УДК 519.688

1.2.2. Математическое моделирование, численные методы и комплексы программ (физико-математические науки)

О МЕТОДАХ И ТЕХНОЛОГИЯХ РАЗРАБОТКИ КОМПЬЮТЕРНЫХ ПРОГРАММ ДЛЯ ГЕНЕРАЦИИ И ПРОВЕРКИ ЗАДАНИЙ ПО МАТЕМАТИЧЕСКИМ ДИСЦИПЛИНАМ

Аршинов Георгий Александрович
д.т.н., профессор
ФГБОУ «Кубанский государственный аграрный университет», 350044, Россия, г. Краснодар, ул. Калинина 13

Лаптев Владимир Николаевич
к.т.н., доцент
ФГБОУ «Кубанский государственный аграрный университет», 350044, Россия, г. Краснодар, ул. Калинина 13

Михайленко Евгений Владимирович
к.ф.-м.н., доцент, заместитель начальника кафедры
ФГКОУ ВПО «Краснодарский университет МВД России»,350005, Россия, г. Краснодар, ул. Ярославская, 128,

Статья посвящена технологиям подготовки программных комплексов в среде Visual Basic for Application на базе офисных пакетов Microsoft Office, OpenOffice и LibreOffice для автоматизированной подготовки индивидуальных комплектов практических заданий по математическим дисциплинам, а также методам верификации выполненных обучающимися работ. В ней изложены основные принципы разработки представляемых программных продуктов, описаны структура типовых программных разработок и содержание входящих в них программных модулей, исследуются алгоритмы генерирования различных классов задач по разделам высшей математики: теории множеств, теории чисел, матричной алгебры, булевой алгебры и теории вероятностей, приведены фрагменты кодов разработанных процедур. Статья предназначена для профессорско-преподавательского состава, инженеров-программистов образовательных организаций

Ключевые слова: АЛГОРИТМ, БУЛЕВА АЛГЕБРА, ВЕРИФИКАЦИЯ, ГЕНЕРАЦИЯ, КОМПЬЮТЕРНАЯ ПРОГРАММА, МАКРОС, МЕТОДЫ ОПТИМИЗАЦИИ, ПОЛИНОМ, ПРАКТИЧЕСКОЕ ЗАДАНИЕ, ПРОГРАММНЫЙ МОДУЛЬ

http://dx.doi.org/10.21515/1990-4665-193-008

UDC 519.688

1.2.2. Mathematical modeling, numerical methods and software packages (Physics and Mathematics)

ABOUT METHODS AND TECHNOLOGIES FOR DEVELOPING COMPUTER PROGRAMS FOR GENERATING AND CHECKING TASKS IN MATHEMATICAL DISCIPLINES

Arshinov Georgy Alexandrovich
Doctor of Technical Sciences, Professor
Kuban State Agricultural university, 350044, Russia, Krasnodar, Kalinina,13

Laptev Vladimir Nikolaevich
Cand.Tech.Sci., associate professor
Kuban State Agricultural university, 350044, Russia, Krasnodar, Kalinina, 13

Mikhaylenko Evgeny Vladimirovich
Cand.Phys.-Math.Sci., associate professor,
Deputy Head of a Department
Krasnodar University of Ministry of the Interior of Russian Federation, Krasnodar, Russia

The article describes the technologies for preparing software systems in the Visual Basic for Application environment based on the office suites Microsoft Office, OpenOffice and LibreOffice for the automated preparation of individual sets of practical tasks in mathematical disciplines, as well as methods for verifying the work completed by students. It outlines the basic principles of development of the presented software products, describes the structure of typical software developments and the content of the software modules included in them, examines algorithms for generating various classes of problems in the branches of higher mathematics: set theory, number theory, matrix algebra, Boolean algebra and probability theory, provides fragments of codes of developed procedures. This work is aimed at teaching staff, software engineers of educational organizations

Keywords: ALGORITHM, BOOLEAN ALGEBRA, VERIFICATION, GENERATION, COMPUTER PROGRAM, MACRO, OPTIMIZATION METHODS, POLYNOMIAL, PRACTICAL TASK, SOFTWARE MODULE

Настоящая статья посвящена технологиям подготовки программных модулей для автоматизированной подготовки индивидуальных комплектов практических заданий, выполняемых обучаемыми в рамках практикума по дисциплинам математика и высшая математика, а также методам верификации выполненных работ. Указанные дисциплины включают такие разделы математики, как линейная алгебра и аналитическая геометрия, общая алгебра, математический анализ, теория вероятностей и математическая статистика, дискретная математика, методы оптимизации. По темам дисциплинам предусмотрены расчетные задания, которые выполняются курсантами на практических занятиях и в часы самостоятельной подготовки [12]. Описываемые программные продукты предназначены для избавления преподавателей от трудоемкой работы по подбору типовых задач и проверки их решений на бумажных носителях, кроме того, каждый обучающийся получает свой уникальный вариант заданий, сгенерированных по разработанным алгоритмам, что исключает списывание готовых уже рассчитанных решений курсантов друг у друга.

Представляемые комплексы программных продуктов реализованы в среде Microsoft Office, но также при необходимости могут использоваться в офисных приложениях OpenOffice и LibreOffice. Так, при переходе компьютерного центра университета на операционную систему Astra Linux преподаватели продолжали работать на Windows в MS Office, а одновременно с этим обучаемые использовали LibreOffice.

Суть описываемого метода заключается в создании шаблонов типовых расчетных заданий, размещенных на листах книг MS Excel, в которые после выполнения макросов будут добавлены сгенерированные числовые, строковые или графические компоненты заданий [2]. Преподаватель может сгенерировать условия заданий как для одного курсанта, так и одновременно для всей учебной группы. Подготовленные файлы со сгенерированными заданиями помещаются в доступное для

обучаемых облачное пространство или папку на сервере, имена файлов содержат номера тем и практических занятий, а также фамилии курсантов.

Обучающиеся, выполняя задания, помещают свои решения в поля для ввода числовых или текстовых ответов и предоставляют решенные задания с ответами для проверки. Теперь, получив файлы с решениями, преподавателю достаточно будет обработать их соответствующими макросами, которые автоматически откроют выполненные работы, проверят промежуточные и окончательные решения, а также, согласно установленным критериям, выставят оценки [1].

