МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

ВВЕДЕНИЕ В РАЗРАБОТКУ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Методические рекомендации к лабораторным работам для студентов специальности 6-05-0611-01 «Информационные системы и технологии» очной формы обучения

Могилев 2024

УДК 004
ББК 32.81
В74

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «16» января 2024 г., протокол № 7

Составитель канд. техн. наук, доц. В. М. Ковальчук

Рецензент канд. техн. наук, доц. С. К. Крутолевич

Методические рекомендации предназначены для студентов специальности 6-05-0611-01 «Информационные системы и технологии» очной формы обучения.

Учебное издание

ВВЕДЕНИЕ В РАЗРАБОТКУ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Ответственный за выпуск А. И. Якимов

Корректор И. В. Голубцова

Компьютерная верстка Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Цель методических рекомендаций к лабораторным работам по дисциплине «Введение в разработку программного обеспечения» заключается в овладении и закреплении студентами практических навыков работы в среде приложений MS Office.

Целью преподавания дисциплины является изучение основных современных пакетов прикладных программ для научных и инженерных расчетов, изучение основ программирования, общих вопросов алгоритмизации и приобретение навыков решения задач с применением средств вычислительной техники.

Дисциплина «Введение в разработку программного обеспечения» является неотъемлемой частью современных инженерных знаний и входит в состав естественно-научных дисциплин, компонентов учреждения высшего образования.

Полученные при изучении дисциплины знания и навыки будут востребованы при изучении специальных дисциплин инженерной направленности и станут инструментом для грамотного выполнения и оформления рефератов, курсовых и дипломных работ.

Каждая работа рассчитана на два часа.

Выполнение каждой работы производится в следующем порядке:

1) ознакомиться с теоретическими положениями работы;

2) из таблицы «Варианты заданий для выполнения работы» по указанию преподавателя выбрать исходные данные для выполнения задания и оформить рукописный отчет.

Отчет содержит: название и цель работы; постановку задачи; исходные данные; использованные технологии; результаты выполнения; анализ полученных результатов; выводы. В отчете можно привести также ответы на наиболее сложные вопросы, приведенные в конце каждой работы.

1 Лабораторная работа № 1. Алгоритмический язык VBA. Разработка линейной программы со вводом и выводом данных

Цель работы: ознакомиться с основными окнами среды **VBA** и с процессом конструирования визуального проявления программы.

Методические указания

Корпорация Microsoft интегрировала в свои офисные продукты язык программирования Visual Basicfor Applications (VBA). С помощью этого языка каждый пользователь может автоматизировать работу приложения и максимально приспособить его работу для решения текущих задач.

Код VBA набирается в редакторе VisualBasic. Для того чтобы попасть в этот редактор, выберите в MS Excel команду Сервис | Макрос | Редактор Visual Basic или нажмите комбинацию клавиш <Alt> + <F11>. В результате откроется интегрированная среда разработки приложений (IDE).

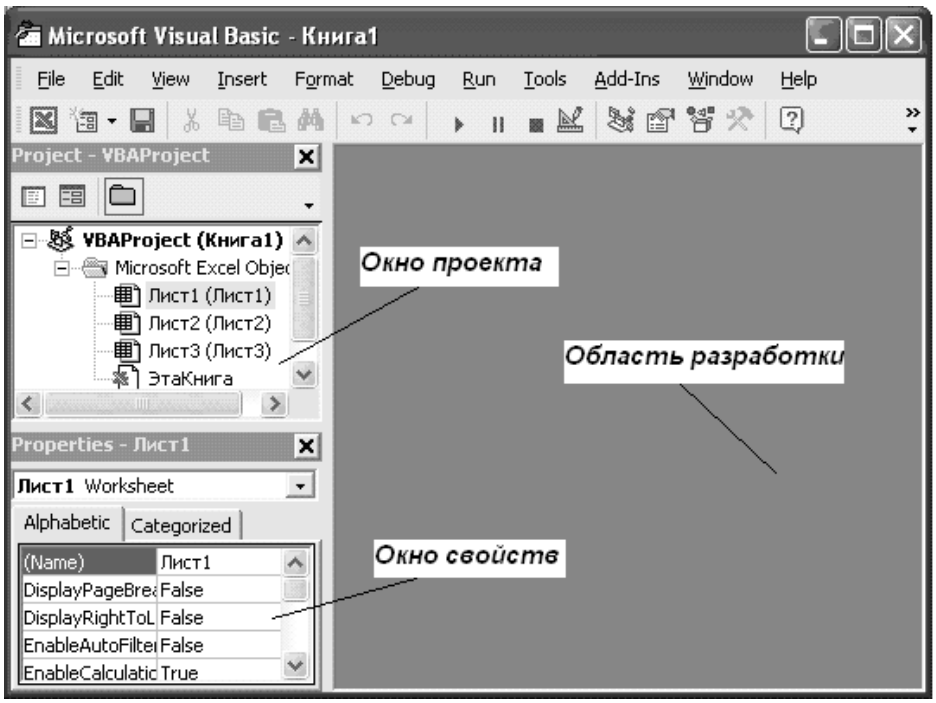

Рисунок 1.1 – Интегрированная среда разработки приложений (IDE)

Возвратиться из редактора Visual Basic в рабочую книгу можно, нажав кнопку с пиктограммой Excel.

Все приложения, написанные на VBA, создаются как проекты. В проект входят несколько модулей: код программы, параметры формы, конфигурация интегрированной среды разработки и др.

