
 Dim i As Long, j As Long, k As Long, C As Double
  EC = 0 'Надеемся, что без ошибок
  For i = 1 To n 'Для всех диагональных элементов
  'Поиск главного элемента
    GlobalMainElement A,B,XOrder,n,i
  'Если главный элемент не равен нулю
    If A(i,i)<>0 Then
    'Для строк ниже текущего элемента
      For j = i + 1 To n
      'Если элемент столбца уже не нулевой
        If A(j,i)<>0 Then
          C=A(j,i)/A(i,i) 'Вычисляем коэффициент
        'Вычитаем из обнуляемой строки
          B(j)=B(j)-B(i)*C
        'Для элементов обнуляемой строки
          For k=i To n
          'Вычитаем из обнуляемой строки
            A(j,k)=A(j,k)-A(i,k)*C
          Next k 'Следующий элемент
        End If
      Next j 'Следующая строка
  'Если и главный элемент и элемент вектора
  'свободных членов B нулевые
    ElseIf B(i) = 0 Then
      EC = 1 'Система неопределённа
      i = n 'Выходим
  'Если главный элемент нулевой, а элемент
  'вектора свободных членов B ненулевой
    Else
      EC = 2 'Система несовместна
      i = n 'Выходим
    End If
  Next i 'Следующий диагональный элемент
```

```
End Sub
```

Процедура поиска главного элемента.

Поиск и перемещение главного элемента состоит в поиске максимального по абсолютной величине элемента среди элементов, находящихся правее и ниже элемента главной диагонали текущего столбца, включая сам этот элемент, и перемещении его на главную диагональ путём перестановки строк и столбцов с текущим диагональным элементом и с найденным максимальным по модулю. Совместно с перестановкой строк переставляются соответствующие элементы вектора свободных членов, с перестановкой столбцов – позиции элементов вектора решения.

Информационная структура

Входные данные

n – количество уравнений – целое число >0;
A – двумерный массив с матрицей – nxn типа Double;
B – одномерный массив с вектором – n чисел типа Double;
XOrder – одномерный массив с исходными порядковыми номерами вектора решений – n целых чисел;
R – текущий диагональный элемент;

Локальные данные

i – номер текущего диагонального элемента массива, под которым будет обнуляться столбец – целое число от 1 до n;
j – номер текущей строки массива, в которой обнуляется i-й элемент – целое число от 1 до n;
