ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ
Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра «Информатики и вычислительной техники»

И.А. Стефанова

ПРОГРАММИРОВАНИЕ В СИСТЕМЕ MATHCAD

Задания и методические указания
к лабораторным работам по информатике

Самара
2015

УДК: 004.42: 519.85

С

Рекомендовано к изданию методическим советом ПГУТИ,
протокол № 22, от 16.04.2015 г.

Стефанова, И. А.

С **Информатика:** методическое пособие по выполнению лабораторных работ / И. А. Стефанова. – Самара: ПГУТИ, 2015. – 52 с.

Методическое пособие «Программирование в системе Mathcad» содержит 4 лабораторные работы, позволяющие студентам освоить программирование основных алгоритмических структур, таких как линейные, разветвляющиеся и циклические (регулярные и итерационные), которые реализуются в системе Mathcad в виде модулей.
Методическое пособие разработано в соответствии с ФГОС ВПО по направлениям подготовки: бакалавра «11.03.02 *Инфокоммуникационные технологии и системы связи (ИКТ)»,* бакалавра «12.03.03 *Фотоника и оптоинформатика* (ОИТ)» и бакалавра «11.03.01 *Радиотехника*». Предназначено для использования на практических занятиях по дисциплине «Информатика» при подготовке студентов ФБТО очной полной формы обучения 1 курса во 2 семестре.

ISBN

Оглавление

РЕЦЕНЗЕНТ:
АЛАШЕЕВА Е.А., к.ф.-м.н., доц. кафедры ВМ

Введение

Данный цикл лабораторных работ включает в себя четыре лабораторные работы, направленные на изучение основ алгоритмизации и программирования в математической системе *Mathcad*. Цикл может использоваться на практических занятиях по дисциплине «Информатика» при подготовке бакалавров телекоммуникационных направлений (11.03.02) и направлений фотоники и оптоинформатики (12.03.03).

Настоящее методическое пособие поможет студентам сориентироваться в учебном материале по изучению возможностей математических пакетов при решении математических, научно-технических и инженерных задач, а так же успешно выполнить учебный план указанных дисциплин в целом.

Содержание отчета

1. Название работы, цель работы, задание в соответствии с вариантом.
2. Блок-схем алгоритма в соответствии с номером варианта.
3. Программы в виде *ScreenShots* выполнения заданий в *Mathcad*.
4. Выводы относительно возможностей программирования основных алгоритмических структур в системе *Mathcad*.

Интерфейс системы Mathcad

Все работы проделываются в рабочем окне, которое открывается после запуска системы *Mathcad* (рис.1). В окне системы присутствует панель палитр математических знаков. В рабочем поле *Mathcad* создаются программные блоки (модули), которые строятся с помощью операторов палитры *Программирование*.

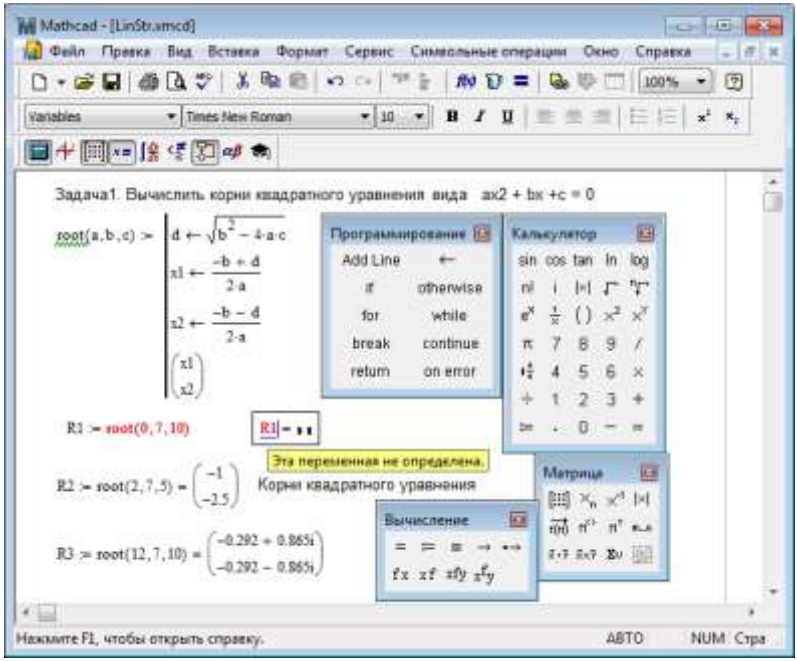


Рис.1 Окно системы *Mathcad*-14 и панели с математическими знаками

1. Алгоритмы линейной структуры (1 час)

1.1 Цель работы

Научиться программировать разными способами линейные алгоритмические структуры в системе *Mathcad*.

1.2 Подготовка к работе

По указанной литературе изучить:

– состав палитры *Программирование*;
– виды операторов присваивания и их действие;
– понятие и создание функции пользователя;
– правила создания программы-функции;

1.3 Задание и порядок выполнения работы

1. Создать в текстовой области заголовок документа «Программирование линейных структур».

2. Задача 1. Рассчитать функцию, заданную в табл. 1.1. по варианту двумя способами:

– программированием в тексте документа,
– с использованием программы-функции.

В первом случае:

– ввести исходные данные (см. рис. 1.1),
– задать ранжированную переменную, изменяющуюся в пределах $xo \div xn$ с шагом h,
– рассчитать и вывести в виде таблицы значения функции и аргумента в заданном интервале,
– построить график функции. Отформатировать его для наглядного представления задания.

Во втором случае:

– создать программу функцию, с использованием операторов палитры *Программирование*,
– предусмотреть в задании ввод исходных данных с использованием оператора как локального, так и глобального присваивания (см. рис. 1.1).

Таблица 1.1 Варианты заданий

N	Функция	*xo*	*xn*	*h*	*a*	*b*
1	$y = \frac{x^2(x+1)}{b} - b\sin^2(x+1);$	-2.5	2	0,5	-	5
2	$z = \sqrt{\frac{(x+b)^2}{a}} + \cos(x+b)$	-3,5	3	0,7	2	4
3	$f = \sqrt{a\sin x + \lvert b\sin x \rvert}$	-3,0	3,0	0,6	3	5
4	$r = \frac{a^{2x} + b^{-x}\cos(a+b)x}{\sin(a+b)x}$	-3,3	-0,3	0,3	5	5
5	$d = e^{-bx}\frac{x + \sqrt{(x+a)^2}}{x - \ln\lvert x+a \rvert}$	-2,4	-0,6	0,2	3	1,5
6	$y = \sin^3(x^2 + a) - \sqrt{\frac{x^2+a}{b}}$	-3	2,4	0,6	2	5
7	$w = (a - x)\frac{b - a/(a-x)}{1 + (a-x)^2}$	-3	2	0,5	-1	10
8	$s = \ln(a + x^2) + \sin\frac{(a+x^2)}{b}$	-3,5	3,5	0,6	5	2
9	$r = e^{-bx}\sin(ax+b) - \sqrt{\lvert ax+b \rvert}$	-2,4	1,5	0,4	-0,5	2
10	$s = x^{b/x} - \sqrt{\lvert b/x \rvert}$	-0,5	2,2	0,3	-	10
11	$w = \sqrt{x^2 + b} - \frac{\cos(x^2+b)}{a}$	-5	5	0,2	-0,5	15,5
12	$s = e^{-ax}\sqrt{\lvert x \rvert + 1} + e^{-bx}\sqrt{\lvert x \rvert + 1}$	-0.5	5	0,3	0,5	0,2
13	$y = \ln(a + x^2) + \sin\frac{(a+x^2)}{b}$	-3,5	3,5	0,2	5	2
14	$g = \sqrt{x^2 + b} - \frac{\cos(x^2+b)}{a}$	-5	5	0,2	-0,5	15
15	$z = \sin^3(x^2 + a) - \sqrt{\frac{x^2+a}{b}}$	-3	2,9	0,3	2	5
16	$t = \frac{a^{2x} + \cos(a+b)x}{\sin(a+b)x}$	-3,0	3,0	0,4	3	5

3. Задача 2. Составить программу для вычисления корней квадратного уравнения: $a \cdot x^2 + b \cdot x + c$ по известной формуле:

$$x_{1,2} = \frac{-b \pm \sqrt{d}}{2 \cdot a}, \quad где \quad d = b^2 - 4 \cdot a \cdot c$$

Решить задачу двумя способами:

– вызовом операторов в вычисляемые области, т.е. в тексте документа,
– с использованием модульного программирования.

В первом случае:

– задать произвольные значения коэффициентам *a, b* и *c,*
– вычислить дискриминант *d,*
– вычислить корни *x1, x2* по приведенным формулам,
– произвести проверку решения путем подстановки найденных корней в квадратное уравнение, заданное коэффициентами *a, b* и *c.*

Во втором случае:

– задать программу функцию, например *roots(a,b,c),*
– создать оператором A*dd Line* программный блок, в строках которого рассчитать последовательно *d, x1, x2,*
– задать вывод корней в модуле в векторном виде $\begin{pmatrix} x1 \\ x2 \end{pmatrix}$,
– вызвать результат расчета функции-модуля для разных значений *a, b* и *c.* Проверить разные ситуации, когда 1) *a=0* и уравнение становится линейным (сообщение ошибки, что на ноль делить нельзя), 2) $d < 0$ и корни мнимые, 3) $d > 0$ – корни действительные,
– произвести проверка решения путем подстановки найденных корней в квадратное уравнение, заданное коэффициентами *a, b* и *c.*

1.4 Методические указания

Под линейным алгоритмом понимается вычислительный процесс, в котором необходимые операции выполняются строго последовательно друг за другом. Операторы, реализующие этот процесс, размещаются последовательно и выполняются все, начиная с первого оператора и кончая последним.