Среда разработки приложений имеет стандартный для Windows приложений вид: строка меню, панель инструментов и еще несколько открытых окон. На рисунке 1.1 открыты два окна: **Project – VBA Project** и **Properties**.

Окно **Project – VBA Project** активизируется выбором команды **View Project Explorer** (1) или нажатием кнопки **Project Explorer** (2) панели инструментов. В окне **Project – VBAProject** представлена иерархическая структура форм и модулей текущего проекта. В этом окне отображается реестр модулей и форм, входящих в создаваемый проект. Двойным щелчком на значке модуля в окне **Project – VBA Project** можно открыть соответствующий модуль.

В проекте автоматически создается по модулю для каждого рабочего листа и для всей книги. Кроме того, можно создавать и другие модули.

При написании кода программы необходимо придерживаться следующей структуры:

SubИмяПрограммногоМодуля ()
Объявление переменных и констант
Тело программы (последовательность исполняемых операторов)
EndSub

Переменные в программе используются для хранения данных, которые могут изменяться в процессе выполнения процедуры. Объявление типа переменной означает, что пользователь устанавливает определённые границы, в которых может изменяться переменная. Тип переменной можно вообще не определять. Если тип переменной не объявляется, по умолчанию он принимается как тип Variant. В таблице 1.1 содержится информация о размере данных, т. е. об объёме памяти, выделяемом для хранения данных.

Таблица 1.1 – Основные типы переменных языка VBA

Тип	Содержание	Объем	Диапазон значений
Byte	Короткое неотрицательное число	1 байт	0...255
Integer	Целое число	2 байта	–32768...+32767
Long	Длинное целоечисло	4 байта	–2147483648... +2147483647
Single	Десятичное число обычной точности	4 байта	$\pm(1{,}4011298e^{-45}$... $3{,}402823e^{+38})$
Double	Десятичное число двойной точности	8 байт	$\pm(1e^{-324}...1e+^{308})$
String	Набор символов (строка)	Зависит от числа символов в строке	
Boolean	Логическая величина	2 байта	Истина (True) илиЛожь (False)
Date	Дата	8 байт	Информация о дате
Object	Указательна объект	4 байта	Значение мявляется с сылка на объект
Variant	Произвольное значение	Не менее 16 байт	Может быть переменной любого типа

Для объявления переменной используется оператор Dim. Этот оператор имеет следующий синтаксис:

DimИмяПеременнойAsTunДанных

Имена переменных должны начинаться с буквы, могут содержать также цифры и знаки подчёркивания. Имя не может содержать пробелы, точки, запятые, восклицательные знаки и символы @, &, $, # и не должно иметь более 255 символов.

Если в вычислениях нужна величина, которая бы не меняла своего значения, то применяются константы. Для их объявления используется оператор Const, имеющий следующий синтаксис:

ConstИмяКонстантыAsTunДанных = Значение

Например:

ConstGruppaAsInteger = 25

Для задания переменным исходных значений используется функция InputBox, а для вывода значений переменных на экран – процедура MsgBox. Синтаксис их написания можно посмотреть в справочной системе при работе с кодом программы с помощью клавиши F1.

Для вычисления значений переменных в программе используется оператор присваивания, имеющий следующий синтаксис:

Переменная = Выражение

Если *Выражение* математическое, то в нем используются знаки арифметических действий, математические функции и круглые скобки, например:

$$A = 2 \cdot \sin(x+2) / \cos(x-2)$$

Задание

1 Получить от преподавателя математическое выражение, результат которого необходимо вычислить.

2 Составить необходимый код программы с обеспечением ввода исходных данных и вывода результата на экран.

3 Запустить программу на выполнение и сравнить полученный результат со значением, вычисленным другим способом, например с помощью калькулятора.

Контрольные вопросы

1 Как объявить переменную, которая может иметь целые значения в диапазоне от 0 до 100?

2 Как в программе присвоить значение переменной?
3 Как показать результат вычислений?

2 Лабораторная работа № 2. Алгоритмический язык VBA. Разработка разветленной программы с использованием оператора IF

Цель работы: изучить оператор ветвления ***If***; создать программу с использованием оператора ветвления ***If*** на языке программирования VBA.

Методические указания

Алгоритм называется разветвляющимся, если последовательность выполнения его шагов изменяется в зависимости от выполнения некоторых условий. Условие – это логическое выражение, которое может принимать одно из двух значений: «*ДА*» – если условие верно (истинно, *TRUE*); «*НЕТ*» – если условие неверно (ложно, *FALSE*).

Схема алгоритма конструкции условного оператора ***If*** представлена на рисунке 2.1. Синтаксис условного оператора ***If*** в однострочной форме следующий:

If<лог. выраж.>ThenP_1 :P_2 : ... :P_NElseM_1 : M_2 : ... M_N

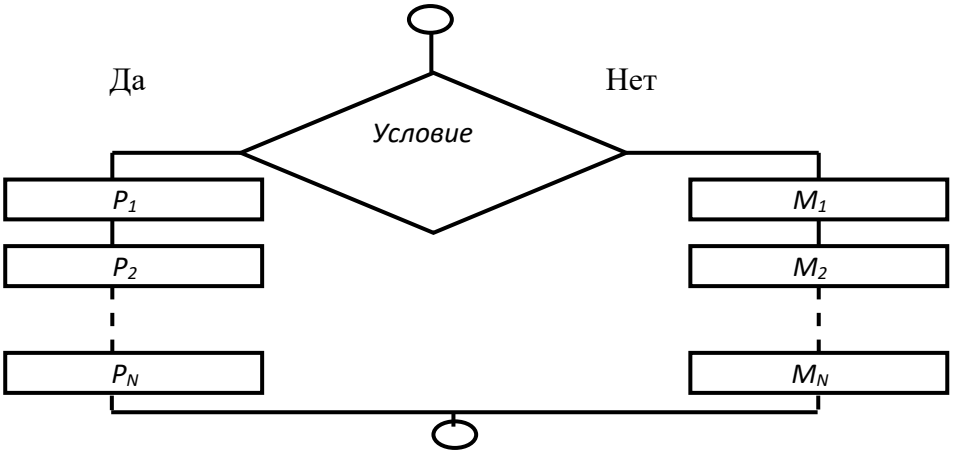

Рисунок 2.1 – Схема алгоритма конструкции условного оператора ***If***

Возможна и другая синтаксическая форма – блочная (структурная):

If<логическое выражение> Then
    P_1
...
    P_N
Else

M_1

...

M_N

End If

где *If*, *Then*, *Else*, *End If* – зарезервированные слова;

P_1, P_2, P_N, M_1, M_2, M_N – операторы.

Задание

1 Запросить у пользователя ввод числа.

2 Сравнить введенное число с другим, заданным числом, например 20.

3 По результатам сравнения вывести соответствующее сообщение:

«25>20» или «15<20»,

где 25, 15 – введенные пользователем числа.

Контрольные вопросы

1 Можно ли вставлять инструкцию ***Else*** перед инструкцией ***ElseIf*** в блочном варианте оператора ветвления?

2 В каких случаях в программе используется полный условный оператор? Как он оформляется? Как он работает (что происходит при его выполнении)?

3 В каких случаях в программе используется неполный условный оператор? Как он оформляется?

3 Лабораторная работа № 3. Алгоритмический язык VBA. Разработка разветленной программы с использованием вложенного оператора IF

Цель работы: изучить оператор многозначного ветвления ***If***; разработать программу, реализующую выбор из нескольких альтернатив (более 2).

Методические указания

Схема алгоритма конструкции оператора многозначных ветвлений ***If*** представлена на рисунке 3.1.

Синтаксис оператора многозначных ветвлений ***If*** в блочной (структурной) форме следующий:

If<лог.выражение1>Then

P_1

ElseIf<лог.выражение2>Then

P_2
Else
P_3
End If

где *If*, *Then*, *Else*, *End If* – зарезервированные слова;
P_1, P_2, P_3 – операторы.

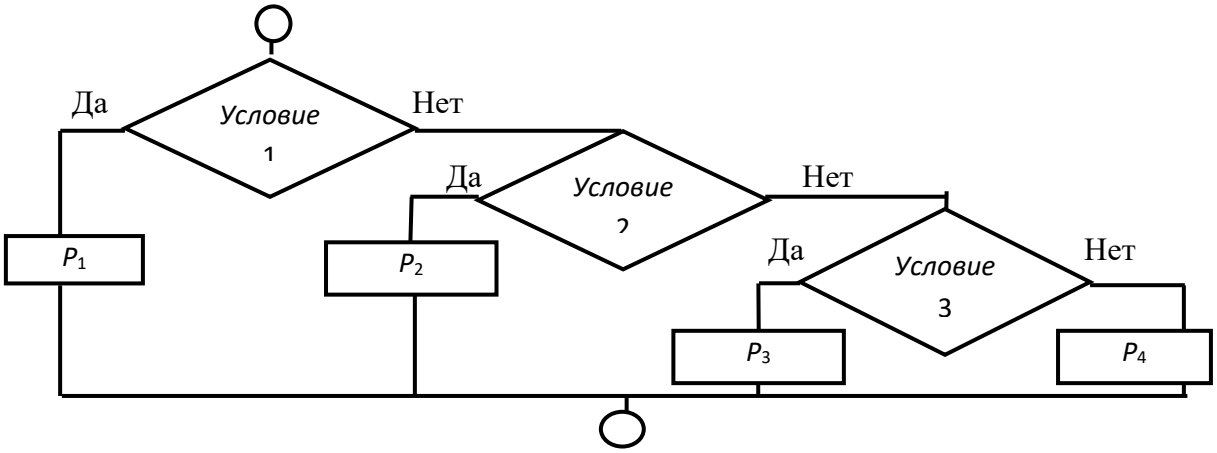

Рисунок 3.1 – Схема алгоритма конструкции оператора многозначных ветвлений ***If***

Алгоритм работы такой конструкции:

– если логическое выражение **1** истинно, то выполняется оператор $\boldsymbol{P_1}$ (или блок операторов), следующий за конструкцией ***Then***, а остальные операторы пропускаются;

– если логическое выражение **1** ложно, то оператор $\boldsymbol{P_1}$ пропускается и анализируется логическое выражение **2**, следующее за ***ElseIf***. Если оно истинно, то выполняется оператор $\boldsymbol{P_2}$ (или блок операторов), следующий за ***Then***, а остальные операторы пропускаются;

– оператор $\boldsymbol{P_3}$ (или блок операторов), следующий за последним ***Else***, выполняется лишь в том случае, если ложны все логические выражения.