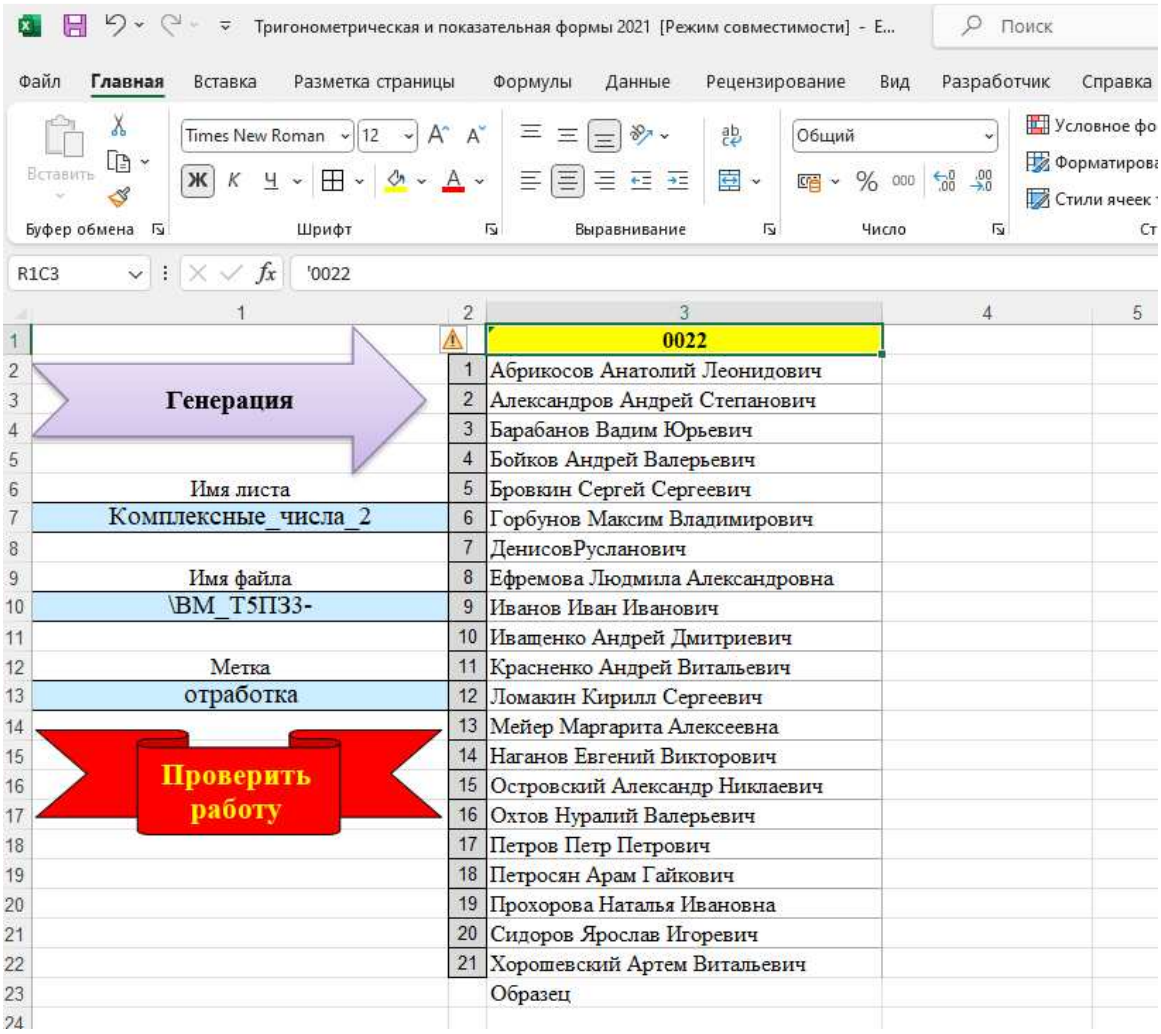
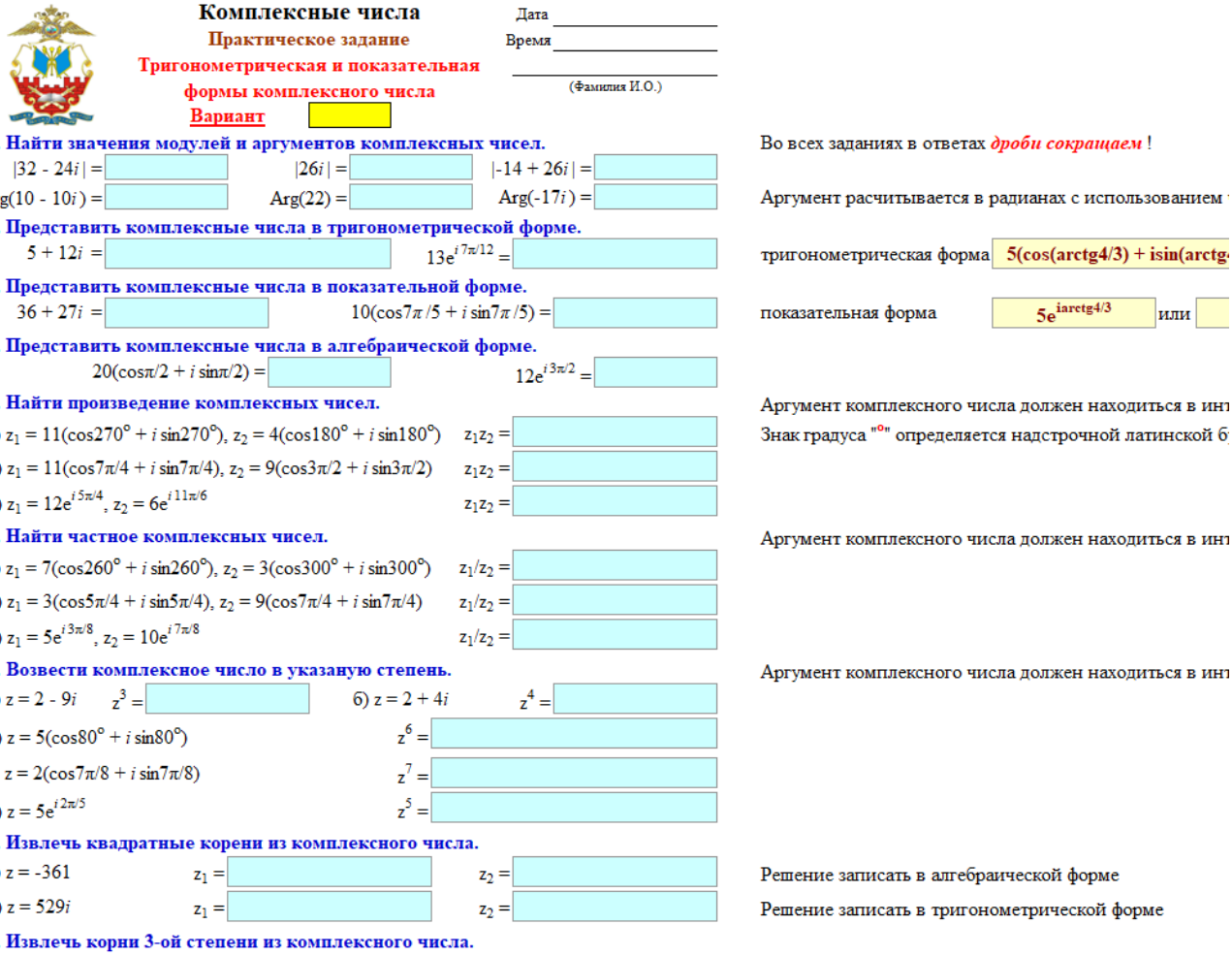


Рис. 1. Первый лист типового комплекса программ.

Рассмотрим структуру описываемых программных продуктов на примере комплекса программ «Тригонометрическая и показательная формы комплексного числа» по теме 5 «Теория множеств».

Каждый программный комплекс состоит из двух листов рабочей книги и комплекта макросов, позволяющих генерировать и проверять практические задания. Первый лист содержит списки курсантов, выполняющих данную работу, описание основных параметров заданий и элементы управления для генерирования и проверки работ (рис. 1). На лист можно поместить одну или несколько учебных групп, здесь преподаватели могут вызывать макросы генерации и проверки работ, указать наименование листа шаблона или изменить имена файлов с заданиями.

Второй лист является шаблоном практического задания (рис. 2).



Комплексные числа

Практическое задание

Тригонометрическая и показательная формы комплексного числа

Вариант

Дата _____

Время _____

_____

(Фамилия И.О.)

1. Найти значения модулей и аргументов комплексных чисел.

$|32 - 24i| =$ $|26i| =$ $|-14 + 26i| =$

$Arg(10 - 10i) =$ $Arg(22) =$ $Arg(-17i) =$

2. Представить комплексные числа в тригонометрической форме.

$5 + 12i =$ $13e^{i\,7\pi/12} =$

3. Представить комплексные числа в показательной форме.

$36 + 27i =$ $10(\cos 7\pi/5 + i\sin 7\pi/5) =$

4. Представить комплексные числа в алгебраической форме.

$20(\cos\pi/2 + i\sin\pi/2) =$ $12e^{i\,3\pi/2} =$

5. Найти произведение комплексных чисел.

а) $z_1 = 11(\cos 270^\circ + i\sin 270^\circ)$, $z_2 = 4(\cos 180^\circ + i\sin 180^\circ)$ $z_1z_2 =$

б) $z_1 = 11(\cos 7\pi/4 + i\sin 7\pi/4)$, $z_2 = 9(\cos 3\pi/2 + i\sin 3\pi/2)$ $z_1z_2 =$

в) $z_1 = 12e^{i\,5\pi/4}$, $z_2 = 6e^{i\,11\pi/6}$ $z_1z_2 =$

6. Найти частное комплексных чисел.

а) $z_1 = 7(\cos 260^\circ + i\sin 260^\circ)$, $z_2 = 3(\cos 300^\circ + i\sin 300^\circ)$ $z_1/z_2 =$

б) $z_1 = 3(\cos 5\pi/4 + i\sin 5\pi/4)$, $z_2 = 9(\cos 7\pi/4 + i\sin 7\pi/4)$ $z_1/z_2 =$

в) $z_1 = 5e^{i\,3\pi/8}$, $z_2 = 10e^{i\,7\pi/8}$ $z_1/z_2 =$

7. Возвести комплексное число в указанную степень.

а) $z = 2 - 9i$ $z^3 =$ б) $z = 2 + 4i$ $z^4 =$

в) $z = 5(\cos 80^\circ + i\sin 80^\circ)$ $z^6 =$

г) $z = 2(\cos 7\pi/8 + i\sin 7\pi/8)$ $z^7 =$

д) $z = 5e^{i\,2\pi/5}$ $z^5 =$

8. Извлечь квадратные корени из комплексного числа.

а) $z = -361$ $z_1 =$ $z_2 =$

б) $z = 529i$ $z_1 =$ $z_2 =$

9. Извлечь корни 3-ой степени из комплексного числа.

Во всех заданиях в ответах *дроби сокращаем* !

Аргумент расчитывается в радианах с использованием

тригонометрическая форма 5(cos(arctg4/3) + isin(arctg

показательная форма $5e^{i\arctg4/3}$ или

Аргумент комплексного числа должен находиться в инт

Знак градуса "°" определяется надстрочной латинской б

Аргумент комплексного числа должен находиться в инт

Аргумент комплексного числа должен находиться в инт

Решение записать в алгебраической форме

Решение записать в тригонометрической форме

Рис. 2. Верхняя часть шаблона практического задания Тригонометрическая и показательная формы комплексного числа.

Лист шаблона размечен так, чтобы на нем компактно размещались задания и поля для ответов. Шаблон может быть пустым и не содержать условий предлагаемых задач. В процессе обработки листа все задания будут вновь сгенерированы и помещены в нужные места.

Стандартный программный комплекс содержит также программные модули для генерирования заданий и проверки выполненных работ [5].

Модуль 1. Процедуры и функции разработчика.

Модуль включает процедуры и функции, использующиеся во всех выполняемых макросах Состав этого модуля пересекается с аналогичными модулями других программных комплексов, однако здесь находятся и специфические подпрограммы, разработанные специально для изучаемый темы.

Модуль 2. Запуск.

Модуль предназначен для копирования файлов с формой для размещения на нем заданий. В зависимости от положения активной ячейки листа «Генерация» создается один или группа новых файлов для размещения в них шаблона и заполнения формы сгенерированными данными.

```
If NoRow = 1 Then
        a = 1: b = N                    'Задания для всей группы
      Else
        a = NoRow - 1: b = NoRow - 1    'Задание для выбранного курсанта
End If
For i = a To b
    Sheets(list_name).Copy
    t = Name_Kur(i, 1)
    If Name_Kur(i, 2) <> "" Then t = t + " " + Mid(Name_Kur(i, 2), 1, 1) + "."
    If Name_Kur(i, 3) <> "" Then t = t + Mid(Name_Kur(i, 3), 1, 1) + "."
    Cells(3, 14) = t
    Call формы_комплексного_числа        'вызов генерации задания
    t = Name_Kur(i, 1) + Mid(Name_Kur(i, 2), 1, 1) + Mid(Name_Kur(i, 3), 1, 1)
    ActiveWorkbook.SaveAs Filename:=MyPath + file_name + NoGroup + t + " " + _
    metka + ".xls", FileFormat:=xlExcel8, Password:="", WriteResPassword:=""
    ActiveWindow.Close
Next i
```

Рис.3. Фрагмент кода макроса Запуск.