Реализовать тот или иной алгоритм вычисления в пакете *Mathcad* можно двумя способами:

– вызывая соответствующие операторы, или функции в текст документа;
– используя программы-функции.

Первый способ реализуется записью соответствующих конструкций непосредственно в математических областях документа *Mathcad*, и он приемлем для сравнительно простых алгоритмов.

Второй способ реализуется в виде отдельных программных модулей, которые называют программами-функциями (ПФ). Такое программирование включает два этапа:

– описание программы-функции;
– вызов программы-функции.

Описание программы - функции

Перед тем как использовать программу-функцию нужно ее задать, т.е. выполнить описания. Описание программы-функции размещается в рабочем документе перед вызовом программы-функции и включает в себя имя программы-функции, список формальных параметров (который может отсутствовать) и тело программы-функции.

Каждая программа-функция *Mathcad* имеет уникальное имя, используя которое осуществляется обращение к этой программе-функции. Через это же имя (и только через него) возвращается в рабочий документ результат выполнения программы-функции. После имени программы-функции идет список формальных параметров, заключенный в круглые скобки. Через формальные параметры внутрь программы-функции передаются данные необходимые для выполнения вычислений внутри программы. В качестве формальных параметров могут использоваться имена простых переменных, массивов и функций. Формальные параметры отделяются друг от друга запятой.

Если программа-функция не иметь формальных параметров, тогда данные передаются через имена переменных, определенных выше описания программы-функции.

Тело программы-функции (за жирной чертой оператора *Add Line*) включает любое число операторов локальных операторов присваивания, условных операторов и операторов цикла, а также вызов других программ-функций и функций пользователя.

Для создания подпрограмм можно использовать операторы, хранящиеся в палитре *Программирование* (рис. 1.1). Программный блок организуется с помощью оператора *Add Line* (добавить строку). Этот оператор создает и расширяет жирную вертикальную линию, справа от которой записывается программный блок. В теле модуля в качестве оператора локального присваивания используется символ ←. Например, присвоение переменной x значения 2 запишется как $x \leftarrow 2$. Остальные операторы блока *Программирование* будут описаны в последующих лабораторных работах.

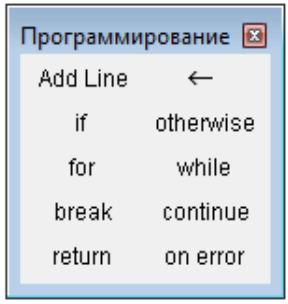


Рис. 1.1 Палитра «Программирование»

Порядок создания программы-функции Mathcad

Для ввода в документ описания программы-функции необходимо:

− ввести имя программы-функции и список формальных параметров, заключенный в круглые скобки,

− ввести оператор локального присваивания : =;

− открыть панель *Программирования* и щелкнуть кнопкой *Add Line* . На экране появится вертикальная черта и вертикальный столбец с двумя полями ввода для ввода операторов, образующих тело программы-функции.

− в маркеры ввести поочередно операторы тела программы-функции.

$$root(a,b,c) := \left|\begin{array}{l}\blacksquare\\ \blacksquare\end{array}\right.$$

$$f(x) := \left|\begin{array}{l} a \leftarrow 2 \\ x \leftarrow x + a \\ z \leftarrow x^2 \\ z \end{array}\right.$$

При повторном щелчке по кнопке *Add Line* панели программирования в модуль добавляется дополнительный маркер для образования новой строки модуля в место после установки кур-

сора. Для удаления оператора или поля ввода из тела программы-функции, нужно заключить его в выделяющую рамку и нажать клавишу *Delete*.

− заполнить самое нижнее поле ввода, введя туда выражение, определяющее возвращаемое через имя программы-функции значение (на рисунке это переменная *Z*).

Как известно, обычным переменным в Mathcad нужно хотя бы раз присвоить значение. Для этого используется оператор присваивания. Оператор присваивания может быть локальным или глобальным:

1. переменные, значения которым присвоены локально, считаются известными в блоках справа и ниже от выражения. Для локального присваивания используется оператор :=, а в блоках программирования – оператор ←. Он имеет вид:

   < имя - переменной > ← < выражение >.

2. переменные, значения которым присвоены глобально, считаются известными во всех блоках документа. Для глобального присваивания используется оператор ≡.

Операторы присваивания :=, ≡, ← вводятся с помощью соответствующих палитр *Калькулятор, Вычисление, Программирование* (рис.1.1 и 1.2)

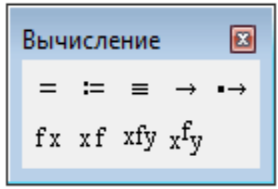


Рис. 1.2 Палитра «Вычисление»

Обращение к программе-функции Mathcad

Для выполнения программы-функции необходимо обратиться к имени программы-функции с указанием списка фактических параметров (если в описании программы присутствует список формальных параметров), т.е.

< имя_программы > (список фактических параметров)

Фактические параметры указывают, при каких конкретных значениях осуществляются вычисления в теле программы. Фактические параметры отделяются друг от друга запятой.

Очевидно, что между фактическими и формальными пара-

метрами должно быть соответствие по количеству, порядку следования и типу.

На рис. 1.3 приведен пример расчета функции в тексте документа, и с помощью ПФ программного блока, Как видно результаты решения по обоим способам совпали.

Программирование линейных структур

Вычисление функции с использованием ранжированной переменной

$a := 5$

$f(x) := x^3 \cdot \sin\left(\sqrt{x+2}\right) + \frac{\sqrt{x+2}}{a}$ Функция пользователя

$xo := -2.0 \quad xn := 2.5 \quad h := 0.5$ Исходные данные

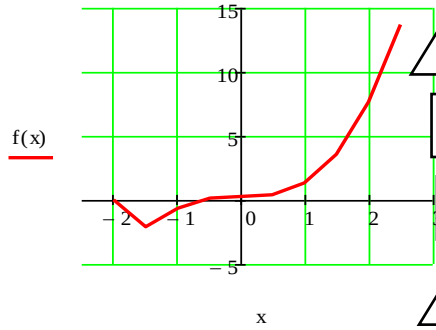
$x := xo, xo + h .. xn$ Ранжированная переменная

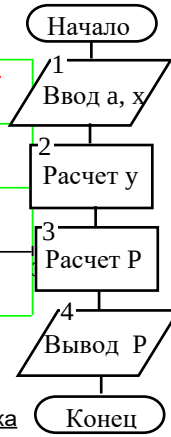
f(x) =

0
-2.051
-0.641
0.127
0.283
0.441
1.333
3.598
7.674
13.741

x =

-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5





Вычисление функции с использованием программного блока

1. С применением оператора локального присвоения

$P := \left| \begin{array}{l} a \leftarrow 5 \\ x \leftarrow 2 \\ y \leftarrow \sqrt{x+2} \\ P \leftarrow x^3 \cdot \sin(y) + \frac{y}{a} \end{array} \right.$

$P = 7.674$

2. С применением оператора глобального присвоения

$FP(x) := \left| \begin{array}{l} y \leftarrow \sqrt{x+2} \\ x^3 \cdot \sin(y) + \frac{y}{a} \end{array} \right.$

$x \equiv 5 \qquad a \equiv 5$

$FP(x) = 60.001$

$FP(2) = 7.674$

Рис. 1.3 Программирование линейных структур

Между фактическими и формальными параметрами должно быть соответствие, что означает:

− если формальным параметром является простая переменная, то в качестве фактического значения может использоваться константа, переменная, арифметическое выражение;

− если формальным параметром является вектор или матрица, то фактическим должен быть вектор или матрица;

− если формальным параметром является имя встроенной функции или другой программы, то и фактическим параметром должен являться тот же объект.

Обращение к программе-функции должно находиться после описания программы-функции и к моменту обращения фактические параметры должны быть определены.

В системе *Mathcad* может быть реализовано модульное программирование, общая идея которого, состоит в следующем:

− реализация вычислительных процессов в виде отдельных программных единиц - модулей;

− обращение к этим модулям в других программах с передачей данных, необходимых для вычислительного процесса.

Как известно модульное программирование позволяет уменьшить объем исходных текстов программ, сделать их более «читаемыми», ускорить написание и тестирование программ, уменьшить расходы на сопровождение (эксплуатацию) программ.

Модульное программирование в пакете *Mathcad* можно реализовать двумя методами:

−модульное программирование в пределах одного документа;

− модульное программирование в нескольких документах.

Модульное программирование в пределах одного документа *Mathcad* характеризуется тем, что:

− для реализации простых вычислений используются локальные функции, а более сложных - программы-функции;

− описание локальных функций, программ-функций и их вызов (т.е. обращение к ним) находятся в пределах одного документа и хранятся в одном файле.

Для передачи в программный блок значений переменных можно использовать переменные документа, которые ведут себя в блоке как глобальные переменные. Модулю присваивается имя

со списком переменных, после которого идет оператор присвоения:=. Переменные в списке являются локальными, и им можно присваивать значения при вызове функции, заданной модулем.

Модульное программирование в нескольких документах M*a*thcad характеризуется тем, что:

− описание локальных функций, программ-функций находятся в одном документе и хранятся в определенном файле,

− вызов локальных функций, программ-функций (т.е. обращение к ним) осуществляется в другом документе, и хранятся в другом файле.