Далее представлены примеры использования условного оператора ***if*** и многозначного ветвления, позволяющего определить возможность поступления абитуриента в учреждения образования. На первом шаге происходит ввод пользователем балла, полученного на тестировании. На втором шаге, если полученный балл более 60, приложение выдает сообщение: «Вам можно поступать в ВУЗ». На третьем шаге, если полученный балл более 20, приложение выдает сообщение: «Вам можно поступать в СУЗ». На четвертом шаге, если полученный балл более 5, приложение выдает сообщение: «Вам можно поступать в ПТУ».

```
Subball()
DimballAsVariant
ball = InputBox(«Введите полученный Вами балл на тестировании»)
```

```
Ifball>=60 Then
MsgBox («Вам можно поступать в ВУЗ»)
ElseIfball>=20 Then
        MsgBox («Вам можно поступать в СУЗ»)
ElseIfball>=5 Then
        MsgBox («Вам можно поступать в ПТУ»)
EndIf
```

Задание

1 Запросить у пользователя ввод целого числа от 0 до 100 включительно – оценка по 100-балльной системе.

2 С применением оператора многозначных ветвлений ***If*** осуществить преобразование введенной оценки по 100-балльной системе в 5-балльную по шкале, предварительно созданной на листе Excel (таблица 3.1).

Таблица 3.1 – Преобразование оценок

Значение оценки по 100-балльной системе	Значение оценки по 5-балльной системе
От 0 до 20	Оценка 1
От 20 до 40	Оценка 2
От 40 до 60	Оценка 3
От 60 до 80	Оценка 4
От 80 до 100 включ.	Оценка 5
Меньше 0 или больше 100	Ошибка ввода данных

3 По результатам вывести сообщение с введенной оценкой по 100-бальной системе и полученной оценкой по 5-балльной системе либо сообщение об ошибке.

Контрольные вопросы

1 В каких случаях в программе используется вложенный условный оператор?

2 Сколько инструкций ***ElseIf*** может быть в блочном варианте оператора ветвления?

3 Нарисуйте алгоритмическую схему выполнения вложенного условного оператора.

4 Лабораторная работа № 4. Алгоритмический язык VBA. Разработка разветленной программы с использованием оператора выбора

Цель работы: ознакомиться с оператором выбора ***SelectCase*** и закрепить полученные знания на практике.

Методические указания

При наличии большого количества ветвлений конструкция многозначных ветвлений ***If*** становится тяжёлой для восприятия. В подобных случаях хорошей альтернативой оператору ***If*** служит оператор выбора ***SelectCase***, который позволяет выбрать одно из нескольких возможных продолжений программы.

В то время как ***If...Then...Else*** для каждой инструкции ***ElseIf*** оценивает разные выражения, инструкция ***SelectCase*** оценивает выражение только один раз, в начале управляющей структуры. ***SelectCase*** выполняет одну из нескольких групп инструкций в зависимости от значения выражения. Синтаксис:

SelectCase<*выражение*>
*[**Case***<*списокВыражений-n*>
[инструкции-n]] ...
*[**CaseElse***
[инструкции_else]]
EndSelect

<выражение> − обязательный. Любое числовое выражение или строковое выражение.

<списокВыражений-n> − обязательный при наличии предложения Case. Список с разделителями, состоящий из одной или нескольких форм следующего вида:

<инструкции-n> − необязательный. Одна или несколько инструкций, выполняемых в том случае, если выражение совпадает с любым компонентом списка <***списокВыражений-n***>;

<инструкции_else>− необязательный. Одна или несколько инструкций, выполняемых в том случае, если выражение не совпадает не совпадает ни с одним из предложений ***Case***.

Синтаксис оператора ***SelectCase*** также может содержать следующие элементы:

- **выражение *To* выражение**;
- ***Is*** оператор сравнения **выражение**.

Ключевое слово ***To*** задает диапазон значений. При использовании ключевого слова ***To*** перед ним должно находиться меньшее значение. Ключевое слово ***Is*** с операторами сравнения (кроме ***Is*** и ***Like***) задает диапазон значений. Если ключевое слово ***Is*** не указано, оно вставляется по умолчанию.

Если выражение совпадает с любым выражением из ***списка Выражений*** в предложении ***Case***, выполняются все инструкции, следующие за данным предложением ***Case*** до следующего предложения ***Case***, или, для последнего предложения, до инструкции ***EndSelect***. Затем управление передается инструкции, следующей за ***EndSelect***. Если выражение совпадает с выражениями из списка в нескольких предложениях Case, выполняется только первый подходящий набор инструкций.

Предложение ***CaseElse*** задает список ***инструкции_else***, которые будут выполнены, если не обнаружено ни одно совпадение выражения и компонента ***список Выражений*** ни в одном из остальных предложений ***Case***. Хотя данное предложение не является обязательным, рекомендуется помещать предложение ***CaseElse*** в блок ***SelectCase,*** чтобы предусмотреть неожиданные значения выражения. Если ни в одном предложении ***Case список Выражений*** не содержит компонента, отвечающего аргументу выражения, и отсутствует инструкция ***CaseElse***, выполнение продолжается с инструкции, следующей за инструкцией **EndSelect**. Пример использования оператора **SelectCase** представлен в таблице 4.1.

Если значение переменной ***vozrast*** меньше или равно **7**, отображается сообщение «*Ты дошкольник*»; если значение находится в диапазоне от 8 до 16 – сообщение «*Ты учишься в школе*»; если в диапазоне от 17 до 30 – сообщение «*Тебе пора заняться делом*»; если в диапазоне от 31 до 60 – сообщение «*Кто не работает, тот не ест*». Если значение возраста не равно ни одному из предложенных диапазонов значений, выводится сообщение «*Заслуженный отдых*».