На рисунке 3 представлен фрагмент макроса Запуск, реализующий создание новых листов, помещения на них формы, вызов модуля генерации, определения имен файлов, сохранение файлов.

Модуль 3. Генерирование заданий.

Модуль содержит макросы, генерирующие условия практических заданий и заполнения скопированных шаблонов полученными данными. В макросе Формы_комплексного_числа с использованием системного времени и имени обучаемого рассчитывается вариант работы и осуществляется инициализация генератора псевдослучайных чисел (рис. 4). Всем заданиям работы соответствуют отдельные процедуры, вызываемые из макроса поочередно.

```
Sub Формы_комплексного_числа()
PAROL_1 = "GJPBNBD"
ActiveSheet.Protect Password:=PAROL_1, AllowFormattingCells:=True
ActiveSheet.EnableSelection = xlUnlockedCells
ActiveSheet.Unprotect Password:=PAROL_1
Name = Cells(3, 14)
Range("A1:Z7").Locked = True
t = Timer
Cells(2, 15) = sTimer(t)
s = Cells(2, 15)
Cells(1, 15) = Date
V = t Mod 31000 + codeName(Name)
Cells(6, 9) = V
Rnd (V * (-1))
Randomize (V)
Call Задание_1(V)
Call Задание_2(V)
Call Задание_3(V)
Call Задание_4(V)
Call Задание_5(V)
Call Задание_6(V)
Call Задание_7(V)
Call Задание_8(V)
Call Задание_9(V)
ActiveSheet.Protect AllowFormattingCells:=True, Password:=PAROL_1
End Sub
```

Рис. 4. Макрос Формы_комплексного_числа.

Модуль 4. Проверка выполненных работ.

Модуль включает ряд программ для поиска файлов с выполненными заданиями, повторной генерации заданий, проверки ответов и выставления оценок. Основной программой четвертого модуля является макрос Найти_файл (рис. 5). Выбор и открытие проверяемого файла с решениями организовано с помощью метода GetOpenFilename [7], затем вызывается организационный макрос проверки, файл сохраняется под новым именем, к имени добавляется полученная оценка, а проверяемый старый файл удаляется.

```
Private Sub Найти_файл()
MyPath = ThisWorkbook.path

fn = Application.GetOpenFilename _
    ("Excel files(*.xls*),*.xls*,All files(*.*),*.*", 1, _
     "Выберите Excel файлы для проверки", , False)

'MsgBox "Выбран файл: '" & fn & "'", vbInformation, "Заголовок окна"
Workbooks.Open Filename:=fn

Call PR_Complex_Number_Forms(оценка)

k = Len(fn)
file_name = Mid(fn, 1, k - 4) & " " & оценка & ".xls"

ActiveWorkbook.SaveAs Filename:=file_name, _
        FileFormat:=xlExcel8, Password:="", WriteResPassword:="", _
        ReadOnlyRecommended:=False, CreateBackup:=False
Kill fn
End Sub
```

Рис. 5. Код макроса Найти_файл.

Макрос проверки PR_Complex_Number_Forms предназначен для вызова процедур проверки (рис. 6). Здесь производится верификация работы: проверяются на соответствие номер варианта, дата и время генерации работы и идентификатора пользователя. Использование функции Randomize [10] обязательно для получения такой же, как и при генерации, последовательности параметров заданий. Здесь, как и в макросе Формы_комплексного_числа, в том же порядке поочередно используются алгоритмы генерации параметров заданий, далее по ним рассчитываются

решения задач для сравнения с введенными ответами курсантов. В завершении проверки рассчитывается оценка.

```
Sub PR_Complex_Number_Forms(оценка)
PAROL_1 = "GJPBNBD"
ActiveSheet.Protect Password:=PAROL_1, AllowFormattingCells:=True
ActiveSheet.EnableSelection = xlUnlockedCells
ActiveSheet.Unprotect Password:=PAROL_1
V = Cells(6, 9)
Name = Cells(3, 14)
stime = Cells(2, 15)
Name1000 = V - (Val(Mid(stime, 2, 2)) * 3600 + Val(Mid(stime, 5, 2)) * 60 _
    + Val(Mid(stime, 8, 2))) Mod 31000
If Name1000 <> codeName(Name) Then
  Cells(6, 12) = "Неверный код фамилии!!!"
  ActiveSheet.Protect Password:=PAROL_1
  End
End If
Rnd (V * (-1)): Randomize (V): mark = 0
Call PR_Задание_1(V, mark)
Call PR_Задание_2(V, mark)
Call PR_Задание_3(V, mark)
Call PR_Задание_4(V, mark)
Call PR_Задание_5(V, mark)
Call PR_Задание_6(V, mark)
Call PR_Задание_7(V, mark)
Call PR_Задание_8(V, mark)
Call PR_Задание_9(V, mark)
Cells(58, 15) = mark
Select Case mark
  Case 30 To 33:    оценка = "отлично"
  Case 24 To 29:    оценка = "хорошо"
  Case 18 To 23:    оценка = "удовлетворительно"
  Case Else:    оценка = "неудовлетворительно"
End Select
Cells(59, 13) = оценка
End Sub
```

Рис. 6. Код макроса PR_Complex_Number_Forms.

Рассмотрим технологии генерации и проверки работ на примере задания «9. Извлечь корни 3-ой степени из комплексного числа» рассматриваемого программного комплекса (рис.7).

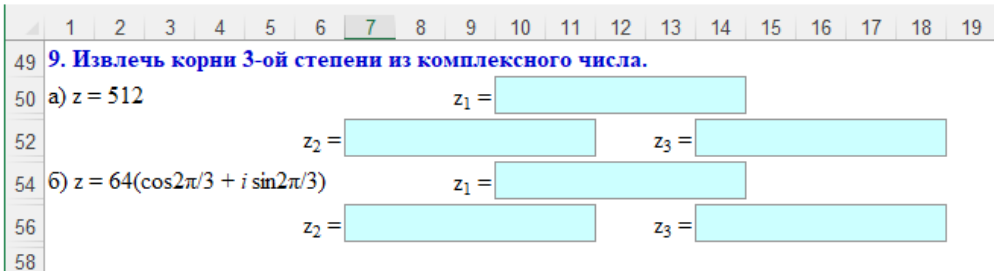


Рис. 7. Сгенерированное задание «9. Извлечь корни 3-ой степени из комплексного числа».

В задании 9.а с помощью генератора псевдослучайных чисел генерируется только одно целочисленное значение *x*. Затем формируется значение строковой переменной t, в которое входит *x* уже в третьей степени. И, наконец, ячейке, находящейся на пересечении 50-й строки и первого столбца присваивается значение сгенерированного выражения.

```
Sub Задание_9(V)
'9. Извлечь корни 3-ой степени из комплексного числа.
Pi = ChrW((960)) ' ПИ
'а
x = Int(Rnd(V) * 6 + 4)
t = "а) z = " + st(x ^ 3)
Cells(50, 1) = t
'б
Do
  r = (Int(Rnd(V) * 4 + 2)) ^ 3
  f1 = Int(Rnd(V) * 7 + 2)
  f2 = Int(Rnd(V) * 4 + 2): If f2 = 5 Then f2 = 6
Loop Until НОД(f1, f2) = 1 And f1 / f2 < 2 And НОД(f1, 3) = 1
t = "б) z = " + st(r) + "(cos" + st(f1) + Pi + "/" + st(f2) + " + i"
k = Len(t)
t = t + "sin" + st(f1) + Pi + "/" + st(f2) + ")"
Cells(54, 1) = t
Cells(54, 1).Font.FontStyle = "обычный"
Cells(54, 1).Characters(Start:=k, Length:=1).Font.FontStyle = "курсив"
End Sub
```

Рис. 8. Код макроса Задание_9.

Особенность задания 9.б состоит в том, что генерируемое число представляется в тригонометрической форме и для его вывода потребуется рассчитать модуль *r*, а также параметры аргумента f_1 и f_2. Кроме того, число *i* выделяется курсивом.