Присоединение файла с другим документом *Mathcad*, в котором находится описание вызываемой программы-функции, происходит с помощью специального оператор *Reference* (ссылка).

1.5 Содержание отчета

1. Название и цель работы.
2. Задания, блок-схемы и *ScreenShots* выполнения заданий, вставленные в документ, созданный в редакторе *MS Word*.
3. Выводы по работе относительно способов программирования алгоритмических структур и действие операторов глобального и локального присваивания.

1.6 Контрольные вопросы

1. Поясните состав палитры *Программирование*;
2. Привести примеры использования операторов локального и глобального присваивания. Объясните их назначение.
3. Какие способы реализации алгоритмов вычисления существуют в *Mathcad*?
4. Назначение программы-функции?
5. Поясните на примере описание программы-функции.
6. Как происходит обращение к программе-функции?
7. Каков порядок создания программы-функции *Mathcad*?
8. Что такое формальные параметры программы - функции?
9. Какое существует соответствие между фактическими и формальными параметрами?
10. Преимущества модульного программирования и его реализация в *Mathcad*.

2. Алгоритмы структуры ветвления

2.1 Цель работы

Научиться программировать алгоритмические структуры ветвления разными способами

2.2 Подготовка к работе

По указанной литературе изучить:
- состав палитры *Программирование, Графика*;
- правила записи функции условных выражений;
- операторы условия … *if*… и … *otherwise*;
- программирование вложенных условных выражений.

2.3 Задание и порядок выполнения работы

1. Создать заголовок документа «Программирование разветвляющихся структур».

3. Задача 1. Составить программу – функцию, вычисляющую функцию *t(x)*, заданную по варианту в табл. 2.1. Вывести:
- таблицу значений аргумента и функции шагом 1 и
- график заданной функции с шагом 0,2 (рис. 2.1, 2,4).

Таблица 2.1 Варианты заданий

N	Функция
1	$t = \begin{cases} e^x & если \quad 0 \le x < 1 \\ 20 \cdot \sin(x) & если \quad 1 < x \le 4 \\ \sum\limits_{n=1}^{10} \frac{x}{n^n} & если \quad 4 < x < 5 \\ x^3 + 1 & если \quad x \ge 5 \end{cases}$
2	$t = \begin{cases} 95 \cdot \sin(x^2) & если\ x \le 1 \\ x^2 - 5 & если\ 1 < x \le 6 \\ \sum\limits_{n=1}^{10} \frac{2^x}{n!} & если\ 6 < x < 9 \\ x^{2.5} & если\ x \ge 9 \end{cases}$

3	$t = \begin{cases} 9 \cdot \cos(x+1) & если \quad x \le 1 \\ x^2 - 25 & если \quad 1 < x \le 9 \\ \sum\limits_{n=1}^{10} \frac{1}{2^n} & если \quad 6 < x < 9 \\ x^2 + 5 & если \quad x \ge 9 \end{cases}$
4	$t = \begin{cases} 99 \sin(\sqrt{x+1}) & если -1 \le x < 5 \\ x^2 - 9 & если\ 5 < x \le 9 \\ \sum\limits_{n=1}^{10} \frac{50 \cdot n!}{(3 \cdot n)^n} & если\ 9 < x < 12 \\ x^2 + 10 & если \quad x \ge 12 \end{cases}$
5	$t = \begin{cases} 2 \cdot x + \sqrt[3]{x^2} & если -6 \le x < 0 \\ 40 \cdot \sin(x^2) & если \quad 0 < x \le 5 \\ \sum\limits_{n=1}^{15} \frac{30 \cdot n!}{n^n} & если\ 5 < x < 10 \\ x^{1.5} + e^{-x} & если \quad x \ge 10 \end{cases}$
6	$t = \begin{cases} \frac{1}{e^x - 1} & если \quad x \le 0 \\ \cos(3 \cdot x) & если\ 0 < x \le 8 \\ \sum\limits_{n=1}^{18} \frac{3^x}{(n+1)!} & если\ 8 < x < 11 \\ \sin(x^2) & если \quad x \ge 11 \end{cases}$
7	$t = \begin{cases} e^{-x} & если -5 \le x < 0 \\ 150 \cdot \sin^3(x) & если \quad 0 < x \le 7 \\ \sum\limits_{n=1}^{12} \frac{x}{3^n} & если \quad 7 < x < 10 \\ x^3 - x & если \quad x \ge 10 \end{cases}$

8	$t = \begin{cases} \sqrt[3]{x+1} & если -3 \le x < 0 \\ 30 \cdot \sin^3(x) & если \quad 0 < x \le 6 \\ \sum\limits_{n=1}^{15} \frac{30 \cdot n!}{(2 \cdot n)^n} & если\ 6 < x < 10 \\ \sqrt{x^3} & если \qquad x \ge 10 \end{cases}$
9	$t = \begin{cases} 80 \cdot \sin(x) & если \qquad x \le 0 \\ x + 100 & если \quad 0 < x \le 7 \\ \sum\limits_{n=1}^{3} \frac{2^x}{(n+1)!} & если \quad 7 < x < 10 \\ x^2 + \sqrt{x} & если \qquad x \ge 10 \end{cases}$
10	$t = \begin{cases} tg(x) & если \quad x \le 0 \\ x^2 + 12 & если \quad 0 < x \le 5 \\ \sum\limits_{n=1}^{10} \frac{\cos(x)}{n!} & если \quad 7 < x < 11 \\ \sqrt{x} + 1 & если \quad x \ge 11 \end{cases}$
11	$t = \begin{cases} x^3 + x & если \quad x \le 1 \\ x^2 - 36 & если \quad 1 < x \le 9 \\ \sum\limits_{n=1}^{10} \frac{1}{3^n} & если \quad 9 < x < 15 \\ x + 9 \cdot \sin(x) & если \quad x \ge 15 \end{cases}$
12	$t = \begin{cases} x^2 + \sin(x) & если -5 \le x < -1 \\ 9 \cdot \cos^2(x) & если -1 < x \le 9 \\ \sum\limits_{n=1}^{12} \frac{40 \cdot n!}{(3 \cdot n)^n} & если\ 9 < x < 13 \\ x^{-2} + 5 & если \qquad x \ge 13 \end{cases}$

13	$t = \begin{cases} 1 + e^{-x} & если -5 \le x < 0 \\ 40 \cdot \cos(x^3) & если \quad 0 < x \le 5 \\ \sum\limits_{n=1}^{20} \frac{4 \cdot n}{3^n} & если\ 5 < x < 10 \\ \ln^4(x) & если \qquad x \ge 10 \end{cases}$
14	$t = \begin{cases} x^2 + 5 & если \quad -5 \le x < 0 \\ 30 \sin(x) e^{-x} & если \quad 0 < x \le 5 \\ \sum\limits_{n=1}^{20} \frac{e^n}{(n+1)!} & если \quad 5 < x < 10 \\ tg(x^2) & если \qquad x \ge 10 \end{cases}$
15	$t = \begin{cases} e^{-x} + x & если -4 \le x < 0 \\ x^2 + 1 & если \quad 0 < x \le 3 \\ \sum\limits_{n=1}^{12} \frac{70 \cdot x}{3^{(n+2)}} & если\ 3 < x < 9 \\ 50 \cdot \sin(x) & если \qquad x \ge 9 \end{cases}$
16	$t = \begin{cases} x^2 + x & если -5 \le x < 0 \\ 50 \cdot \cos(x^2) e^{-x} & если \quad 0 < x \le 7 \\ \sum\limits_{n=1}^{10} \frac{e^{n-2}}{2^n} & если \quad 7 < x < 10 \\ tg(x^3) & если \qquad x \ge 10 \end{cases}$

4. Задача 2. Задана функция двух аргументов *f(x,y)*. Найти область определения этой функции. Составить программу, вычисляющую значение функции в области ее определения. Решение иллюстрировать графиками (см. рис. 2.5):

- поверхностным и
- контурным.

Варианты задания функции приведены в таблице 2.2

Таблица 2.2 Варианты заданий

N	Функция	N	Функция
1	$f(x,y)=\frac{1}{x^2\cdot y^2}$	9	$f(x,y)=x\cdot\ln(x^2\cdot y^2)$
2	$f(x,y)=x\cdot\sin(x\cdot y)$	10	$f(x,y)=\ln(x^2\cdot y^2)$
3	$f(x,y)=x^2\cdot\ln(x^2\cdot y^2)$	11	$f(x,y)=x^2\cdot\cos(x^2\cdot y^2)$
4	$f(x,y)=\ln(\frac{x^2}{y^2})$	12	$f(x,y)=\sin(x^2\cdot y^2)$
5	$f(x,y)=\frac{x^2}{y^2}$	13	$f(x,y)=\frac{x^2}{\ln(x^2\cdot y^2)}$
6	$f(x,y)=x^2\cdot\cos(x\cdot y)$	14	$f(x,y)=\sqrt{5\cdot x^2\cdot y^4}$
7	$f(x,y)=\sqrt{\frac{x^2}{y^2}}$	15	$f(x,y)=\frac{1}{\sqrt{x^2\cdot y^2}}$
8	$f(x,y)=\frac{x}{\ln(x^2\cdot y^2)}$	16	$f(x,y)=x^3\cdot\cos(x\cdot y)$

2.4 Методические указания

Алгоритм, в котором последовательность действий разделяется на два направления, в зависимости от итога проверки условия называется разветвляющимся. Они могут реализоваться в системе *Mathcad*. Существует ряд встроенных функций, у которых возвращаемый ими результат зависит от знака или значения аргумента. При их вычислении производится сравнение аргумента с некоторыми числовыми константами, например, с нулем или целыми числами. Для создания условных выражений широкие возможности дает функция *if*(). Выражение называется условным, если имеется несколько вариантов (альтернатив) вычисления его значений. Формат задания функции условных выражений:

if (условие, выражение_1, выражение_2).