Таблица 4.1 – Синтаксис и пример использования оператора ***SelectCase***

Синтаксис оператора *Select Case*	Пример использования оператора *SelectCase*
Select Case Ключ Выбора *Case Is выражение* *оператор* *Case диапазон значений* *оператор* *Case диапазон значений* *оператор* *Case диапазон значений* *оператор* *CaseElse* *оператор* *End Select*	*Select Case vozrast* *Case Is <=7* *Msgbox «Ты дошкольник»* *Case 8 to 16* *Msgbox «Ты учишься в школе»* *Case 17 to 30* *Msgbox «Тебе пора заняться делом»* *Case 31 to 60* *Msgbox «Кто не работает, тот не ест»* *CaseElse* *Msgbox «Вы заслужили отдых»* *EndSelect*

Из представленного в таблице 4.1 примера видно, что код этой процедуры более прост для восприятия, чем многозначные ветвления ***If***.

Задание

Выполнить задание предыдущей лабораторной работы с применением оператора выбора ***SelectCase***.

Контрольные вопросы

1 В каких случаях в программе используется оператор выбора?

2 В чем преимущество оператора выбора варианта перед многовариантным оператором ветвления?

3 Когда применение оператора ***Select Case*** эффективнее оператора ***If Then Else EndIf*** ?

5 Лабораторная работа № 5. Алгоритмический язык VBA. Разработка программы с использованием оператора выбора FOR. Ввод и вывод одномерного массива

Цель работы: изучить оператор цикла ***For***.

Методические указания

Часто в программах необходимо реализовать определённые операторы несколько раз. В этих случаях организуют циклические вычисления. Алгоритм называется ***циклическим***, если определенная последовательность шагов выполняется несколько раз в зависимости от заданных условий. Циклические алгоритмы могут быть осуществлены с применением следующих операторов цикла: ***For … Next***, ***While … Wend***, ***Do … Loop***, которые позволяют повторить группу операторов или один оператор заданное количество раз.

Общий вид алгоритма конструкции оператора цикла ***For … Next*** представлен на рисунке 5.1.

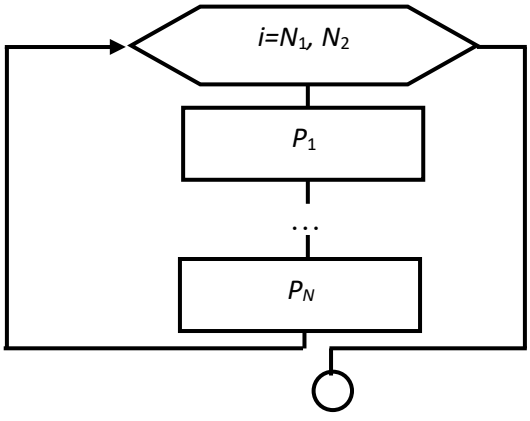

Рисунок 5.1 – Схема алгоритма конструкции оператора цикла ***For … Next***

Синтаксис конструкции оператора цикла ***For … Next*** следующий:

For i = N_1 To N_2 [Step h]
P_1
...
[Exit For]
P_N
Next i

Телоцикла

For (для), ***To*** (до), ***Step*** (шаг), ***Exit For*** (выход из ***For***), ***Next*** (следующий) – служебныеслова ***VBA***, а ***P*1**, ***PN*** – операторы. ***Step*** является необязательным параметром. Если он опущен в программе, то значение параметра ***i*** увеличивается на **1**. Параметр ***Step*** может быть любым действительным числом, как целым, так и дробным, как положительным, так и отрицательным. Оператор ***Exit For*** позволяет выйти из цикла ***For … Next*** до его завершения. Тем самым программа сможет среагировать на определённое событие, не выполняя цикл заданное число раз.

Задание

1 Запросить у пользователя ввод целого числа больше 0.

2 Определить произведение последовательности чисел от 1 до n включительно (n! – «n факториал»).

3 Результат вычисления вывести в виде сообщения.

Пример сообщения: «Факториал 10! = 1 · 2 · 3 · 4 · 5 ·6 · 7 · 8 · 9 · 10 = = 3 628 800».

Контрольные вопросы

1 Может ли тело оператора цикла с параметром не выполниться ни разу?

2 Как должен быть оформлен оператор цикла с параметром, чтобы тело цикла выполнялось при уменьшающихся значениях параметра цикла?

3 Чему равно количество повторений тела оператора цикла с параметром, если параметр цикла принимает:

а) все целые значения от 1 до 10;

б) все значения от 10 до 100 с шагом 7;

в) все значения от a до b с шагом ***step***.

4 Можно ли в теле оператора цикла использовать условный оператор?

5 Какие вы знаете операторы для принудительного (преждевременного) выхода из оператора цикла?

6 Лабораторная работа № 6. Алгоритмический язык VBA. Разработка программы с вложенными циклами. Обработка многомерных массивов

Цель работы: изучить вложенные операторы цикла ***For***.

Методические указания

Если телом цикла является циклическая структура, то такие циклы называются вложенными. Цикл, содержащий в себе другой цикл, – внешний, а цикл, содержащийся в теле другого цикла, – внутренний. Синтаксис конструкции вложенных операторов цикла ***For … Next*** следующий:

Fori = N_1 *To*N_2
Forj = M_1 *To*M_2
P_1
…
P_N
Nextj
Nexti

тело внутреннего цикла

тело внешнего цикла

Общий вид алгоритма конструкции вложенных операторов цикла ***For … Next*** представлен на рисунке 6.1.