k – номер элемента строки, i-й столбец которой обнуляется – целое число от 1 до n;
Max – максимальное значение в области поиска – число типа Double;
MaxX – номер строки элемента с максимальным в области поиска значением – целое число;
MaxY – номер строки элемента с максимальным в области поиска значением – целое число;

Выходные данные

A – двумерный массив после перестановки R-го главного элемента – nxn типа Double;
B – одномерный массив вектора свободных членов после перестановки R-го главного элемента – n чисел типа Double;

XOrder – одномерный массив с порядковыми номерами вектора решений после перестановки R-го главного элемента – n целых чисел;

Алгоритм

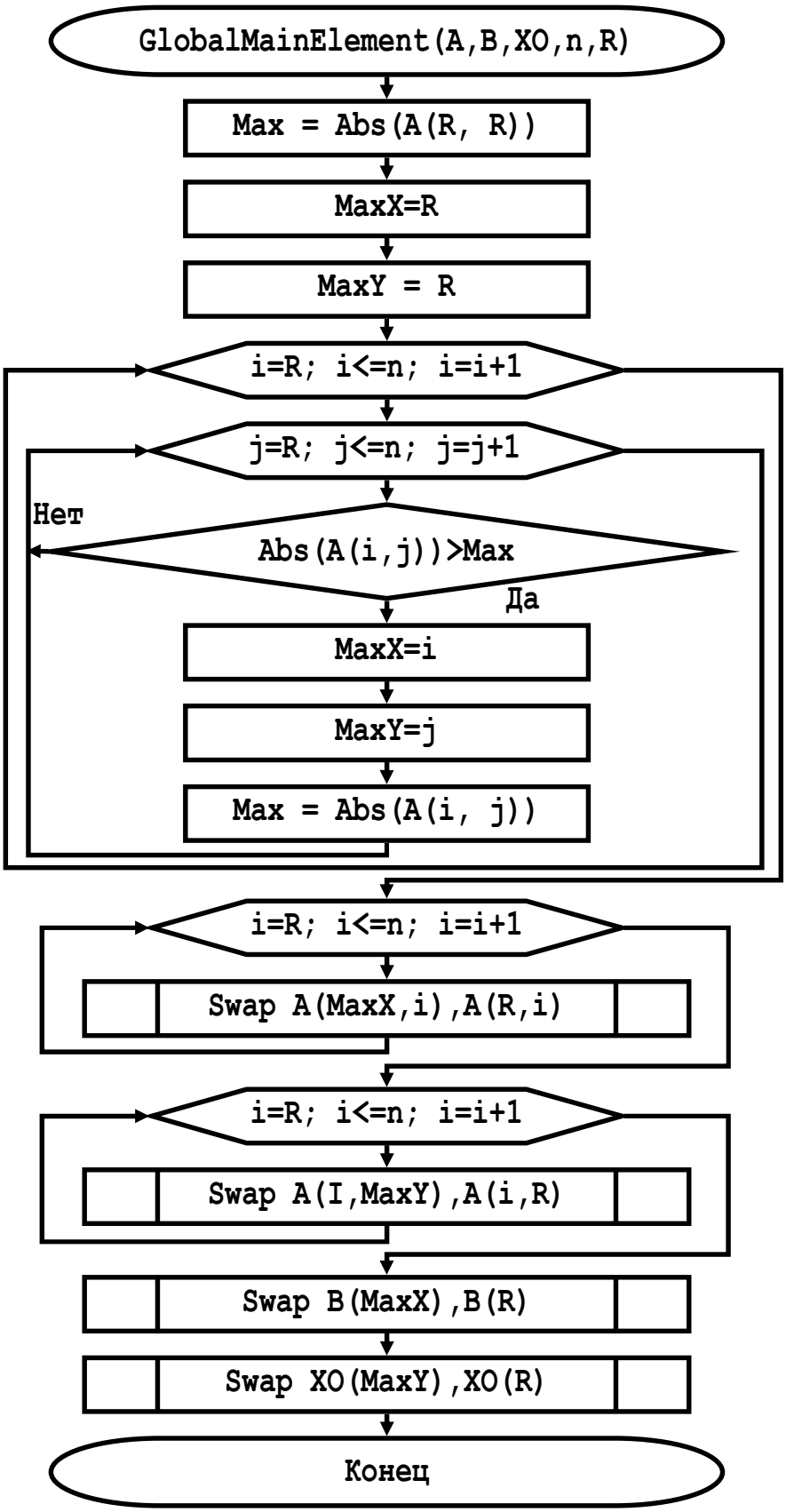

GlobalMainElement(A,B,XO,n,R)
Max = Abs(A(R, R))
MaxX=R
MaxY = R
i=R; i<=n; i=i+1
j=R; j<=n; j=j+1
Нет
Abs(A(i,j))>Max
Да
MaxX=i
MaxY=j
Max = Abs(A(i, j))
i=R; i<=n; i=i+1
Swap A(MaxX,i),A(R,i)
i=R; i<=n; i=i+1
Swap A(I,MaxY),A(i,R)
Swap B(MaxX),B(R)
Swap XO(MaxY),XO(R)
Конец

Рисунок.62. Алгоритм поиска и перемещения главного элемента.

Исходный текст

```
Sub GlobalMainElement(A() As Double, B() _
As Double, XO() As Long, n As Long, R As Long)
Dim i As Long, j As Long, k As Long, _
Max As Double, MaxX As Double, MaxY As Double
'Начальное значение для поиска
  Max = Abs(A(R, R))
  MaxX = R 'Номер строки начального значения
  MaxY = R 'Номер столбца начального значения
'Область поиска – вправо и вниз от текущего
'диагонального элемента
  For i = R To n
    For j = R To n
    'Если найдено значение больше текущего
      If Abs(A(i,j))>Max Then
       MaxX = i 'Запоминаем его позицию
       MaxY = j
       Max=Abs(A(i,j)) 'Запоминаем его значение
      End If
    Next j 'Следующий столбец
  Next i 'Следующая строка
'Меняем местами строку с главным элементом и
'строку с текущим диагональным элементом
  For i = 1 To n
    Swap A(MaxX,i),A(R,i)
  Next i
'Меняем местами столбец с главным элементом и
'столбец с текущим диагональным элементом
  For i = 1 To n
    Swap A(i,MaxY),A(i,R)
  Next i
'Меняем элементы вектора свободных членов
  Swap B(R), B(MaxX)
'Меняем порядок элементов решения
  Swap XO(R), XO(MaxY)
End Sub
```

Процедура *Swap* описана в предыдущих примерах.