Если условие выполняется, то вычисляется выражение 1, в противном случае – выражение 2.

Для вставки функции *if(...)* необходимо:

− щелкнуть по пиктограмме *f(x)* – вставить функцию,

− в окне *Вставка функции* найти категорию *Кусочно-непрерывные*,

− в этой категории найти в списке функций требуемую функцию if и нажать на кнопку *ОК*,

− вставляется ее шаблон $\text{if}(\blacksquare,\blacksquare,\blacksquare)$, в маркеры которого вводятся аргументы.

Если по условию задания необходимо написать выражение со сложным условием, то можно использовать логические операторы И (/\) ИЛИ(\/) и вложенный вариант функции if(), например

if (условие1, выражение_1, *if* (условие2,
выражение_2, выражение_3))

Более наглядно подобные задачи решаются средствами программирования с помощью оператора *if* (если)**,** который находится на палитре *Программирование.* Он задается в виде:

выражение *if* условие

Например, *sin*(x) *if* x > 0 {Возвращается sin(x) , если *x* > 0}.

Если условие выполняется, то возвращается значение выражения *sin(x).*

Совместно с оператором *if* используется оператор *otherwise* (иначе) в следующей конструкции:

$$f(x) := \begin{cases} 1 \ \text{if} \ x > 0 & \{\text{возвращает 1, если } x > 0\} \\ -1 \ \text{otherwise} & \{\text{возвращает} - 1, \text{во всех остальных случаях}\} \end{cases}$$

В системе *Mathcad* используется модульное программирование. Модуль может вести себя как функция без имени и параметров, но возвращает результат. Программный модуль может выполнять и роль тела функции пользователя с именем и параметрами и так же возвращать результат. В некоторых случаях один программный модуль может размещаться внутри другого.

На рис. 2.1 приведены примеры использования заданных функций *u(x)* и *v(x)* с использованием модульного программирования и операторов условных выражений.

Обратим внимание, что функция v не определена в диапазоне $0 < x < 5$. В этой ситуации программа должна обеспечить вывод соответствующего сообщения.

Пример 1. Вычисление функции u(x) с помощью программного блока

$$u(x) := \begin{cases} x^2 & \text{if } x \le 0 \\ 20 \cdot \sin(x) & \text{otherwise} \end{cases}$$

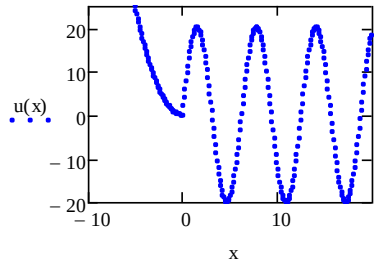
$x := -5, -4.9 .. 20$

$$u(x) = \begin{cases} x^2, & при\ x \le 0 \\ 20\sin(x), & при\ x > 0 \end{cases}$$

$u(-2) = 4$

$u(2) = 18.186$

$u(5) = -19.178$



$$v(x) = \begin{cases} x^2, & при\ x \le 0 \\ 20\sin(x), & при\ x \ge 5 \end{cases}$$

Пример 2. Вычисление функции V(x) с помощью программного блока

$$v(x) := \begin{cases} x^2 & \text{if } x \le 0 \\ \text{"no"} & \text{if } 0 < x < 5 \\ 20 \cdot \sin(x) & \text{otherwise} \end{cases}$$

$x := -5, -4.9 .. 20$

$v(-3) = 9$

$v(2) = \text{"no"}$

$v(10) = -10.88$

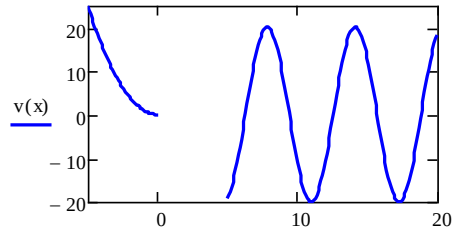


Рис. 2.1 Программы-функции, реализующие структуры ветвления

Как видно из примера 2, область определения аргумента функции *v(x)* разбита на три диапазона. В двух из них $x \le 0$ и $x \ge 5$

функция определена и вычисляется согласно условию задачи. В диапазоне от 0 до 5 функция не определена, поэтому необходимо выдать сообщение пользователю в случае, когда значения переменной попадает в этот диапазон. Подобные сообщения относятся к переменным строкового типа, и заключается в кавычки.

На рис. 2.2 и 2.3 приведены блок схемы решения задач.

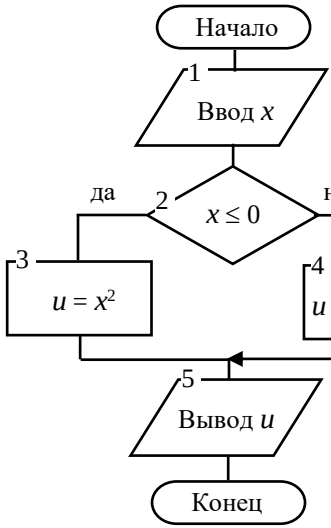


Рис. 2.2 Блок-схема алгоритма задачи 1

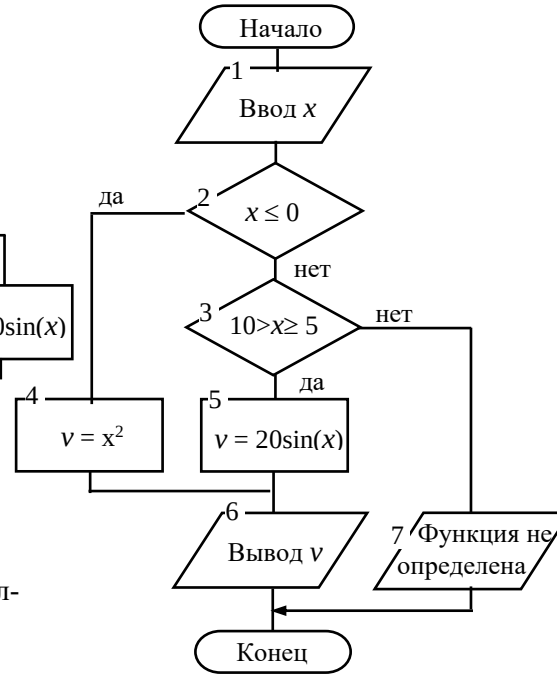


Рис. 2.3 Блок-схема алгоритма задачи 2

На рис. 2.4 рассмотрены примеры решения этих же задач *u(x), v(x),* но с использованием функций условных выражений.

Для подтверждения правильности составления выражения для вычисления функций *u(x)* и *v(x)* приведены таблицы их вычисленных значений. Во втором случае при *x*=2,5 выдается сообщение, что функция не определена.

Для создания сложных условий можно применять логические операторы /\ (И), \/ (ИЛИ).

Например, условие праздничные дни с 1.01 по 10.01 будет выглядеть в *Mathcad* следующим образом:

$$z(m,d) := \text{if}(m = 1 \wedge 1 \le d \le 10, \text{"holidays"}, \text{"work"})$$

$$z(2,5) = \text{"work"}$$

где *m* – переменная месяц, *d* – переменная день.

Причем, количество вложений может быть довольно большим.

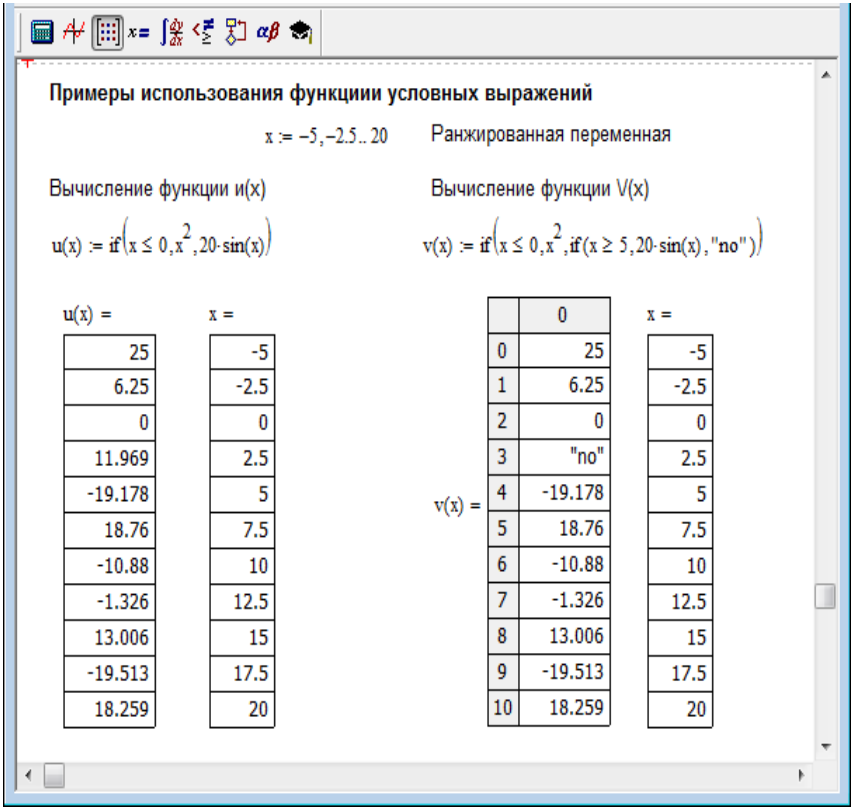


Рис. 2.4 Примеры использование функции условных выражений

Функция и операторы условных выражений могут применяться и в случае решения задач при использовании функций с двумя переменными, например *z(x,y)=cos(x*y).* Для решения по-

добных задач необходимо:

- найти область определения заданной функции – *z(x,y)*,
- создать программу-функцию или функцию пользователя, с ограничивающими условиями,
- вывести результат вычисления в виде графика (или матрицы значений).