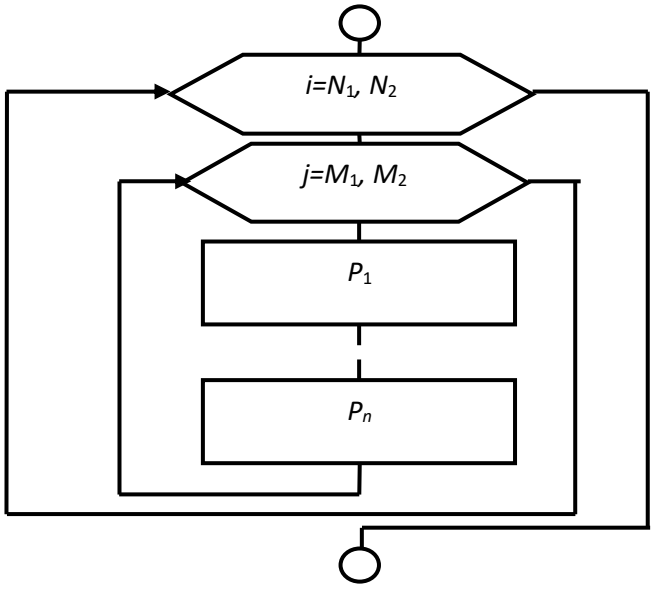

Рисунок 6.1 – Схема алгоритма конструкции вложенных операторов цикла ***For … Next***

При первом вхождении в цикл параметр внешнего цикла ***i*** принимает значение, равное N_1, и управление передаётся во внутренний цикл, в котором параметр цикла ***j*** принимает значение, равное M_1, и выполняется оператор (операторы), который записан во внутреннем цикле. Затем параметр внутрен-

него цикла ***j*** увеличивается на **1**, и вновь повторяется тело цикла.

Операторы P_1, P_N будут реализовываться до тех пор, пока параметр цикла ***j*** не станет больше величины M_2. Затем параметр внешнего цикла ***i*** увеличивается на **1**, и вновь начинает свою работу внутренний цикл, в котором параметр цикла ***j*** будет изменяться от M_1 до M_2, и при каждом прохождении цикла будут выполняться операторы P_1 и P_N. Внешний цикл закончит свою работу, когда параметр цикла ***i*** станет больше величины N_2.

Задание

С помощью вложенных операторов цикла ***For…Next*** реализовать запись на рабочий лист ***Excel*** матрицы размерностью $n \times n$ (значение числа ***n*** запросить у пользователя), элементы которой должны являться произведением номера строки на номер колонки.

Контрольные вопросы

1 В каких случаях используются вложенные операторы цикла?

2 Как оформляются вложенные операторы цикла?

3 Нарисуйте алгоритмическую схему выполнения вложенных операторов цикла.

4 Вложенный цикл образован двумя операторами цикла с параметром. Что является телом?

5 Может ли внешний оператор вложенного цикла:

а) не выполниться ни разу;

б) выполняться бесконечное число раз (или до того момента, когда пользователь прервет его выполнение)?

7 Лабораторная работа № 7. Использование пакета MathCad для решения математических задач. Вычисление функций и построение графиков в MathCad

Цель работы: ознакомиться со структурой математического процессора MathCad; освоить ввод данных вычисления математических выражений.

Методические указания

Система MathCad – одна из самых мощных и эффективных систем математического направления, которая ориентирована на широкий круг пользователей и позволяет выполнять математические расчеты как в численном, так и в символьном виде. Причем описание решения задач задается с помощью привычных математических формул и знаков.

Документ системы MathCad строится из областей, которые делятся на вычислительные, графические, текстовые.

В вычислительных областях можно задавать данные, выражения, операторы и управляющие структуры. Все данные системы можно разделить на простые и структурированные. Простые данные представлены константами и переменными. Структурированные данные представлены дискретными переменными, массивами и файлами.

Константы – элементы данных, хранящие некоторые значения, которые не могут быть изменены

Переменные – поименованные объекты, имеющие некоторые значения, которые могут изменяться в процессе выполнения документа.

Имена переменных в системе MathCad могут содержать любые латинские и греческие буквы, а также цифры, они должны начинаться только с буквы. Строчные и прописные буквы в именах различаются. Имена должны быть уникальными, т. е. они не должны совпадать с именами встроенных или определенных пользователем функций. Примеры имен переменных:

A f k21 sum γ φ5

Выражение – это совокупность данных, функций и математических объектов, связанных знаками операций. Выражения могут содержать скобки.

Операции, используемые в выражениях, можно разделить на арифметические и логические. Арифметические операции представлены в палитре арифметических операторов. Логические операции и операции отношения представлены в палитре логических операторов.

К базовым операторам системы относятся:

:= – оператор локального присваивания;

≡ – оператор глобального присваивания;

= – оператор вычисления и вывода.

Оператор локального присваивания (:=) распространяет свое действие на область документа, расположенную в строке и ниже места присваивания. Этот оператор выполняется так: данному, стоящему в левой части оператора присваивается вычисленное значение выражения, стоящего в правой части оператора.

Оператор глобального присваивания (≡) не зависит от места присвоения и распространяет свое действие на весь документ. Этот оператор выполняется точно так же, как и оператор локального присваивания.

Оператор вычисления и вывода (=) выводит вычисленное значение выражения, стоящего в его левой части, на экран.