Процедура обратного хода.

Если система имеет единственное решение (при этом код ошибки будет равен нулю), выполняется обратный ход. На этом этапе на основе преобразованной в верхнетреугольную главной матрицы и соответствующего вектора свободных членов вычисляются значения вектора решений системы уравнений. Так как главная матрица – верхнетреугольная, элементы последней строки за исключением элемента на главной диагонали – нулевые, последнее уравнение содержит всего одно неизвестное значение при ненулевом коэффициенте, которое можно вычислить. После этого, в предыдущем уравнении остаётся одно неизвестное при коэффициенте на главной диагонали, которое вычисляется с учётом решения последнего уравнения. Аналогично, строка за строкой снизу вверх вычисляются все неизвестные значения вектора решений. С помощью массива позиций порядок элементов вектора решений преобразуется в исходный.

Информационная структура

Входные данные

n – количество уравнений – целое число >0;

A – двумерный массив с матрицей – nxn типа Double;

B – одномерный массив с вектором – n чисел типа Double;

XOrder – одномерный массив с исходными порядковыми номерами вектора решений – n целых чисел;

Локальные данные

i – номер текущего диагонального элемента массива, под которым будет обнуляться столбец – целое;

j – номер текущей строки массива, в которой обнуляется i-й элемент – целое;

s – сумма произведений известных значений вектора решения на ненулевые коэффициенты текущей строки – число типа Double;

Выходные данные

X – одномерный массив с вектором решений – n чисел типа Double;

Алгоритм

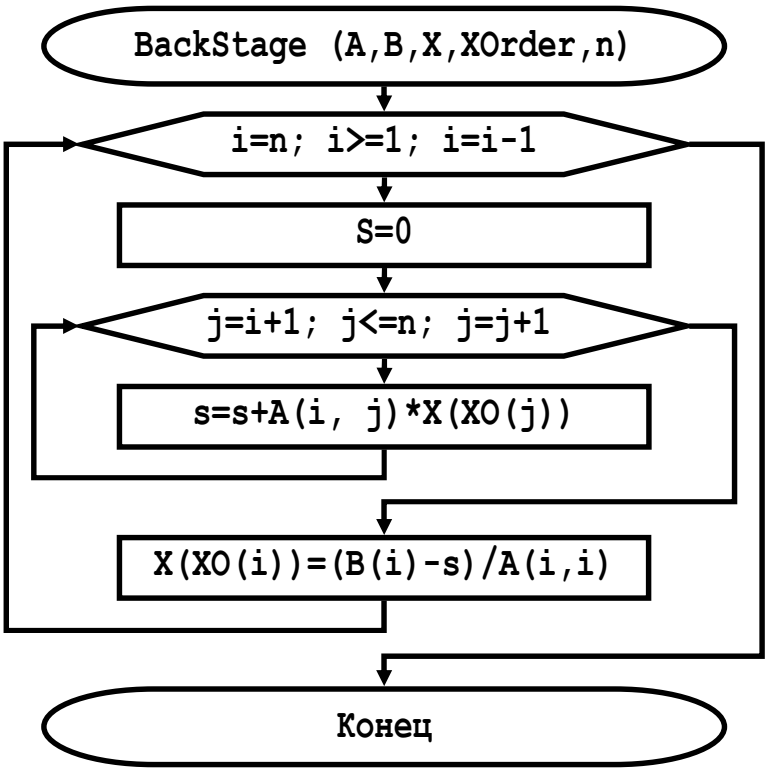


Рисунок.63. Схема алгоритма процедуры обратного хода

Исходный текст

```
Sub BackStage(A() As Double, B() As Double, _
X() As Double, XO() As Long, n As Long)
Dim i As Long, j As Long, s As Double
'Для всех строк начиная с последней
  For i = n To 1 Step -1
    s = 0 'В каждой строке будет разная сумма
  'В области правее главной диагонали
    For j = i + 1 To n
      s=s+A1(i,j)*X(XO(j)) 'Находим сумму s
  Next j 'Следующий элемент
'Вычисляем неизвестное значение
  X(XO(i))=(B1(i)-s)/A1(i,i)
Next i 'Следующая строка
End Sub
```

Процедура вывода результата

Процедура *OutputDRow* отличается от описанной ранее *OutputRow* тем, что в *OutputDRow* определён тип элементов массива – *Double*.