При проверке задания (рис. 9) не нужно анализировать находящееся на листе задание, это может быть ненадежно и трудоемко. Для получения значений переменных x, r, f_1 и f_2. достаточно еще раз использовать алгоритмы, используемые при генерации задания. Решения рассматриваемых заданий можно найти при помощи формулы Муавра, а затем сравнить их с ответами на листе, предварительно убрав из них пробелы. Если решение курсанта совпало с расчетным, то количество набранных баллов увеличивается на единицу, в противном случае проверяемая область перечеркивается.

```
Sub PR_Задание_9(V, mark)
'9. Извлечь корни 3-ой степени из комплексного числа.
Pi = ChrW((960)) ' ПИ
'а Проверка
x = Int(Rnd(V) * 6 + 4)
tr1 = st(x)
If tr1 = No_Space(Cells(50, 10)) Then mark = mark + 1 Else Call krest(50, 10)
tr2 = st(x) + "ei2" + Pi + "/3"
If tr2 = No_Space(Cells(52, 7)) Then mark = mark + 1 Else Call krest(52, 7)
tr3 = st(x) + "ei4" + Pi + "/3"
If tr3 = No_Space(Cells(52, 14)) Then mark = mark + 1 Else Call krest(52, 14)
'б Проверка
Do
  r = (Int(Rnd(V) * 4 + 2)) ^ 3
  f1 = Int(Rnd(V) * 7 + 2)
  f2 = Int(Rnd(V) * 4 + 2): If f2 = 5 Then f2 = 6
Loop Until НОД(f1, f2) = 1 And f1 / f2 < 2 And НОД(f1, 3) = 1
r = r ^ (1 / 3)
tr1 = st(r) + "(cos" + st(f1) + Pi + "/" + st(f2 * 3) + _
    "+isin" + st(f1) + Pi + "/" + st(f2 * 3) + ")"
If tr1 = No_Space(Cells(54, 10)) Then mark = mark + 1 Else Call krest(54, 10)
tr2 = st(r) + "(cos" + st(f1 + 2 * f2) + Pi + "/" + st(f2 * 3) + _
    "+isin" + st(f1 + 2 * f2) + Pi + "/" + st(f2 * 3) + ")"
If tr2 = No_Space(Cells(56, 7)) Then mark = mark + 1 Else Call krest(56, 7)
tr3 = st(r) + "(cos" + st(f1 + 4 * f2) + Pi + "/" + st(f2 * 3) + _
    "+isin" + st(f1 + 4 * f2) + Pi + "/" + st(f2 * 3) + ")"
If tr3 = No_Space(Cells(56, 14)) Then mark = mark + 1 Else Call krest(56, 14)
End Sub
```

Рис. 9. Код макроса проверки задания.

На рисунке 10 показан результат проверки задания. Пять заданий выполнены верно, за них обучаемый получил 5 баллов, неверное выполненное задание перечеркнуто.

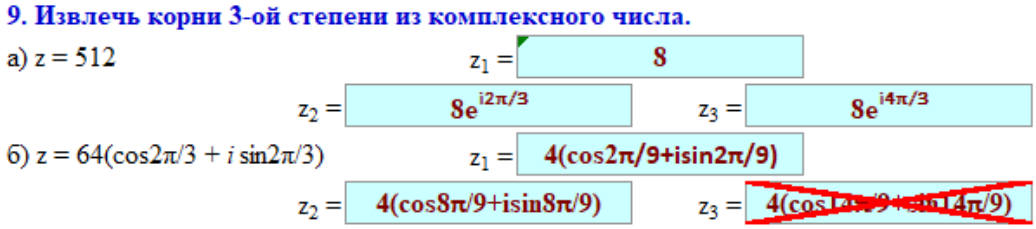


Рис. 10. Результат проверки задания.

Сейчас мы рассмотрели достаточно простой пример генерирования и проверки заданий. На практике часто бывает недостаточно производить операции с отдельными переменными, в этом случае все действия выполняются с использованием массивов различной размерности.

Рассмотрим такие методы на примере программного комплекса «Операции над многочленами» по теме «Многочлены». На рисунке 11 представлено сгенерированное задание «1. Вычислить сумму многочленов».

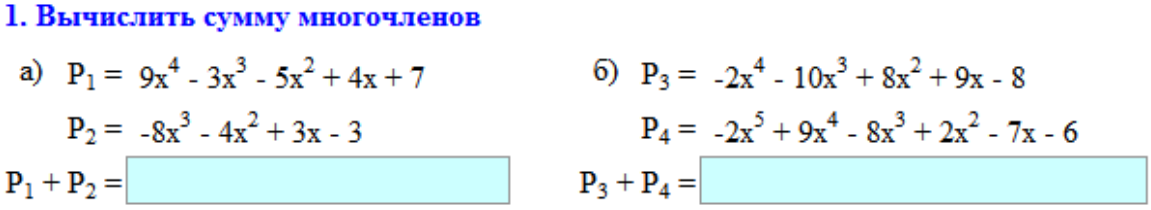


Рис. 11. Сгенерированное задание «1. Вычислить сумму многочленов».

Для генерации коэффициентов полиномов P_1 и P_2 четвертой и третьей степени соответственно (рис. 12) используем циклы, в которых рассчитываемые коэффициенты многочленов принимают значения в интервале от -10 до 10. С использованием нестандартной функции *pol* многочлены выводятся на лист [6].

```
' Задание 1
' Вычислить сумму многочленов
Erase P1
For i = 0 To 4
  P1(i) = Rnd(v) * 20 - 10
Next i
pol1 = pol(P1, 10, 3)
Erase P2
For i = 0 To 3
  P2(i) = Rnd(v) * 20 - 10
Next i
pol2 = pol(P2, 12, 3)
```

Рис. 12. Программный код задания «Вычислить сумму многочленов».

Вызываемая в программе функция *pol* формирует строку, включающую коэффициенты многочлена, обозначения переменных, степени переменных и арифметические знаки «+» либо «–». Местоположение переменных в записи фиксируется в массиве *ms*, и после вывода полинома на лист определяется надстрочный шрифт для степеней переменных.

При проверке решения данного задания процедура генерации коэффициентов многочленов P_1 и P_2 остается без изменений (рис. 13). С помощью функции *polsum* осуществляется сумма полиномов *PR = P1 + P2*. В строковую переменную *tz* считывается ответ обучаемого и из нее удаляются пробелы. С помощью функций *poltext* и *ubr_prob* происходит преобразование найденного решения. Затем происходит сравнение рассчитанного решения *t* и считанного *tz*. В случае равенства значений *t* и *tz* к количеству набранных баллов *mark* добавляется балл, в противном случае поле с неверным решением зачеркивается.

```
' Задание 1 (Проверка)
' Вычислить сумму многочленов
Erase P1
For i = 0 To 4
  P1(i) = Rnd(v) * 20 - 10
Next i
Erase P2
For i = 0 To 3
  P2(i) = Rnd(v) * 20 - 10
Next i
p = polsum(P1, P2, Pr)
tz = Cells(14, 3)
tz = ubr_prob(tz)
t = poltext(Pr)
t = ubr_prob(t)
If t = tz Then mark = mark + 1 Else Cells(14, 3) = krest(14, 3)
```

Рис. 13. Фрагмент кода проверки задания «Вычислить сумму многочленов.

Часто для подготовки заданий следует идти от обратного: сначала необходимо сгенерировать решение задачи, а затем на основе готового решения формировать условие задачи. Рассмотрим процесс генерации и проверки задания 7.б «Разложить многочлены на множители» (рис. 14). Суть задания заключается в том, чтобы при разложении полинома *n*-й степени все *n* множителей-одночленов имели целые коэффициенты. Случайным подбором коэффициентов разлагаемого полинома здесь не обойтись, следовательно, их нужно специальным образом рассчитать [9].

7. Разложить многочлены на множители (знаки умножения не ставить)

а) $11x^2 + 176x + 308$

б) $7x^3 + 140x^2 + 763x + 630$

в) $x^4 - 7x^3 - 43x^2 + 175x + 450$

Рис. 14. Задание «Разложить многочлены на множители».

Для формирования выводимого полинома (рис. 15) после очистки массивов *P1*, *P2*, *Pr* и *R* в цикле *Do .. Loop Until* [3] сгенерируем все его корни: x_1, x_2, x_3, а также коэффициент *a*. Осуществим проверку на то, чтобы кони не повторялись. Воспользуемся теоремой Виета для расчета коэффициентов искомого полинома. В случае многочлена третьей степени $ax^3 + bx^2 + cx + d$ его коэффициенты можно найти из произведения

одночленов на старший коэффициент: $a(x-x_1)(x-x_2)(x-x_3)$. Результирующий полином *Pr* выводится на лист формы заданий.

```
'Задание 7.б. Разложить на множители
Erase P1: Erase P2
Erase Pr: Erase R
Do
  x1 = Rnd(v) * 20 - 15      '1-й корень
  x2 = x1 + Rnd(v) * 10 + 1 '2-й корень
  x3 = x2 + Rnd(v) * 10 + 1 '3-й корень
  a = (-1) ^ v * Rnd(v) * 10 + 2
Loop Until x1 * x2 * x3 <> 0 And Abs(x1) <> Abs(x2) _
                             And Abs(a) > 1 And x3 < 20

P1(1) = 1: P1(0) = -x1: R(0) = a
p = polmult(P1, R, P2)
R(1) = 1: R(0) = -x2
p = polmult(P2, R, P1)
R(1) = 1: R(0) = -x3
p = polmult(P1, R, Pr)
polr = pol(Pr, 58, 2)
```

Рис. 15. Программный код генерации задания 7.б.

Произведение многочленов производится последовательно с помощью разработанной функции *polmult* (рис. 16). Функция предназначена для умножения двух многочленов. Степень множителей ограничена числом 15 только из потребностей обучения и ее легко можно увеличить. Итерационный процесс осуществляется в двух вложенных циклах, где параметры внешнего цикла *i* и внутреннего цикла *j* являются степенями мономов, входящих в первый и второй многочлены соответственно.

```
Function polmult(P1, P2, pm)
Erase pm
For i = 0 To 15
  For j = 0 To 15
    pm(i + j) = pm(i + j) + P1(i) * P2(j)
  Next j
Next i
End Function
```

Рис. 16. Функция произведения полиномов *polmult*.