Функция – выражение, согласно которому проводятся некоторые вычисления с ее аргументами и определяется числовое значение. Функция имеет имя и может иметь список параметров. Различают стандартные и пользовательские функции.

Общий вид описания функции следующий:

ИМЯ (СФП):=выражение,

где *ИМЯ* – имя функции;

СФП – список формальных параметров функции.

При обращении к функции формальные параметры заменяются на фактические, т.е. на выражения, имеющие числовые значения. Например,

$z(m,n) := m^2 + n^2$ – описание функции,

$z(2,3) = 13$ – обращение к функции.

MathCad позволяет обрабатывать различные виды графической информации. Возможности системы по работе с графикой таковы:

– построение двумерных графиков в декартовой и полярной системах координат;

– построение трехмерных поверхностных графиков;

– внесение рисунков, созданных другими компьютерными системами;

– создание анимационных клипов.

Для построения графиков используется палитра графиков. Перечень возможных типов графиков приведен в основном меню ***Insert*** – ***Graph.***

При построении двумерных графиков после нажатия соответствующей кнопки на панели графических инструментов появляется шаблон (рисунок 7.1).

В шаблоне графика по вертикали задаются через запятую функции, а по горизонтали – аргументы. Крайние шаблоны данных служат для указания предельных значений абсцисс и ординат, т. е. они задают масштабы графика. Если оставить эти шаблоны незаполненными, то масштабы по осям графика будут устанавливаться автоматически.

В полярной системе координат каждая точка задается углом ***a***, радиусом и длиной радиус-вектора ***R(a).*** График функции обычно строится при изменении угла ***a*** в пределах от 0 до 2π.

После построения график может быть отформатирован по следующим направлениям: форматирование осей графика; форматирование линий графика; форматирование надписей на графике.

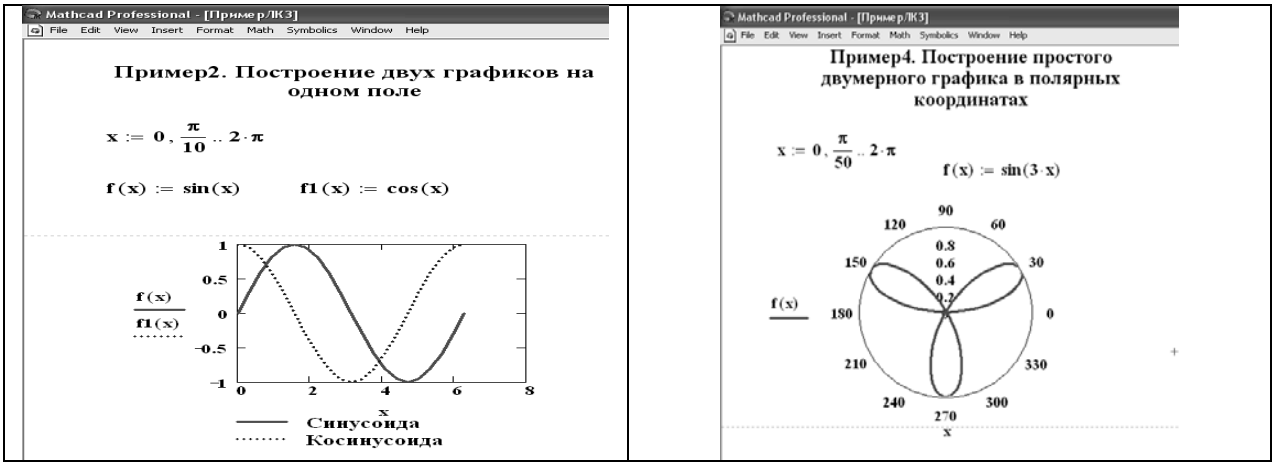

Рисунок 7.1 – Примеры

Порядок выполнения работы

1 Построить графики функций (рисунок 7.2).

В декартовых координатах: В полярных координатах:

1. x^2 x^2+2 $(x+2)^2$	5. $12x-4+3$ $\lvert 12x-4\rvert+3$ $\lvert\lvert 12x-4\rvert+3\rvert$	1. Кардиоида $d(n) := 4(1-\cos(n))$	5. Спираль «жезл» $k(n) := \frac{8}{\sqrt{n}}$ (R=16)
2. $\sqrt{x}$ $\sqrt{x}-1$ $\sqrt{x-1}$	6. $\ln x$ $\ln x+1$ $\ln(x+1)$	2. Декартов лист $s(a) := \frac{9\cdot\sin(a)\cdot\cos(a)}{\sin(a)^3+\cos(a)^3}$	6. Розы $st(a) := 5\sin(8a)$ (R=8)
3. $\cos x$ $\cos x+3$ $\cos(x+3)$	7. e^x+1 e^x e^{x+1}	3. Розы $st(a) := 5\sin(5a)$ (R=8)	7. Розы $h(t) := 2\sin\left(\frac{t}{3}\right)$
4. $\sin x$ $\sin x+3$ $\sin(x+3)$	8. $\log_5 x$ $\log_5 x+3$ $\log_5(x+3)$	4. Спираль Архимеда $h(t) := 7t$	8. Розы $st(a) := 5\sin(3a)$ (R=8)

Рисунок 7.2 – Варианты задания

Контрольные вопросы

1 На какие типы задач ориентирована система Mathcad?
2 Какие указатели присутствуют в окне системы Mathcad?
3 Чем отличаются команды :=, =, ≡?
4 Как классифицируются функции MathCad?
5 Какой формат имеют пользовательские функции?
6 Как создается массив дискретных переменной аргумента функции?
7 Какую графическую информацию обрабатывает MathCad?
8 Как можно разместить на одном графике несколько функций?