На рис. 2.5 приведен пример вычисления логарифмической функции с графиками, подтверждающими решение.

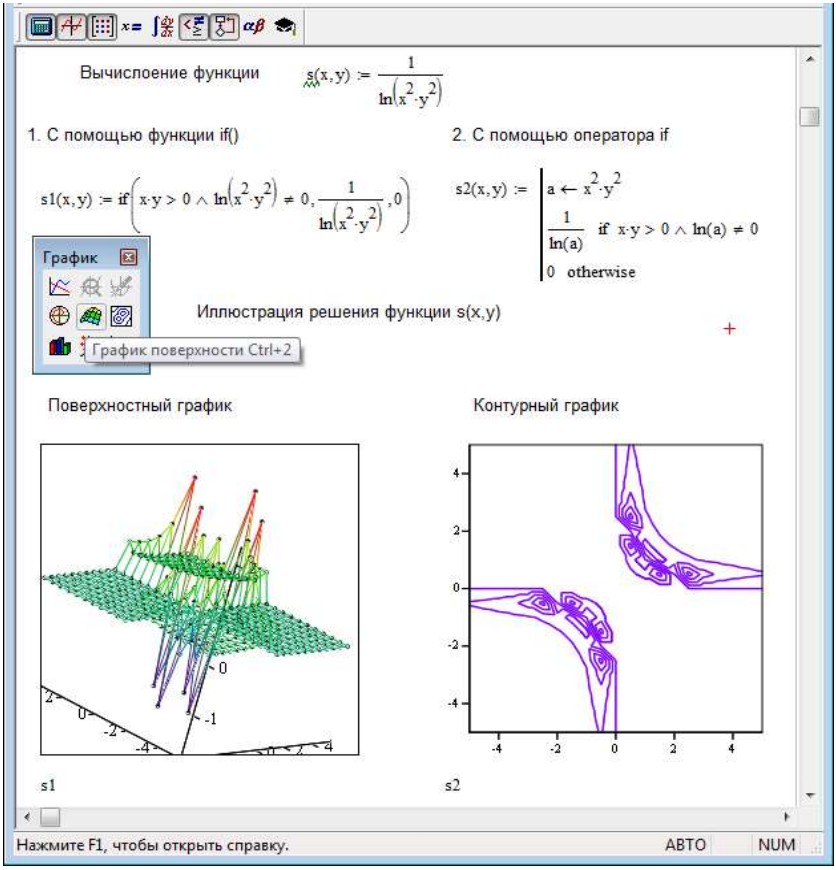


Рис. 2.5 Программирование функции двух переменных

Как известно, логарифмическая функция не существует при *x=0 (y=0)*, кроме того, она находится в знаменателе, который не

должен обращаться в 0, поэтому в условии ООФ должны входить два этих ограничения одновременно, что отображается в модуле логическим оператором И (/\).

Поскольку функция зависит от двух переменных (*x*,*y*), то поведение такой функции хорошо отражают поверхностные (вид сбоку) и контурные (вид сверху) графики, шаблоны которых находятся на одноименной палитре. Для построения графика достаточно создать функцию и указать ее имя в месте ввода шаблона графика. Недостатком такого метода построения является неопределенность в масштабировании, поэтому для получения приемлемого вида графика требуется его форматирование.

2.5 Содержание отчета

1. Название и цель работы.
2. Задания, блок-схемы и *ScreenShots* выполнения заданий, вставленные в документ, созданный в редакторе *MS Word*.
3. Выводы относительно методов организации разветвляющихся процессов (усеченного, полного и вложенного ветвления).

2.6 Контрольные вопросы

1. Состав палитры *Программирование.*
2. Охарактеризовать операторы, используемые для организации разветвляющихся процессов?
3. Каковы правила записи функции условных выражений.
4. Каковы правила записи данных логического типа в системе *Mathcad*?
5. Какие разновидности логических условий Вы знаете?
6. Как с помощью логических операторов $\wedge$ (И) и $\vee$ (ИЛИ) можно строить сложные условия? Привести примеры применения этих операторов для решения условных выражений.
7. Какие действия реализуются при выполнении условного оператора?
8. Как осуществляется программирование вложенных условий?
9. Зачем при отладке программы нужно тестировать все ветви разветвляющегося процесса?
10. Чем отличается функция *if*() от оператора …*if*…? Области их применения.

3. Алгоритмы итерационной циклической структуры

3.1 Цель работы

Научиться программировать итерационные циклы различными способами в системе *Mathcad.*

3.2 Подготовка к работе

По указанной литературе изучить:

– организацию циклов с неизвестным числом повторений;
– формат записи и область применения функции *until*();
– возможности системы программирования для организации циклов с неизвестным числом повторений с помощью операторов *while*…;
– формат записи и область применения оператора останова и продолжения вычислений.

3.3 Задание и порядок выполнения работы

1. Создать заголовок документа «Программирование итерационных циклических структур».

2. Задача 1. Рассчитать функцию в интервале от *xo* до *xn* с шагом 0,3, заданную в таблице 3.1. Определить количество значений функции в заданном диапазоне, значения аргумента и функции. Результаты представить в виде таблицы и графика.

Решить задачу двумя способами:
1) программирование в областях документа и
2) с использованием модуля программирования и оператора *while.*

Таблица 3.1 Варианты заданий

N	Функция	*xo*	*xn*	*a*
1	$t = \begin{cases} \ln^2(x+a), & при\ 0 < x < 3 \\ 3tgx, & при\ x \geq 3 \end{cases}$	0	5	1,65

2	$y=\begin{cases} a\cos^2 x, & npu\ x>4 \\ a\ln x+\sqrt{\lvert x\rvert}, & npu\ 2<x\le 4 \end{cases}$	2	5	2,5
3	$p=\begin{cases} tg(x+ax^2), & npu\ 1<x\le 3 \\ 2\cos(x+a), & npu\ x=1 \end{cases}$	0	3	2
4	$c=\begin{cases} a\cos x, & npu\ \ x<2 \\ x\sqrt{x+a}, & npu\ \ 2\le x<4 \end{cases}$	0	6	1,5
5	$f=\begin{cases} a-3\ln x, & npu\ x>2.5 \\ x-7/x^2, & npu\ x<2.5 \end{cases}$	1	5	2,3
6	$z=\begin{cases} ax^2+5x, & npu\ 3<x<5 \\ x+tgx, & npu\ x\le 3 \end{cases}$	1	5	2
7	$t=\begin{cases} a10^x, & npu\ x\ge 2 \\ \sin(\ln x), & npu\ 1\le x<2 \end{cases}$	0,5	3	1,5
8	$p=\begin{cases} a\ln x^2, & npu\ x\ge 4 \\ (x-2)^2, & npu\ 1\le x<4 \end{cases}$	0	5	1,5
9	$y=\begin{cases} \sin ax, & npu\ x<0 \\ x\cos x, & npu\ x>3 \end{cases}$	-2	5	2
10	$y=\begin{cases} \sqrt{x}-\cos x, & npu\ x<0 \\ \cos^2 ax, & npu\ 0\le x\le 2 \end{cases}$	-2	3	0,5
11	$y=\begin{cases} \sqrt{\lvert x\rvert}-\cos x, & npu\ \ x<0 \\ tgx-a, & npu\ \ x>2 \end{cases}$	-2	3	3,2
12	$b=\begin{cases} x\ln x^2, & npu\ \ x>4 \\ a\cos x, & npu\ \ x<3 \end{cases}$	1	5	3
13	$y=\begin{cases} \sin a\cdot x^2, & npu\ \ x<0 \\ x\cos^3 x, & npu\ \ x>3 \end{cases}$	-2	5	2
14	$y=\begin{cases} x+a\cdot tgx, & npu\ x=3 \\ x\cos x^2, & npu\ \ x>3 \end{cases}$	1	4	0,5
15	$z=\begin{cases} ax^2+x, & npu\ 3<x<5 \\ x+\sin x, & npu\ x\le 3 \end{cases}$	1	5	2

16	$t = \begin{cases} \ln(x^2 + a), & при\ 0 < x < 3 \\ x^2 + x, & при\ x \geq 3 \end{cases}$	0	6	1,5

3.4 Методические указания

Процессы, в которых многократно повторяются отдельные участки вычислений, называются циклическими. Каждое повторение происходит при новых значениях переменных. Различают:

– регулярные циклы, в которых число повторений заранее известно или определено;

– итерационные циклы, в которых число повторений неизвестно и выход из цикла осуществляется при выполнении некоторого условия (обычно, достижении заданной точности вычисления). Для циклических вычислений используются функция *until*() и операторы *for, while.*

С помощью средств системы *Mathcad* можно задать различные виды циклов (в том числе вложенных), реализовать различные итерационные и рекурсивные процедуры. Программирование в среде *Mathcad* осуществляется с помощью создания программных модулей. Программный модуль, в сущности, является функцией, но описанной с применением операторов панели *Программирование*. Эта функция возвращает значение, определяемое последним оператором. Это значит, что после такого модуля, выделенного как целый блок, можно поставить знак равенства для вывода результата его работы. В блоке могут содержаться любые операторы и функции входного языка системы. Для передачи в блок значений переменных можно использовать переменные документа, которые ведут себя в блоке как глобальные переменные.