Информационная структура

Входные данные отличаются от входных данных процедуры *OutputRow* тем, что для входного массива A определён тип элементов *Double*.

Выходные и локальные данные совпадают с данными *OutputRow*.

Алгоритм

Алгоритм процедуры *OutputDRow* совпадает с алгоритмом *OutputRow* (Рисунок .**50**).

Исходный текст

```
Sub OutputDRow(A() As Double, n As Long, _
r As Long, c As Long)
  Dim i As Long
  For i = 1 To n 'Для всех элементов массива A
  'Выводим в нужную ячейку текущего листа
    Cells(r,i+c-1)=A(i)
  Next i 'Следующий элемент
End Sub
```

Лабораторная работа №5

Цель работы: освоение навыков структурного программирования с использованием процедур и функций. Совершенствование навыков работы с *Host* приложениями. Совершенствование навыков использования структурированных данных и базовых алгоритмических конструкций.

Задание

Используя принципы структурного программирования, разработать алгоритм и программу обработки двумерного массива с ис-

пользованием процедур и функций на VBA для решения задачи по варианту. Программу ввести в компьютер, отладить и выполнить.

Порядок выполнения работы

1. Внимательно изучить задачу по варианту. Соотнести с приведёнными примерами. Выяснить тип исходных данных и результата. Определить способ их ввода и вывода. Формализовать информационную структуру задачи. В качестве примеров можно использовать приведенные выше в порядке возрастания сложности решения задач транспонирования матрицы, обработки диагоналей матрицы, сортировки матрицы и решения систем линейных уравнений. Сложность предлагаемых вариантов задач приблизительно соответствует второму примеру и меньше, чем в третьем примере.
2. Разработать алгоритм программы с использованием подпрограмм в соответствии с принципом нисходящего программирования. Типовая последовательность подпрограмм для решения задачи включает в себя процедуры ввода исходных данных, обработки двумерного массива и вывода результата. Обратить внимание на то, чтобы каждая процедура выполняла одну функциональную задачу, входные данные к началу выполнения процедуры были сформированы, среди передаваемых параметров не содержалось неиспользуемых или инициализируемых внутри процедуры. Где целесообразно, применить функции. Не использовать подпрограммы там, где это не нужно, например, если можно обойтись одной инструкцией.
3. Определить информационные структуры и разработать алгоритмы подпрограмм в соответствии с принципом нисходящего программирования.
4. Если возможно, выделить подпрограммы, применяя принцип восходящего программирования, разработать их информационные структуры и алгоритмы.
5. Разработать исходные тексты программы и подпрограмм по алгоритмам.
6. Ввести программу в редактор VBA и добиться её правильного выполнения.
7. Ответить на контрольные вопросы.

8. Подготовить отчёт.

Содержание отчёта

1. Титульный лист.
2. Формулировка задачи.
3. Для каждой из процедур:
   - Информационная структура.
   - Схема алгоритма.
   - Текст программы с комментариями.
4. Ответы на контрольные вопросы.

Контрольные вопросы

1. В каких случаях структурное программирование нецелесообразно?
2. Как привести неструктурированный алгоритм к структурированному виду? Всегда ли это возможно?
3. Что такое и как используется восходящее и нисходящее программирование?
4. В каких случаях целесообразно использование локальных, глобальных, общих переменных? Зачем нужны статические переменные?
5. В каких случаях лучше использовать процедуры, а в каких функции?
6. В каких случаях целесообразно передавать параметры по ссылке, по значению?
7. В каких условиях целесообразно использовать рекурсию, а в каких нет?
8. Как гарантировать выход из рекурсии?
9. Как использовать процедуры и функции пользователя в хост-приложениях и функции хост-приложений в пользовательских процедурах и функциях?