8 Лабораторная работа № 8. Использование пакета MathCad для решения математических задач. Решение алгебраических уравнений в MathCad

Цель работы: ознакомиться с методами решения алгебраических уравнений в MathCad.

Методические указания

Для алгебраических уравнений вида $f(x) = 0$ решение в MathCad находится с помощью функции *root*.

Общий вид функции следующий:

$$root\left(f\left(x\right),x\right),$$

где $f(x)$ – функция, описывающая левую часть выражения вида $f\left(x\right)=0$;

x – имя переменной, относительно которой решается уравнение.

Функция *root* реализует алгоритм поиска корня численным методом и требует предварительного задания начального приближения искомой переменной *x*. Поиск корня будет производиться вблизи этого числа. Таким образом, присвоение начального значения требует предварительной информации о примерной локализации корня.

Если после многих итераций MathCad не находит подходящего приближения, то появится сообщение «отсутствует сходимость».

Эта ошибка может быть вызвана следующими причинами:

– уравнение не имеет корней;

– корни уравнения расположены далеко от начального приближения;

– выражение $f(x)$ имеет разрывы между начальным приближением и корнем;

– выражение имеет комплексный корень, но начальное приближение было вещественным и наоборот.

В зависимости от типа задачи функция *root* может включать либо два, либо четыре аргумента и, соответственно, использует разные алгоритмы поиска корней:

$$root\left(f\left(x\right),x\right);$$

$$root\left(f\left(x\right),x,a,b\right),$$

где $f(x)$ – скалярная функция, определяющая уравнение $f\left(x\right)=0$;

x – имя скалярной переменной, относительно которой решается уравнение;

a,b – границы интервала, внутри которого происходит поиск корня.

Первый тип функции *root* требует дополнительного задания начального значения переменной *x* (примерная локализация корня, поиск корня будет

производиться вблизи этого числа).

Для задания начального значения или интервала удобно предварительно построить график функции. Чем точнее выбрано начальное приближение корня, тем быстрее будет *root* сходиться.

Пример – Найти корень уравнения $\sin(2x)+\cos(2x)=\sqrt{x}\sin(3x)$ при заданном начальном значении $x\approx 5$. Выполнить графическую интерпретацию результата (рисунок 8.1).

Реализация в MathCad:

– привести уравнение к виду $f(x)=0$, если это необходимо, например, уравнение $\sin(2x)+\cos(2x)=\sqrt{x}\sin(3x)$ преобразуется в $\sin(2x)+\cos(2x)-\sqrt{x}\sin(3x)=0$;

– установить курсор в свободное место рабочего окна документа MathCad;

– определить функцию, содержащуюся в левой части уравнения вида $f(x)=0$,например, $f(x)=\sin(2x)+\cos(2x)=\sqrt{x}\sin(3x)$;

– построить график функции $f(x)$;

– установить оси графика в виде креста;

– задать начальное приближение искомой переменной ***x***;

– подставляя в качестве параметров функции ***root*** имя функции левой части уравнения и переменную, содержащую начальное приближение, вывести значение корня уравнения при заданном начальном приближении с помощью оператора «=»;

– задать аргумент функции ***f (x)*** в виде дискретной переменной, при этом диапазон изменения аргумента должен включать найденный корень уравнения.

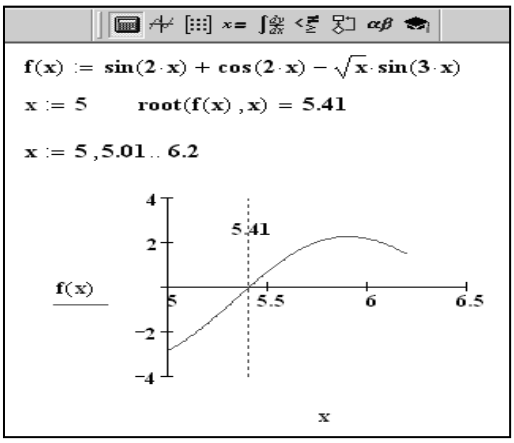

Рисунок 8.1 – Пример

Порядок выполнения работы

1 Ознакомиться с методом решения алгебраических уравнений с помощью функции *root* и выполнить разобранный выше пример.

2 С помощью функции *root* решить следующие уравнения:

$\cos(x) - x - 0.2 = 0;$

$x^3 - 10 \cdot x + 2 = 0;$

$4 \cdot x^4 - 10 \cdot x^3 + 2 \cdot x = 0.$

Контрольные вопросы

1 Какие функции имеются в пакете MathCad для решения уравнений?
2 Какие алгоритмы поиска корней уравнений реализует функция *root?*
3 С какой целью желательно предварительно построить график функции?
4 По каким причинам может появится сообщение «отсутствует сходимость»?

Список литературы

1 **Голицына, О. Л.** Информационные системы : учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. – 2-е изд. – Москва: ФОРУМ; ИНФРА-М, 2022. – 448 с.

2 **Гагарина, Л. Г.** Введение в архитектуру программного обеспечения: учебное пособие / Л. Г. Гагарина, А. Р. Федоров, П. А. Федоров. – Москва: ФОРУМ; ИНФРА-М, 2023. – 320 с.