Обычно модулю присваивается имя со списком переменных, после которого идет знак присваивания :=. Переменные в списке являются локальными и им можно присваивать значения при вызове функции, заданной модулем.

Набор программных элементов для создания программных модулей ограничен и содержит следующие операторы, размещенные на палитре *Программирование* (рис. 3.1):

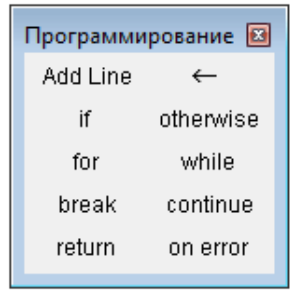


Рис. 3.1 Палитра «Программирование»

– оператор *for*, который служит для организации циклов с заданным числом повторений. Формат его записи:

for Имя переменной ∈ Nmin . . Nmax

Например, запись *for* n ∈ 0 . . 10 означает, что переменная n меняется от 0 до 10 с шагом 1;

– оператор *while* ("пока"). Служит для организации циклов, действующих до тех пор, пока выполняется некоторое условие. Формат записи: while ▮

While Условие ▮

Выполняемое выражение записывается на место, расположенное ниже шаблона

– оператор *break*. Вызывает прерывание работы программы всякий раз, как он встречается. Используется в основном совместно с операторами условного выражения *if* и цикла *while* и *for*;

– оператор *continue*. Используется для продолжения работы после прерывания программы и, обычно, совместно с операторами задания циклов *for* и *while*. Обеспечивает возвращение в точку прерывания и продолжение вычислений;

– оператор *return*. Прерывает выполнение программы и возвращает значение своего операнда, стоящего следом за ним. Например, *return* 0 if x<0 будет возвращать значение 0 при любом x<0.

– оператор *on error* и функция *error* обработки ошибок. Оператор задается в виде:

Выражение 1 *on error* Выражение 2.

Если при выполнении Выражения 1 возникает ошибка, то выполняется Выражение 2. Функция ошибки *error(S)*, которая будучи помещена в программный модуль, при возникновении ошибки возвращает окошко с надписью, хранящейся в символьной переменной *S*.

С помощью оператора *while* можно вычислить и циклы в виде арифметической прогрессии. На рис. 3.2 приведен пример расчета функции t(x) в заданном диапазоне изменения аргумента *x* на интервале -5..10 с шагом 0,5.

$$t = \begin{cases} \sin(x), & x > 0 \\ x^2 & x \le 0 \end{cases}$$

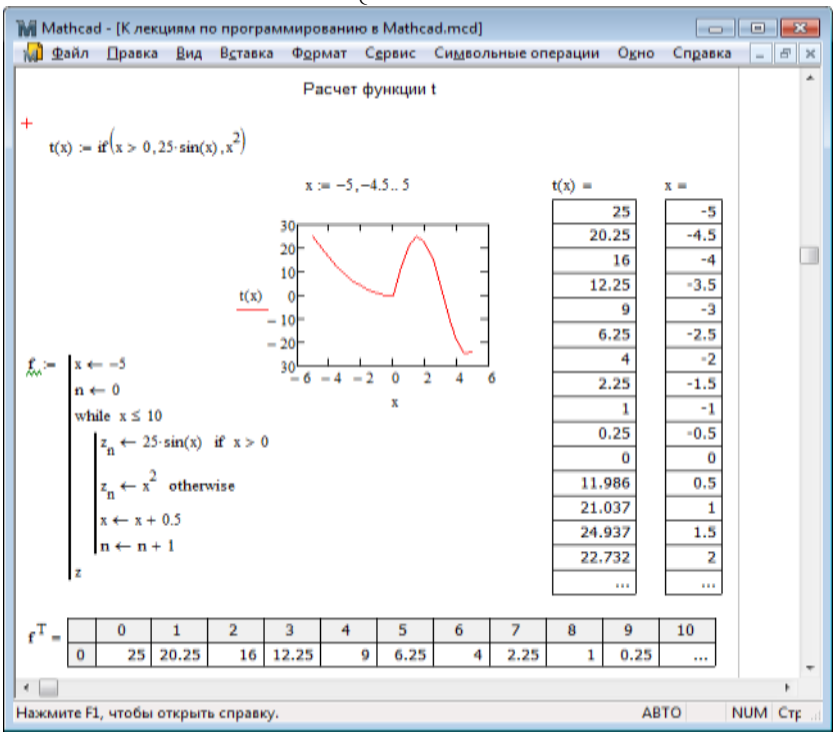
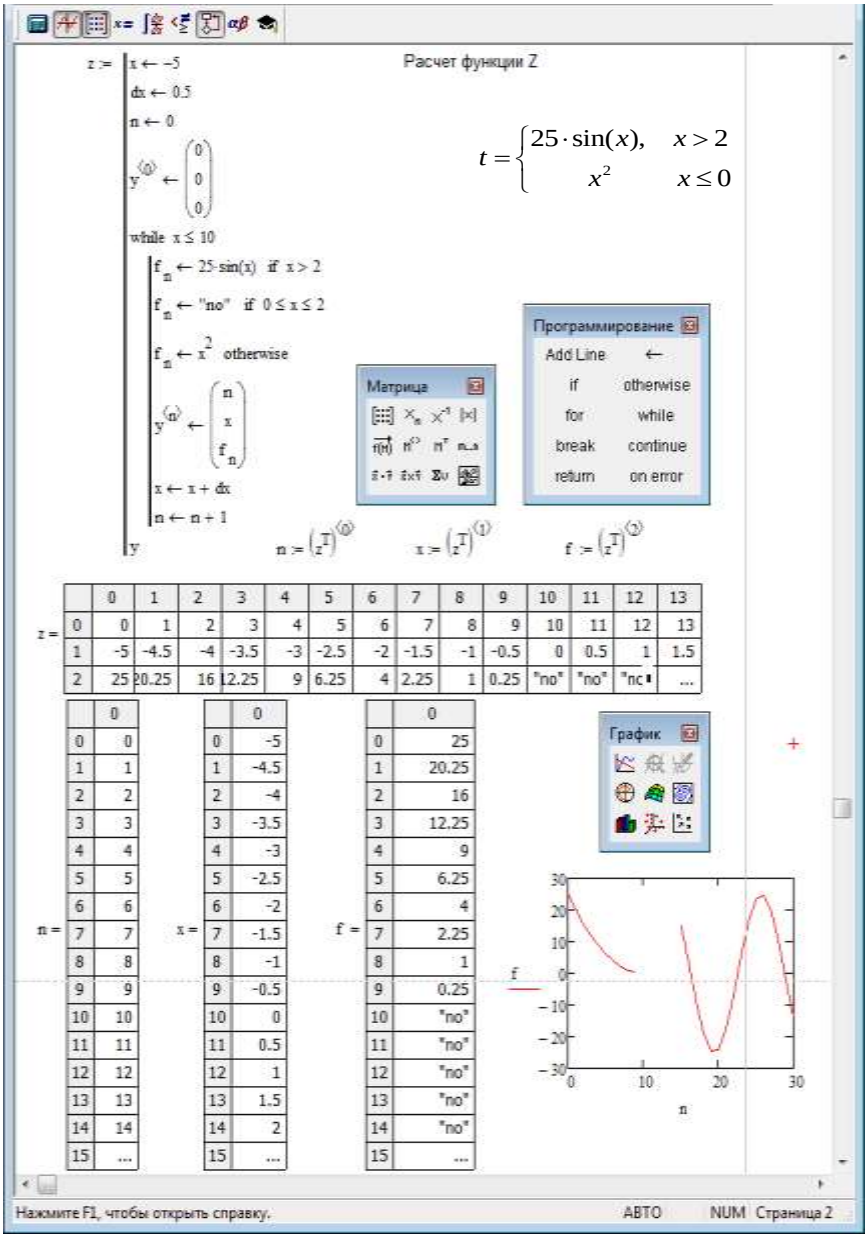


Рис.3.2 Пример программирования итерационных циклов

На рис. 3.5 приведен пример расчета подобной функции t(x), которая в диапазоне 0 .. 2 аргумента *x* не определена.



z =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	-5	-4.5	-4	-3.5	-3	-2.5	-2	-1.5	-1	-0.5	0	0.5	1	1.5
2	25	20.25	16	12.25	9	6.25	4	2.25	1	0.25	"no"	"no"	"nc	...

n =

	0
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	...

x =

	0
0	-5
1	-4.5
2	-4
3	-3.5
4	-3
5	-2.5
6	-2
7	-1.5
8	-1
9	-0.5
10	0
11	0.5
12	1
13	1.5
14	2
15	...

f =

	0
0	25
1	20.25
2	16
3	12.25
4	9
5	6.25
6	4
7	2.25
8	1
9	0.25
10	"no"
11	"no"
12	"no"
13	"no"
14	"no"
15	...

Рис.3.5 Программирование итерационных циклов

Для проверки верности вычисления построен график заданной функции, который наглядно иллюстрирует поведение функции в трех диапазонах определения аргумента x. Выражены эти диапазоны в количестве шагов итерации *n*. Пока $x<0$ (до $n=9$) идет расчет функции по формуле x^2, в другом диапазоне $x \in 0..2$ (n от 10 до 15),функция не определена, а при $x>2$ ($n>15$) – функция рассчитывается по формуле $25 \cdot sin(x)$.