Алгоритм расчета параметров условия задания во время проверки ничем не отличается от того, что был во время генерации (рис. 17). В вычислении коэффициентов полинома *b*, *c* и *d* нет потребности, однако

зная корни многочлена x_1, x_2, x_3 и коэффициент a, по теореме Виета сформируем строку произведений вида $a(x - x_1)(x - x_2)(x - x_3)$, и сравним ее с решением курсанта.

```
'Задание 7.б. Разложить на множители
Erase P1: Erase P2          'Проверка
Erase Pr: Erase r
Do
  x1 = Rnd(v) * 20 - 15     '1-й множитель
  x2 = x1 + Rnd(v) * 10 + 1 '2-й множитель
  x3 = x2 + Rnd(v) * 10 + 1 '3-й множитель
  a = (-1) ^ v * Rnd(v) * 10 + 2
Loop Until x1 * x2 * x3 <> 0 And Abs(x1) <> Abs(x2) _
                             And Abs(a) > 1 And x3 < 20
t = st(a) + "(x "
If x1 < 0 Then
        t = t + " + " + st(Abs(x1)) + ")(x"
    Else
        t = t + " - " + st(Abs(x1)) + ")(x"
End If
If x2 < 0 Then
        t = t + " + " + st(Abs(x2)) + ")(x"
    Else
        t = t + " - " + st(Abs(x2)) + ")(x"
End If
If x3 < 0 Then
        t = t + " + " + st(Abs(x3)) + ")"
    Else
        t = t + " - " + st(Abs(x3)) + ")"
End If
tz = Cells(58, 10)
tz = ubr_prob(tz): t = ubr_prob(t)
If t = tz Then mark = mark + 1 Else Cells(58, 10) = krest(58, 10)
```

Рис. 17. Программный код проверки задания «Разложить на множители.

В работе с элементами булевой алгебры специфичными являются элементарный математический аппарат, ограниченное, заранее рассчитываемое, количество значений булевых функций, зависящее только от количества аргументов, широкое разнообразие методов аналитического представления булевых выражений, алгоритмов выполнения алгебраических операций над ними.

При задании булевых функций будем иметь в виду тот факт, что аналитическое представление любой функции состоит из выражений, представляющих собой конъюнкции некоторых аргументов. Для унификации процесса будем считать, что это конъюнкции всех n аргументов функции, каждый их которых представлен в одном из трех

состояний: прямой, инверсной форме или отсутствует [11]. Причем необходимо, что хотя бы одна из n переменных присутствовала в конъюнкции. Тогда при генерации таких конъюнкций для n-арной функции мы получим $3^n - 1$ вариантов.

Например, в задании 1.б программного модуля «ДНФ, КНФ булевых функций» (рис. 18) функция трех переменных f_2 содержит четыре компонента: $\overline{A}\overline{B}\overline{C}$, $\overline{A}\overline{C}$, $\overline{A}$, AB. В первом выражении все три переменных A, B и C присутствуют в инверсной форме, во втором – инверсии A и C, а переменная B имеет статус отсутствия, в третьем – только инверсия A, а в четвертом – A, B в прямой форме, а C отсутствует.

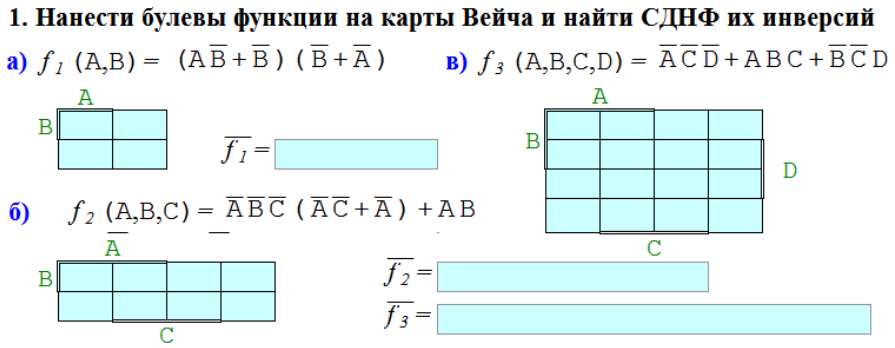


Рис. 18. Первое задание модуля «ДНФ, КНФ булевых функций».

Приведем фрагмент программного кода описываемого процесса (рис. 19).

```
'1.б. Функция f2(a, b, c) = ABC ( ABC  + ABC ) + ABC
                    ' Определение параметров булевых переменных
Do
  Do
  k = 0
    For i = 1 To 4
      For j = 1 To 3
        ab(i, j) = Rnd(v) * 100 Mod 3
      Next j
      If ab(i, 1) + ab(i, 2) + ab(i, 3) = 6 Then k = 1
    Next i
    If ab(1, 1) = ab(2, 1) And ab(1, 2) = ab(2, 2) _
       And ab(1, 3) = ab(2, 3) Then k = 1
    If ab(1, 1) = ab(3, 1) And ab(1, 2) = ab(3, 2) _
       And ab(1, 3) = ab(3, 3) Then k = 1
    If ab(1, 1) = ab(4, 1) And ab(1, 2) = ab(4, 2) _
       And ab(1, 3) = ab(4, 3) Then k = 1
    If ab(2, 1) = ab(3, 1) And ab(2, 2) = ab(3, 2) And _
       ab(2, 3) = ab(3, 3) Then k = 1
  Loop Until k = 0
```

Рис. 19. Фрагмент кода определения статусов булевых переменных.

Для статусов переменных в программе определены следующие значения: 0 – «инверсия», 1 – «прямая форма», 2 – «отсутствие переменной». Во вложенных циклах с параметрами *i* и *j* происходит определение статусов трех булевых переменных, определяемых значениями элементов массива $ab(i, j)$ в каждом из четырех выражений. Здесь же осуществляется проверка на присутствие в конъюнкции хотя бы одной переменной. Затем происходит сравнение полученных конъюнкций. Описанный процесс будет повторяться до тех пор, пока все четыре сгенерированных выражения не будут различными.

При нанесении функций на лист MS Excel возникли проблемы с выводом инверсий булевых переменных. Найти стандартные шрифты с надчеркиванием всех заглавных латинских букв не удалось. Решением проблемы стал вывод булевых выражений не в одной, а одновременно в двух строках: в нижнюю строку заносится выражение без инверсий, а в верхнюю – только символы подчеркивания. Представим программный код вывода на лист первой конъюнкции (рис. 20).

```
                                    'Вывод функции
t = "":: t0 = ""
If ab(1, 1) = 0 Then t = t + "A ": t0 = t0 + "_ "
If ab(1, 1) = 1 Then t = t + "A ": t0 = t0 + "  "
If ab(1, 2) = 0 Then t = t + "B ": t0 = t0 + "_ "
If ab(1, 2) = 1 Then t = t + "B ": t0 = t0 + "  "
If ab(1, 3) = 0 Then t = t + "C ": t0 = t0 + "_ "
If ab(1, 3) = 1 Then t = t + "C ": t0 = t0 + "  "
t = t + "( ": t0 = t0 + "  "
```

Рис. 20. Вывод первой конъюнкции булевой функции.

Сформированные строковые переменные *t0* и *t* заносятся в соответствующие места на листе друг под другом, а затем параллельно преобразуется шрифт в обоих выражениях таким образом, чтобы размер символов на позициях, соответствующих пробелам в строке *t*, уменьшился до трех единиц (рис. 21).

```
n = Len(t0)

Cells(14, 5) = t0: Cells(14, 5).Font.Size = 13
Cells(15, 5) = t: Cells(15, 5).Font.Size = 13

For i = 1 To n - 1  'Уменьшаем шрифт всавленных пробелов
  If Mid(t, i, 1) = " " Then
    Cells(14, 5).Characters(Start:=i, Length:=1).Font.Size = 3
    Cells(15, 5).Characters(Start:=i, Length:=1).Font.Size = 3
  End If
Next i
```

Рис. 21. Изменение шрифтов сформированных записей.

Перейдем теперь к проверке описанного задания 1.б. Сначала проверим правильность нанесения булевой функции на карту Вейча. Для этого воспользуемся значениями таблицы истинности, рассчитанными при генерации параметров функции.

Строковые переменные *tr* и *tc* (рис. 22) определяют координаты заполнения таблицы минтермами $m_0 = \overline{A}\overline{B}\overline{C}$, $m_1 = \overline{A}\overline{B}C$, …, $m_7 = ABC$. После преобразования координат массив значений таблицы истинности сравниваются с ответами обучаемого. В случае правильного ответа промежуточная оценка *m1* увеличивается на единицу, если ответ неверный – зачеркивается проверяемое поле.

```
                                'Заполнение карты Вейча
'     0 1 2 3 4 5 6 7
tr = "2,2,1,1,2,2,1,1"
tc = "4,3,4,3,1,2,1,2"
For i = 0 To 7
  If rf(i) = True Then rr = 1 Else rr = ""
  If Cells(Val(Mid(tr, i * 2 + 1)) + 16, _
           Val(Mid(tc, i * 2 + 1)) + 1) = rr Then
     m1 = m1 + 1
    Else
      Call krest(Val(Mid(tr, i * 2 + 1)) + 16, _
                 Val(Mid(tc, i * 2 + 1)) + 1)
  End If
Next
```

Рис. 22. Фрагмент программного кода проверки заполнения карты Вейча.

Для проверки записи СДНФ инверсии функции (рис. 23.) помещаем в переменную *t* упорядоченные номера наборов, на которых функция принимает нулевые значения, а затем сравниваем *t* с ответом *w*, предварительно вырезав из значений обеих строковых переменных все символы, кроме цифр. Задание считается выполненным, если верно заполнены все девять полей.

```
                                'Вывод СДНФ инверсии f1
t = "("
For i = 0 To 7
  If rf(i) = False Then t = t + st(i) + "; "
Next
t = Mid(t, 1, Len(t) - 2) + ")"
t = OnlyNumer(t)
w = OnlyNumer(Cells(17, 9))
If t = w Then m1 = m1 + 1 Else Call krest(17, 9)
If m1 = 9 Then mark = mark + 1
```

Рис. 23. Проверка записи инверсии булевой функции.