Варианты задач

1. Найти сумму наибольших значений строк действительной матрицы порядка MxN.

2. Найти сумму наименьших значений столбцов действительной матрицы порядка MxN..
3. В действительной матрице порядка MxN найти строки, элементы которых упорядочены по убыванию.
4. В действительной матрице порядка MxN найти столбцы, элементы которых упорядочены по возрастанию.
5. Определить позиции несимметричных относительно главной диагонали элементов матрицы порядка NxN.
6. Определить, является ли матрица порядка NxN симметричной относительно побочной диагонали.
7. Упорядочить строки действительной матрицы порядка MxN по возрастанию значений заданного столбца.
8. Упорядочить столбцы действительной матрицы порядка MxN по убыванию значений заданной строки.
9. Найти средние значения отрицательных и положительных элементов столбцов действительной матрицы порядка MxN.
10. Найти среднее количество отрицательных и положительных элементов строк действительной матрицы порядка MxN.
11. Найти максимальный элемент над и под главной диагональю квадратной матрицы порядка NxN.
12. Найти минимальный элемент над и под побочной диагональю квадратной матрицы порядка NxN.
13. Найти произведение положительных элементов квадратной матрицы порядка NxN, расположенных выше главной диагонали.
14. Найти произведение ненулевых элементов квадратной матрицы порядка NxN, расположенных на побочной диагонали и ниже ее.
15. В действительной матрице порядка MxN указать индексы всех элементов с наибольшим значением.
16. В действительной квадратной матрице порядка NxN указать индексы отрицательных элементов, расположенных выше главной диагонали.
17. В действительной квадратной матрице порядка NxN вывести суммы строк, в которых элементы, относящиеся к главной диагонали, отрицательны.
18. В действительной квадратной матрице порядка NxN вывести максимальные значения столбцов, в которых расположены равные нулю элементы побочной диагонали.

19. В действительной матрице порядка MxN определить номера строк, сумма элементов которых имеет заданное значение.
20. В действительной матрице порядка MxN определить номера столбцов, количество положительных элементов которых превышает заданное.
21. Определить количество отрицательных элементов, расположенных на диагоналях квадратной матрицы порядка NxN.
22. Определить среднее значение положительных элементов, расположенных вне диагоналей квадратной матрицы порядка NxN.
23. В действительной матрице порядка MxN найти столбцы с минимальной суммой отрицательных элементов.
24. В действительной матрице порядка MxN найти строки с максимальным средним значением неотрицательных элементов.
25. В матрице переставить в обратном порядке элементы тех строк, которые заканчиваются отрицательными элементами.
26. В матрице переставить в обратном порядке элементы тех столбцов, которые начинаются нулевыми элементами.
27. Определить, симметричны ли верхняя и нижняя половины матрицы порядка MxN.
28. Из заданной матрицы размером MxN убрать средний столбец, если N – нечетное, и среднюю строку, если M –нечетное.
29. В действительной матрице порядка MxN поменять местами первую строку из содержащих максимальный по величине элемент, с последней строкой из содержащих минимальный по величине элемент.
30. В действительной квадратной матрице порядка NxN поменять местами последнюю строку из содержащих максимальный по величине элемент, с первым столбцом из содержащих минимальный по величине элемент.

Список литературы

1. Гарнаев А.Ю. Самоучитель VBA [Текст]. – СПб.: БХВ-Петербург, 2004. 560с.: ил.
2. Дейкстра Э. Дисциплина программирования. [Текст] – М.: Мир, 1978. - 278 с.
3. Кузьменко В.Г. – Программирование на VBA 2003. [Текст] –М.: Бином-Пресс, 2004. 880с.: ил.
4. Непейвода Н.Н. Стили и методы программирования. [Электронный ресурс] http://www.intuit.ru/shop/product-2493364.html Интернет-университет информационных технологий – ИНТУИТ.ру, 2005
5. Кнут Д. Искусство программирования, том 1. Основные алгоритмы — 3-е изд. [Текст] — М.: «Вильямс», 2006. — 720 с
6. Кнут Д. Искусство программирования, том 2. Получисленные методы — 3-е изд. [Текст] — М.: «Вильямс», 2007. — 832 с.
7. Кнут Д. Искусство программирования, том 3. Сортировка и поиск — 2-е изд. [Текст] — М.: «Вильямс», 2007. — 824 с.