3.5 Содержание отчета

1. Название и цель работы.
2. Задания, блок-схемы и *ScreenShots* выполнения заданий, вставленные в документ, созданный в редакторе *MS Word*.
3. Выводы относительно методов программной реализации циклических процессов.

3.6 Контрольные вопросы

1. Опишите назначение тех элементов палитры *Программирование*, которые используются при решении задач итерационной циклической структуры.
2. Какие операторы используются в системе *Mathcad* для организации циклических процессов и каково их назначение?
3. Каковы назначение и формат записи функции *until*()?
4. Какой циклический процесс называется итерационным? В чём его отличия от цикла с заданным числом повторений?
5. Каково условие выхода из итерационного цикла при вычислении суммы или произведения членов бесконечного ряда?
6. Какие действия реализуются при выполнении циклических операторов?
7. Какие средства палитры программирования позволяют выделить тело цикла?
8. В чём преимущества использования операторов циклов в программе?
9. Как организуется вычисление сумм элементов бесконечного сходящегося ряда?
10. Как обращаться к программе-функции без параметров?

4. Алгоритмы регулярной циклической структуры

4.1 Цель работы

Научиться программировать циклические структуры типа прогрессия разными способами в системе *Mathcad*.

4.2 Подготовка к работе

По указанной литературе изучить:

– организацию циклов с известным числом повторений;
– возможности ранжированной переменной для организации регулярных циклов;
– возможности системы программирования для организации циклов с известным числом повторений с помощью оператора *for*…;
– формат записи и область применения операторов прерывания, продолжения и обработки ошибки.

4.3 Задание и порядок выполнения работы

1. Создать заголовок документа «Программирование регулярных циклических структур».

2. Задача 1. Используя оператор цикла *for*, вычислить сумму и произведение (в зависимости от номера варианта) конечных рядов, заданных в табл. 4.1. Вывести значение заданной функции. Проверить полученный результат с помощью калькулятора.

Таблица 4.1 Варианты заданий

N	Задача 1		
	Функция 1	*a*	*b*
1	$z=\prod_{k=1}^{8}(\cos k+\sin bk)$	-	2
2	$f=\sum_{k=1}^{10}(\cos ak+\sin b)$	2,8	1,5
3	$y=\prod_{i=1}^{8}a^2\cos(i+b)$	1,2	2

4	$u = \sum_{n=1}^{7} n \sin^2 n^2 a$	1,5	-
5	$z = \prod_{n=2}^{9} (\cos an + n)$	5	-
6	$t = \sum_{n=1}^{9} b \sin na$	2,5	2,1
7	$y = \sum_{k=5}^{12} (\ln ak + k \ln b)$	2	1,3
8	$v = \prod_{n=3}^{8} \cos na^2 b^2$	0,6	1,2
9	$y = \sum_{k=1}^{10} tga^2 k$	1,6	-
10	$p = \prod_{n=5}^{11} \cos bn$	-	1,8
11	$f = \sum_{n=2}^{5} \cos^2 an$	2,8	-
12	$t = \prod_{z=5}^{15} z^2 \cos(b+1)$	-	5,1
13	$z = \sum_{k=1}^{10} (\cos ak + \sin^2 b)$	1,5	0.25
14	$v = \prod_{n=1}^{15} \sin(na^2 b^2)$	0.8	2,1
15	$t = \sum_{n=1}^{12} x \sin(n \cdot a)$	2,5	-
16	$y = \prod_{k=1}^{9} a^2 \cos(k^2 + b)$	1,5	1,8

3. Задача 2. Написать программу-функцию решения задачи по предложенному варианту в таблице 4.2. По всем вариантам задан одномерный массив из произвольных целых чисел.

Таблица 4.2 Варианты заданий

N	Имя массива	Задача2
1	*G(16)*	Элементы одномерного массива сдвинуть на 2 позиции влево
2	*A(10)*	Подсчитать количество четных положительных элементов массива *A*
3	*B(12)*	Вычислить среднее арифметическое значение отрицательных элементов массива *B*.
4	*C(10)*	Найти сумму s элементов массива *C*. Вычислить ($s - c_1$, $s - c_2$, …, $s - c_{10}$). Результат вывести в одной строке.
5	*H(14)*	Отсортировать одномерный массив по убыванию.
6	*D(10)*	Найти произведение p элементов массива *D*. Вычислить ($p + d_1$, $p + d_2$, …, $p + d_{10}$). Результат вывести в виде столбца.
7	*M(16)*	Подсчитать количество положительных чисел, находящихся между максимальным и минимальным значениями.
8	*F(12)*	Вычислить максимальный элемент массива сформированных пар ($f_1 + f_{12}$, $f_2 + f_{11}$, … , $f_6 + f_7$).
9	*L(16)*	Переписать в массив *M* подряд положительные элементы массива *L*.
10	*K(12)*	Вычислить минимальный элемент массива сформированных пар ($k_1 \cdot k_7$, $k_2 \cdot k_8$, … , $k_6 \cdot k_{12}$).
11	*P(12)*	Переписать в массив *Q* подряд отрицательные элементы массива *P*.
12	*F(10)*	Вычислить среднее арифметическое значение m элементов массива F. Вычислить ($f_1 + m$, $f_2 + m$, …, $f_{10} + m$).
13	*N(14)*	Подсчитать количество нечетных, отрицательных элементов массива N.
14	*H(16)*	Подсчитать количество отрицательных чисел, находящихся между максимальным и минимальным значениями.
15	*J(8)*	Элементы одномерного массива циклически сдвинуть на 3 позиции вправо
16	*Z((14)*	Отсортировать одномерный массив по возрастанию.

4.4 Методические указания

Как известно, циклические алгоритмы (или проще циклы) содержат повторяющиеся вычисления, зависящие от некоторой

переменной. Такая переменная называется параметром цикла, а сами повторяющиеся вычисления составляют тело цикла.

Циклы можно условно разделить на две группы:

– циклы типа арифметической прогрессии;
– итерационные циклы.

Характерной чертой первой группы циклов является то, что количество повторений тела цикла можно определить до начала выполнения программы, реализующей цикл, т.е. априори. Классическим примером цикла типа арифметической прогрессии является накопление конечных сумм и произведений элементов конечного ряда (Задача 1).

Для итерационных циклов нельзя априори определить количество повторений тела цикла. Это обусловлено тем, что окончание таких циклов определяется не выходом параметра цикла за конечное значение, а более сложными условиями.

С помощью средств системы *Mathcad* можно задать различные виды циклов (в том числе вложенных), реализовать различные итерационные и рекурсивные процедуры. Программирование в среде *Mathcad* осуществляется с помощью создания программных модулей.

Набор программных элементов для создания программных модулей ограничен и содержит следующие операторы, размещенные на палитре *Программирование*:

– оператор *for*, который служит для организации циклов с заданным числом повторений. Формат его записи:

for Имя переменной ∈ Nmin . . Nmax

Выполняемое выражение записывается на место, расположенное ниже шаблона :

for ▪ ∈ ▪
▪

Например, **for** n ∈ 0 **. .** N-1

Переменная n меняется с шагом +1 от значения *Nmin*=0 до значения *Nmax*=*N* – 1.

Совместно с оператором *for* в модульном программировании при организации циклических процессов применяются следующие операторы:

– оператор *break*. Вызывает прерывание работы программы всякий раз, как он встречается. Используется в основном совместно с операторами условного выражения *if* и цикла *while* и *for*;

– оператор *continue*. Используется для продолжения работы после прерывания программы и, обычно, совместно с операторами задания циклов *for* и *while*. Обеспечивает возвращение в точку прерывания и продолжение вычислений;

– оператор *return*. Прерывает выполнение программы и возвращает значение своего операнда, стоящего следом за ним. Например, *return* 0 *if* $x<0$ будет возвращать значение 0 при любом $x<0$.

– оператор *on error* и функция *error* обработки ошибок.

Ярким примером регулярного циклического процесса является вычисление конечных сумм и произведений. Накопление суммы происходит последовательно, начиная с начального значения, которое принимается равное нулю $S = 0$. Произведение вычисляется аналогично, но при начальном значении равным единице $P = 1$.

Рассмотрим пример вычисления функции, в состав которой входят и сумма или произведение. Такие циклы называются регулярными и при их вычислении используют задание программного модуля, в теле цикла которого и происходит накопление соответствующих величин (рис. 4.1).

В этом примере показано применение оператора for, расположенного на палитре программирования. При вычислении используют задание одного программного модуля (тела цикла) внутри другого – основной программы вычисления. Для нескольких модулей, которые должны выполняться в составе циклов, надо использовать их объединение, отображаемое жирной вертикальной чертой. Дополнительная вертикальная черта для другого модуля добавляется оператором *Add Line*.

Тело цикла, содержащее более одного оператора, заключается тоже под общей вертикальной чертой, которая добавляется оператором *Add Line*.

На рис. 4.2 представлена блок схема, реализующая процесс накопления суммы ряда.