Во многих разделах общематематических дисциплин используются матрицы и определители. Особенность обработки матричных выражений заключается в том, что объектами вычислений являются не отдельные числа, а двумерные числовые массивы, и порой разработчику необходимо приложить немало труда для составления специфических алгоритмов их обработки [4].

Рассмотрим пример генерации задания «2.в. Выполнить действия над матрицами» (рис. 24) из программного комплекса «Обратная матрица. Ранг

матрицы» по теме «Матрицы и определители». При выполнении задания обучаемый сначала должен найти обратную матрицу для второго множителя, а затем умножить первую матрицу на полученный результат. Но генерировать задание в таком порядке не следует, так как в процессе нахождения обратной матрицы могут получаться элементы с дробными значениями, что затруднит выполнение работы курсантами [8].

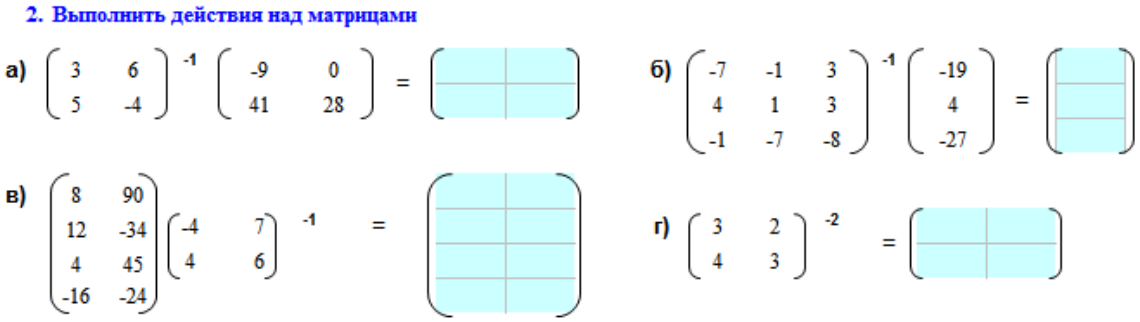


Рис. 24. Задание «2. Выполнить действия над матрицами».

Формируемое задание 2.в можно представить в матричной форме следующим образом: $A \cdot B^{-1} = C$. В приводимом примере матрицы A и B выведены на лист, а значения коэффициентов матрицы C нужно внести. Если в матричном выражении перенести второй множитель в правую часть, то получим $A = \frac{C}{B^{-1}}$ или $A = C \cdot B$. Учитывая данный факт рассчитывать коэффициенты матриц будем в следующем порядке: начнем с матрицы C, затем, генерируя произвольные целочисленные коэффициенты, сформируем матрицу B, а уж потом рассчитаем матрицу A.

В представленном фрагменте кода программы (рис. 25) мы видим, что во вложенных циклах с параметрами i и j генерируются коэффициенты двумерного массива c, являющегося решением задачи. При формировании матрицы b, для которой обучаемый будет строить обратную, в цикле Do .. Loop Until происходит проверка матрицы на вырожденность. Массив a получим, перемножая уже сформированные элементы массивов c и b. При разработке алгоритма расчета коэффициентов матрицы A необходимо

использовать определение матричного произведения и учитывать некоммутативность данной операции.

```
'Задание 2.в. Выполнить действия над матрицами.
' Матрица 4x2 умножается на обратную матрицу 2x2

For i = 1 To 4
  For j = 1 To 2
    c(i, j) = Rnd(v) * 18 - 9
Next j, i

Do
  For i = 1 To 2
    For j = 1 To 2
      b(i, j) = Rnd(v) * 16 - 8
      Cells(i + 24, j + 3) = b(i, j)
  Next j, i
Loop Until (b(1, 1) * b(2, 2) - b(1, 2) * b(2, 1)) <> 0


For i = 1 To 4
  For j = 1 To 2
    a(i, j) = 0
    For k = 1 To 3
      a(i, j) = a(i, j) + c(i, k) * b(k, j)
    Next k
    Cells(i + 23, j + 1) = a(i, j)
Next j, i
```

Рис. 25. Генерация задания «Выполнить действия над матрицами».

При проверке выполненного задания выполненного задания следует иметь ввиду, что в ней необходимо осуществить расчет всех параметров задания точно также, как и при генерации задачи обучаемым (рис. 26). В первом блоке представленного программного кода осуществляется вычисление значение элементов массива *c* и сравнение этих значений с записанным на листе ответом. Расчет значений массива *b* выполняется только для того, чтобы количество обращений к функции *Rnd* с параметром *v* совпадало с генерацией этого задания курсантом. На самом деле, значения массива *b* можно было и не рассчитывать, а заменить рассматриваемую структуру восьмикратным вызовом инструкции *Rnd*(*v*).

```
'Задание 2.в. Выполнить действия над матрицами.
' Матрица 4x2 умножается на обратную матрицу 2x2
                                                        'Проверка
For i = 1 To 4
  For j = 1 To 2
    c(i, j) = Rnd(v) * 18 - 9
    If Cells(i + 23, j + 7) = c(i, j) And Cells(i + 23, j + 7) <> "" Then
        mark = mark + 1
      Else
        Cells(i + 23, j + 7) = krest(i + 23, j + 7)
      End If
Next j, i

For i = 1 To 2
  For j = 1 To 2
    b(i, j) = Rnd(v) * 18 - 9
Next j, i
```

Рис. 26. Код проверки задания «Выполнить действия над матрицами».

Во многих разделах математики решаются типовые текстовые задачи. Особенность генерирования текстовых заданий заключается в совместной обработке числовых параметров и строчных значений. На рисунке 27 представлено задание «1. Решить задачи (Формула Бернулли)», в котором представлены пять текстовых задач из практической работы «Схемы повторных испытаний» (приложение 1) раздела «Теория вероятности».

1. Решить задачи (Формула Бернулли)

а) Какова вероятность того, что из 9 наблюдений событие A произойдет ровно 4 раза, если вероятность наступления события A при каждом испытании равна 0,4.

б) Определить вероятность того, что при 7 бросках игрального кубика "6" выпадет 5 раз.

в) Вероятность выигрыша при опускании жетона в игральный автомат равна 0,4. Найти вероятность того, что из 8 попыток игрок выиграет 5 раз.

г) Опыт состоит в перемешивании колоды из 36 карт и открывании верхней карты, после чего карта возвращается в колоду. Определить вероятность того, что а) из 9 опытов 7 раз откроются туз, король, дама или валет; б) из 9 опытов ни разу не откроются ни туз, ни король, ни дама, ни валет.

д) Расчитать вероятность поражения дальнобойным орудием находящейся на предельном расстоянии цели а) не более 7 раз, б) не менее 5 и не более 7 раз, если орудие произвело 9 выстрелов, а вероятность поражения цели при каждом выстеле равна 0,3.

Рис. 27. Задание «Решить задачи (Формула Бернулли)».

Рассмотрим процесс генерации задания 1.д (рис. 28). В представленном линейном алгоритме рассчитываются всего два параметра: в интервале от 7 до 11 целочисленный *n*, определяющий количество выстрелов дальнобойного орудия, и от 0,1 до 0,4 вещественный *p*, отвечающий за вероятность поражения цели при каждом выстреле. Затем вычисляются параметры задачи *m*1, *m*2, и с использованием *n*, *p*, *m*1, *m*2 осуществляется формирование значение строковой переменной *t*. В программе в определенном порядке складываются фрагменты текста задачи и преобразованные функцией *st* числовые значения, а затем значение *t* выводится на лист.

```
' Задание 1.д
' Расчитать вероятность поражения находящейся на предельном расстоянии цели
' а)не более m2 раз,б) от m1 до m2 раз дальнобойным орудием из n выстрелов,
' если вероятность поражения цели при каждом выстеле равна p.
n = v Mod 5 + 7
p = (v Mod 3 + 1) / 10
m2 = n - 2
m1 = m2 - 2
t = "Расчитать вероятность поражения дальнобойным орудием "
t = t + "находящейся на предельном расстоянии цели а) не более "
t = t + st(m2) + " раз, б) не менее " + st(m1) + " и не более "
t = t + st(m2) + " раз, если орудие произвело " + st(n) + " выстрелов, "
t = t + "а вероятность поражения цели при каждом выстеле равна " + st(p) + "."
Cells(19, 2) = t
```

Рис. 28. Генерация текстовой задачи.

Для проверки решений текстовых задач не требуется формировать строку условия (рис. 29). Здесь достаточно сгенерировать или рассчитать основные параметры задачи: *n*, *p*, *m*1 и *m*2. Учитывая полученные значения по теореме Бернулли, с использованием основных свойств вероятностей, получаем решения *pa* и *pb* и сравниваем их с ответами обучающегося. Так как в поле ответов находятся приближенные числовые значения, то допускаем погрешность 0,0001.

```
' Задание 1.д
n = v Mod 5 + 7                  'Проверка
p = (v Mod 3 + 1) / 10
m2 = n - 2
m1 = m2 - 2
' а) не более m2 раз
pa = 1 - soch(n - 1, n) * p ^ (n - 1) * (1 - p) ^ 1 _
       - soch(n, n) * p ^ (n + 1) * (1 - p) ^ 0
If Cells(19, 16) >= pa - 0.0001 And Cells(19, 16) <= pa + 0.0001 _
   And Cells(19, 16) <> "" Then mark = mark + 1 Else Cells(19, 16) = krest(19, 16)
' б) от m1 до m2 раз
pb = soch(m1, n) * p ^ m1 * (1 - p) ^ (n - m1) _
   + soch(m1 + 1, n) * p ^ (m1 + 1) * (1 - p) ^ (n - m1 - 1) _
   + soch(m2, n) * p ^ m2 * (1 - p) ^ (n - m2)
If Cells(21, 16) >= pb - 0.0001 And Cells(21, 16) <= pb + 0.0001 _
   And Cells(21, 16) <> "" Then mark = mark + 1 Else Cells(21, 16) = krest(21, 16)
```

Рис. 29. Проверка текстовой задачи.

Рассмотрим пример разработки приложения, включающих более сложные алгоритмы. В десятом задании программного комплекса «Графы. Связность и расстояния» по теме «Графы» требуется построить матрицы смежности, сильной связности и достижимости для представленного графа G (рис. 30).

10. Для ориентированного графа G построить матрицу смежности M, матрицу сильной связности W и матрицу достижимости T.

G= *[{g,h,i,j,k},{(g,j),(g,k),(h,g),(h,j),(i,g),(i,j),(j,g),(j,j),(k,k),(k,k)}]*

M= W= T=

Рис. 30. Задание 10 программного комплекса «Графы. Связность и расстояния».

Для определения графа G достаточно сформировать его матрицу смежности GM (рис. 31). Во вложенных циклах с параметрами *i* и j каждому элементу матрицы GM присваиваем одно из значений 0, 1 или 2. Здесь же рассчитываем кратность дуг графа.

Тело цикла в инструкции Do .. Loop Until будет выполняться до тех пор, пока количество направленных ребер без учета кратности будет от восьми до двенадцати и появится хотя бы одна дуга кратности более

единицы. Затем в полученной матрице обнуляем произвольно выбранный столбец.

Построенную матрицу смежности нам предстоит использовать для проверки значений в ответах.

```
A(1) = "g": A(2) = "h": A(3) = "i": A(4) = "j": A(5) = "k"
Do
  Do                                  'Генерируем матрицу смежности GM
    Erase GM
    r = 0: r2 = 0
    For i = 1 To 5
      For j = 1 To 5
        p = Int(Rnd(V) * 100) Mod 3
          GM(i, j) = p
          If p > 0 Then r = r + 1
          If p = 2 Then r2 = r2 + 1
    Next j, i
  Loop Until r >= 8 And r <= 12 And r2 <= 2
  k = Int(Rnd(V) * 100) Mod 5 + 1
    For i = 1 To 5                     'Обнуляем элементы столбца
    GM(i, k) = 0
  Next i
```

Рис. 31. Код генерации матрицы смежности.

Код построения матрицы достижимости представлен на рисунке 32.

```
Erase GSS                              'Формируем матрицу достижимости графа GM
For i = 1 To 5
  GS(i, i) = 1
Next i
For i = 1 To 5                         'Связаны одним ребром
  For j = 1 To 5
    If GM(i, j) > 0 Then GS(i, j) = 1
  Next j
Next i
For i = 1 To 5                         'Связаны двумя ребрами
  For j = 1 To 5
    For k = 1 To 5
      If GM(i, j) > 0 And GM(j, k) > 0 Then GS(i, k) = 1
    Next k
  Next j
Next i
For i = 1 To 5                         'Связаны тремя ребрами
  For j = 1 To 5
    For k = 1 To 5
      For L = 1 To 5
        If GM(i, j) > 0 And GM(j, k) > 0 And GM(k, L) > 0 Then GS(i, L) = 1
      Next L
    Next k
  Next j
Next i
For i = 1 To 5                         'Связаны четырьмя ребрами
  For j = 1 To 5
    For k = 1 To 5
      For L = 1 To 5
        For M = 1 To 5
          If GM(i, j) > 0 And GM(j, k) > 0 And GM(k, L) > 0 _
             And GM(L, M) > 0 Then GS(i, M) = 1
        Next M
      Next L
    Next k
  Next j
Next i
```

Рис.32. Код формирования матрицы достижимости.

В представленном алгоритме формирования матрицы достижимости мы рассматриваем все возможные случаи уникальных маршрутов между пятью вершинами матрицы G. Как видно из представленного кода программы проверяемый маршрут может состоять из одного, двух, трех и четырех дуг.

Построим матрицу сильной связности графа G (рис. 33). Алгоритм ее построения сводится к нахождению пар противоположно направленных маршрутов между вершинами.

```
For i = 1 To 5                         'Рассчитываем матрицу сильной связности GSS
  For j = 1 To 5
    If GS(i, j) = 1 And GS(j, i) = 1 Then GSS(i, j) = 1 Else GSS(i, j) = 0
  Next j
Next i
```

Рис. 33. Расчет матрицы сильной связности графа G.

Верификация выполненного задания заключается в проверке трех заполненных матриц: матрицы смежности, матрицы достижимости и матрицы сильной связности (рис. 34).

```
M = 0                                   'Проверяем матрицу смежности GM
For i = 1 To 5
  For j = 1 To 5
    If Cells(i + 41, j + 1) = GM(i, j) And Cells(i + 41, j + 1) <> "" Then _
        M = M + 1 Else: Call krest(i + 41, j + 1)
  Next j
Next i
If M = 25 Then mark = mark + 1
M = 0                                   'Проверяем матрицу достижимости GS
For i = 1 To 5
  For j = 1 To 5
    If Cells(i + 41, j + 13) = GS(i, j) And Cells(i + 41, j + 13) <> "" Then _
    M = M + 1 Else: Call krest(i + 41, j + 13)
  Next j
Next i
If M = 25 Then mark = mark + 1
M = 0                                   'Проверяем матрицу сильной связности GSS
For i = 1 To 5
  For j = 1 To 5
    If Cells(i + 41, j + 7) = GSS(i, j) And Cells(i + 41, j + 7) <> "" Then _
        M = M + 1 Else: Call krest(i + 41, j + 7)
  Next j
Next i
If M = 25 Then mark = mark + 1
```

Рис. 34. Проверка матриц смежности, достижимости и сильной связности.

За каждую правильно заполненную матрицу курсанты получают по одному баллу.

В задании «11. Опишите результат функционирования представленного автомата» программного комплекса «Дискретные автоматы» обучающиеся должны заполнить таблицу (рис. 35), пошагово отображая элементы входных $x(t)$, выходных $y(t)$ слов, а также состояние автомата.

11. Опишите результат функционирования представленного автомата.

f/g	s0	s1*	s2
В	s1/2	s2/1	s1/1
П	s2/1	s0/0	s0/1
Т	s0/1	s1/0	s2/1

t	0	1	2	3	4	5	6	7	8	9	10
x(t)											
y(t)											
s(t)											

x= В Т П Т П В П П В Т

Рис. 35. Сгенерированное задание «11. Опишите результат функционирования представленного автомата».

Генерация данных происходит здесь следующим образом. Сначала сформируем и выведем на лист алфавит автомата, состоящий всего из трех букв y1, y2 и y3, для этого определим соответствующие номера этих букв в константе *Alfavit*.

```
Const Alfavit As String = "АБВГДЕЗИКЛМНОПРСТУ"
Do
  x1 = Rnd(V) * 17 + 1
  x2 = Rnd(V) * 17 + 1
  x3 = Rnd(V) * 17 + 1
Loop Until x1 <> x2 And x2 <> x3 And x1 <> x3
y1 = Mid(Alfavit, x1, 1)
y2 = Mid(Alfavit, x2, 1)
y3 = Mid(Alfavit, x3, 1)
Cells(49, 2) = y1
Cells(50, 2) = y2
Cells(51, 2) = y3
```

Рис. 36. Определение алфавита автомата.

В представленном фрагменте кода макроса формируется входное слово *slovo1* (рис. 47).

```
For i = 1 To 10
  t = Rnd(V) * 2
  If t = 0 Then
    r(i) = y1
    slovo1 = slovo1 & y1 & " "
  End If
  If t = 1 Then
    r(i) = y2
    slovo1 = slovo1 & y2 & " "
  End If
  If t = 2 Then
    r(i) = y3
    slovo1 = slovo1 & y3 & " "
  End If
Next i
Cells(52, 3) = slovo1
```

Рис. 37. Формирование входного слова.

Затем в программе определяется начальное состояние автомата *p* и заполняются заголовки таблицы (рис. 38).

```
p = Rnd(V) * 2
If p = 0 Then
  Cells(48, 3) = "s0" & "*"
  Cells(48, 4) = "s1"
  Cells(48, 5) = "s2"
End If
If p = 1 Then
  Cells(48, 3) = "s0"
  Cells(48, 4) = "s1" & "*"
  Cells(48, 5) = "s2"
End If
If p = 2 Then
  Cells(48, 3) = "s0"
  Cells(48, 4) = "s1"
  Cells(48, 5) = "s2" & "*"
End If
```

Рис. 38. Инициализация автомата.

Генерация задания заканчивается заполнением таблицы перехода состояний и расчета значений функции $y(x)$автомата Мили (рис. 39).

```
For i = 1 To 3        'Создание и вывод массива M
   For j = 1 To 3
     M(i, j) = Rnd(V) * 2
   Next j
Next i
For j = 1 To 3
  Do
    L(1, j) = Rnd(V) * 2.4
    L(2, j) = Rnd(V) * 2.4
    L(3, j) = Rnd(V) * 2.4
  Loop Until L(1, j) <> L(2, j) And L(2, j) <> L(3, j) And L(3, j) <> L(1, j)
Next j
For i = 1 To 3
  For j = 1 To 3
    Cells(i + 48, j + 2) = "s" & L(i, j) & "/" & M(i, j)
  Next j
Next i
```

Рис. 39. Расчет значений автомата-преобразователя.