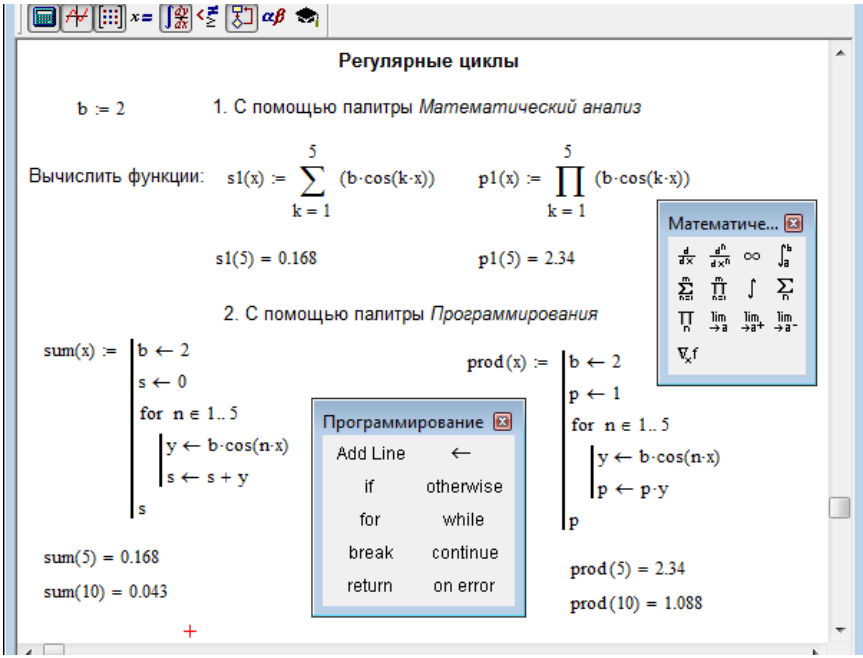


Рис. 4.1 Программирование регулярных циклов

Накопление суммы (произведения) происходит постепенно:

$s_0 = 0$ // начальное значение
$s_1 = s_0 + b \cdot cos(1 \cdot x)$
$s_2 = s_1 + b \cdot cos(2 \cdot x)$
$s_3 = s_2 + b \cdot cos(3 \cdot x)$
$s_4 = s_3 + b \cdot cos(4 \cdot x)$
$s_5 = s_4 + b \cdot cos(5 \cdot x)$
$s = s_5$ // результат накопления

Произведение вычисляется аналогично, но при начальном значении равным единице p = 1.

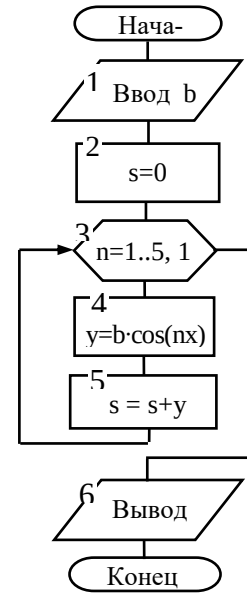


Рис. 4.2 Алгоритм решения Задачи 1

$p_0 = 1$ // начальное значение
$p_1 = p_0 \cdot b \cdot \cos(1 \cdot x)$
$p_2 = p_1 \cdot b \cdot \cos(2 \cdot x)$
$p_3 = p_2 \cdot b \cdot \cos(3 \cdot x)$
$p_4 = p_3 \cdot b \cdot \cos(4 \cdot x)$
$p_5 = p_4 \cdot b \cdot \cos(5 \cdot x)$
$p = p_5$ // результат накопления

Рассмотрим пример работы с одномерным массивом (рис. 4.3) при нахождении максимального элемента и его порядкового номера.

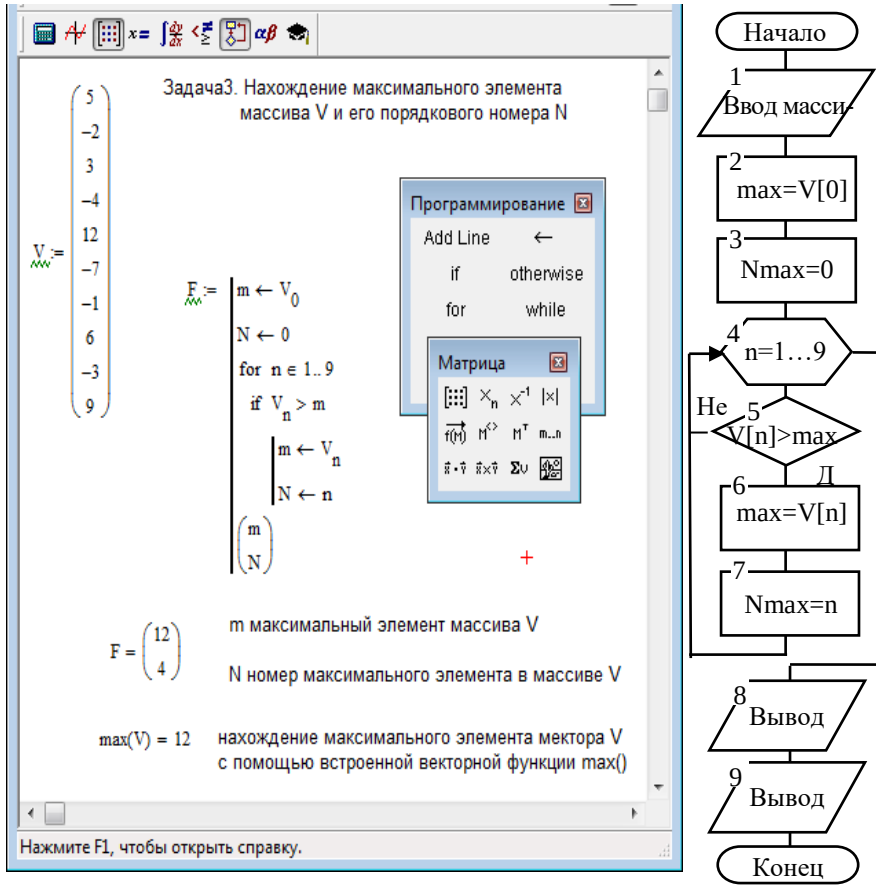


Рис. 4.3 Программирование операций с массивами

4.5 Содержание отчета

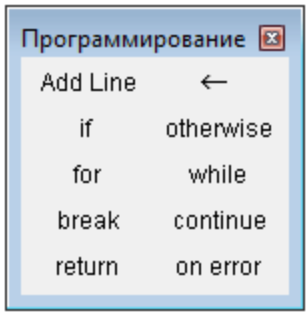
1. Название и цель работы.
2. Задания, блок-схемы и *ScreenShots* выполнения заданий, вставленные в документ, созданный в редакторе *MS Word*.
3. Выводы относительно методов программной реализации циклических процессов средствами *Mathcad* .

4.6 Контрольные вопросы

1. Опишите назначение тех элементов палитры *Программирование*, которые используются при решении задач циклической структуры.
2. Какой циклический процесс называется регулярным? В чём его отличия от цикла итерационного?
3. Какие операторы используются в системе *Mathcad* для организации циклических процессов и каково их назначение?
4. Каково условие выхода из цикла при вычислении циклической задачи?
5. Дайте понятие вложенных циклов.
6. Какие действия реализуются при выполнении циклических операторов?
7. В чём преимущества использования операторов циклов в программе?
8. Как организовать циклический процесс при шаге отличном от единицы?
9. Как организуется вычисление сумм?
10. Как организуется вычисление произведения?
11. Как из блока модуля вывести несколько параметров?

Приложение 1. Операторы программирования

Add Line	Оператор «Добавить строку».
←	Оператор локального присвоения.
if	Оператор условия.
otherwise	Оператор иначе.
for	Оператор цикла.
while	Оператор цикла.
break	Оператор останова.
continue	Оператор продолжения.
return	Оператор возврата.
on error	Оператор «ошибки».



Приложение 2. Специальные функции

Функции с условиями сравнения

Числовые

ceil(x)	наибольшее целое $\geq x$,
floor(x)	наименьшее целое $\leq x$,
mod(x,y)	остаток от *x/y* со *знаком x*,
round(x)	округление *x* до ближайшего целого.

Условных выражений

if (условие, выражение1, выражение2)

Если условие выполняется, то вычисляется выражение 1, в противном случае - выражение 2.

Итерационных вычислений

until(выражение1, выражение2)

Если выражение 1 больше или равно 0, то возвращается значение выражения 2. В противном случае, цикл прекращается.

Рекомендуемая литература

Основная

1. **Дьяконов, В. П.** *Mathcad* 11/12/13 в математике [Текст]: справочник / Дьяконов В. П. - М.: Горячая линия-Телеком, 2007 - 958 с.
2. **Кирьянов, Д. В.** *Mathcad* 15/*Mathcad Prime* 1.0 [Текст] / Кирьянов, Д. В. - СПб.: БХВ-Петербург, 2012. - 428 с. - (В подлиннике).
3. **Кирьянов, Д. В.** *Mathcad* 13 [Текст] / Кирьянов, Д. В. - СПб.: БХВ-Петербург, 2006. - 608 с.
4. **Макаров, Е. Г.** Инженерные расчеты в *Mathcad* 14 [Текст] / Макаров, Е. Г. - СПб. : Питер, 2007. - 592 с.
5. **Макаров, Е. Г.** Инженерные расчеты в *Mathcad* 15 [Текст] : учеб. курс / Макаров, Е. Г. - СПб. : Питер, 2011. - 400 с.

Дополнительная

6. **Выгодский, М. Я.** Справочник по высшей математике [Текст] / Выгодский, М. Я. - М.: АСТ, 2008. - 991 с.
7. **Очков, В. Ф.** *Mathcad* 12 для студентов и инженеров [Текст] / Очков, В. Ф. - СПб. : БХВ-Петербург, 2005. - 464 с.
8. *http://sapr-journal.ru/uroki-mathcad* – электронный ресурс. Уроки по работе в *Mathcad*.
9. *http://physics.herzen.spb.ru/library/*03/02/*mcad_progs.pdf* – электронный ресурс. Программирование в математическом пакете *Mathcad*.