Проверка задания представлена на рисунке 40. Сначала мы считываем и убираем пробелы из ответа обучаемого. Затем в цикле For .. Next определяются и сравниваются с заполненной таблицей значения аргумента $x(t)$, выходной функции $y(t)$ и состояния автомата $s(t)$ на каждом такте работы автомата, собирается и проверяется выходное слово.

```
If No_Space(Cells(51, 8)) <> No_Space("s" & p) Then
  podchet = podchet + 1
  Call krest(51, 8)
End If
For i = 1 To 10
  Select Case p
    Case 0: столб = 1
    Case 1: столб = 2
    Case 2: столб = 3
  End Select
  Select Case r(i)
    Case y1: строка = 1
    Case y2: строка = 2
    Case y3: строка = 3
  End Select
  p = L(строка, столб)
  n = n & M(строка, столб)
  If No_Space(Cells(50, 8 + i)) <> No_Space(M(строка, столб)) Then
    Call krest(50, 8 + i)
    podchet = podchet + 1
  End If
  If No_Space(Cells(51, 8 + i)) <> No_Space("s" & L(строка, столб)) Then
    podchet = podchet + 1
    Call krest(51, 8 + i)
  End If
Next i
If podchet = 0 Then mark = mark + 3
If podchet = 1 Then mark = mark + 2
If podchet = 2 Then mark = mark + 1
```

Рис. 40. Фрагмент кода проверки задания «11. Опишите результат функционирования представленного автомата».

В заключении отметим, что в настоящее время на кафедре информатики и математики Краснодарского университета МВД России разработаны десятки программ по изучаемым разделам математики. Все они внедрены в образовательный процесс на практических занятиях и в часы самоподготовки. Эффективное внедрение описываемых программных комплексов плодотворно влияет на скорость и качество подготовки учебного материала, активизирует процессы мышления, восприятия и усвоения знаний у обучаемых, формирует устойчивые навыки стандартных математических расчетов, побуждает курсантов и слушателей к самостоятельной, творческой работе.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Лаптев, В.Н. О технологиях разработки программных приложений для генерирования и проверки практических заданий по математическим дисциплинам / В.Н. Лаптев, Е.В. Михайленко // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета (Научный журнал КубГАУ) [Электронный ресурс]. – Краснодар: КубГАУ, 2020. – №01(155). С. 164 – 177. – IDA [article ID]: 1552001013. – Режим доступа: http://ej.kubagro.ru/2020/01/pdf/13.pdf, 0,875 у.п.л.

3. Михайленко, Е.В. Введение в программирование: учеб. пособие / Е.В. Михайленко – Краснодар: Краснодарский университет МВД России, 2020. 109 с.

3. Михайленко, Е.В. Введение в программирование: учеб. пособие / Е.В. Михайленко – Краснодар: Краснодарский университет МВД России, 2020. 109 с.

4. Михайленко, Е.В. Математика: учебник / Е. В. Михайленко. – Краснодар: Краснодарский университет МВД России, 2023. – 478 с.

5. Михайленко, Е.В. Методы создания компьютерных программ для автоматизации подготовки и проверки практических заданий на базе офисных приложений. / Е.В. Михайленко // Математические методы и информационно-технические средства [Электронный ресурс]: материалы XVIII Междунар. науч.-практ. конф. (17 июня 2022 г.) – Электрон. дан. – Краснодар: Краснодарский университет МВД России, 2022. – 1 электрон. опт. диск. С. 109 – 116

6. Михайленко, Е.В. О разработке комплекса программ для автоматизированной генерации и проверки заданий практической работы «Операции над многочленами» / Е.В. Михайленко // Проблемы информационного обеспечения деятельности правоохранительных органов: сборник статей IX Всероссийской научно-практической конференции. – Белгород: Белгородский юридический институт Министерства внутренних дел Российской Федерации им. И.Д. Путилина. С. 122 – 130.

7. Михайленко, Е.В. Объектно-ориентированное программирование: учебно-практическое пособие / Е.В. Михайленко, В.А. Епифанцева. – Краснодар: Краснодарский университет МВД России, 2021. –109 С.

8. Михайленко, Е.В. Особенности разработки компьютерных приложений для генерации и верификации практических заданий, включающих обработку матриц / Е.В. Михайленко // Проблемы информационного обеспечения деятельности правоохранительных органов: сборник статей 7-й Всероссийской научно-практической конференции. – Белгород: Белгородский юридический институт МВД России имени И.Д. Путилина, 2020. – С. 3 – 20.

9. Михайленко, Е.В. Принципы автоматизации подбора и проверки практических заданий, включающих обработку полиномов / Е.В. Михайленко // Математические методы и информационно-технические средства: материалы XVI Всерос. науч.-практ. конф. – Краснодар: Краснодарский университет МВД России, 2020. С. 65 – 70.

10. Михайленко, Е.В. Технологии программирования: учеб. пособие / Е.В. Михайленко, А.К. Назаров – Краснодар: Краснодарский университет МВД России, 2019. 510 с.

11. Старостенко И. Н. Прикладная математика: учебник / И. Н. Старостенко, Е. В. Михайленко, А. А. Хромых. – Краснодар: Краснодарский университет МВД России, 2023. – 346 с.

12. Старостенко, И.Н. К вопросу об использовании информационных технологий при организации самостоятельной работы обучающихся / И.Н. Старостенко,

Е.В. Михайленко // Проблемы современного педагогического образования. – Ялта: РИО ГПА, 2019. № 65-1. С. 257 - 262.

References

1. Laptev, V.N. O tehnologijah razrabotki programmnyh prilozhenij dlja generirovanija i proverki prakticheskih zadanij po matematicheskim disciplinam / V.N. Laptev, E.V. Mihajlenko // Politematicheskij setevoj jelektronnyj nauchnyj zhurnal Kubanskogo gosudarstvennogo agrarnogo universiteta (Nauchnyj zhurnal KubGAU) [Jelektronnyj resurs]. – Krasnodar: KubGAU, 2020. – №01(155). S. 164 – 177. – IDA [article ID]: 1552001013. – Rezhim dostupa: http://ej.kubagro.ru/2020/01/pdf/13.pdf, 0,875 u.p.l.

3. Mihajlenko, E.V. Vvedenie v programmirovanie: ucheb. posobie / E.V. Mihajlenko – Krasnodar: Krasnodarskij universitet MVD Rossii, 2020. 109 s.

3. Mihajlenko, E.V. Vvedenie v programmirovanie: ucheb. posobie / E.V. Mihajlenko – Krasnodar: Krasnodarskij universitet MVD Rossii, 2020. 109 s.

4. Mihajlenko, E.V. Matematika: uchebnik / E. V. Mihajlenko. – Krasnodar: Krasnodarskij universitet MVD Rossii, 2023. – 478 s.

5. Mihajlenko, E.V. Metody sozdanija komp'juternyh programm dlja avtomatizacii podgotovki i proverki prakticheskih zadanij na baze ofisnyh prilozhenij. / E.V. Mihajlenko // Matematicheskie metody i informacionno-tehnicheskie sredstva [Jelektronnyj resurs]: materialy XVIII Mezhdunar. nauch.-prakt. konf. (17 ijunja 2022 g.) – Jelektron. dan. – Krasnodar: Krasnodarskij universitet MVD Rossii, 2022. – 1 jelektron. opt. disk. S. 109 – 116

6. Mihajlenko, E.V. O razrabotke kompleksa programm dlja avtomatizirovannoj generacii i proverki zadanij prakticheskoj raboty «Operacii nad mnogochlenami» / E.V. Mihajlenko // Problemy informacionnogo obespechenija dejatel'nosti pravoohranitel'nyh organov: sbornik statej IX Vserossijskoj nauchno-prakticheskoj konferencii. – Belgorod: Belgorodskij juridicheskij institut Ministerstva vnutrennih del Rossijskoj Federacii im. I.D. Putilina. S. 122 – 130.

7. Mihajlenko, E.V. Ob#ektno-orientirovannoe programmirovanie: uchebno-prakticheskoe posobie / E.V. Mihajlenko, V.A. Epifanceva. – Krasnodar: Krasnodarskij universitet MVD Rossii, 2021. –109 S.

8. Mihajlenko, E.V. Osobennosti razrabotki komp'juternyh prilozhenij dlja generacii i verifikacii prakticheskih zadanij, vkljuchajushhih obrabotku matric / E.V. Mihajlenko // Problemy informacionnogo obespechenija dejatel'nosti pravoohranitel'nyh organov: sbornik statej 7-j Vserossijskoj nauchno-prakticheskoj konferencii. – Belgorod: Belgorodskij juridicheskij institut MVD Rossii imeni I.D. Putilina, 2020. – S. 3 – 20.

9. Mihajlenko, E.V. Principy avtomatizacii podbora i proverki prakticheskih zadanij, vkljuchajushhih obrabotku polinomov / E.V. Mihajlenko // Matematicheskie metody i informacionno-tehnicheskie sredstva: materialy XVI Vseros. nauch.-prakt. konf. – Krasnodar: Krasnodarskij universitet MVD Rossii, 2020. C. 65 – 70.

10. Mihajlenko, E.V. Tehnologii programmirovanija: ucheb. posobie / E.V. Mihajlenko, A.K. Nazarov – Krasnodar: Krasnodarskij universitet MVD Rossii, 2019. 510 s.

11. Starostenko I. N. Prikladnaja matematika: uchebnik / I. N. Starostenko, E. V. Mihajlenko, A. A. Hromyh. – Krasnodar: Krasnodarskij universitet MVD Rossii, 2023. – 346 s.

12. Starostenko, I.N. K voprosu ob ispol'zovanii informacionnyh tehnologij pri organizacii samostojatel'noj raboty obuchajushhihsja / I.N. Starostenko, E.V. Mihajlenko // Problemy sovremennogo pedagogicheskogo obrazovanija. – Jalta: RIO GPA, 2019. № 65-1. S. 257 - 262.