Лабораторная работа № 3

Программирование в среде MathCAD

Цель работы

1. Изучить программные операторы.
2. Приобрести практические навыки составления простейших алгоритмов в среде MathCAD.

1. Общие сведения

Программный модуль в системе MathCAD представляет собой самостоятельный модуль, выделяемый в тексте документа жирной вертикальной чертой. Программный модуль может исполнять роль либо функции пользователя с именем и параметрами, либо функции без имени и параметров, но в любом случае, возвращающей результат вычислений, определяемый последним оператором модуля.

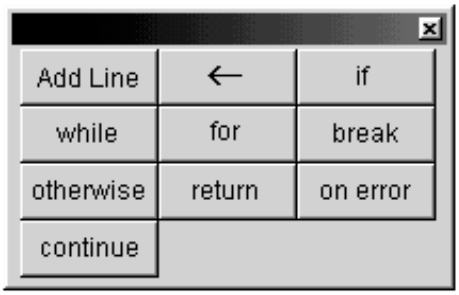

Рис. 3.1. Панель программных элементов

Шаблоны программных элементов можно вызвать при помощи кнопок панели программных элементов, показанной на рисунке 3.1.

Нетрудно заметить, что набор программных элементов для создания программных модулей весьма ограничен и содержит следующие элементы (в скобках указан перевод с английского языка названия программного элемента):

Add Line (добавить линию) – создаёт и при необходимости расширяет жирную вертикальную линию, справа от которой в шаблонах задаётся запись программного блока;
← – символ локального присваивания (в теле модуля);
if (если) – оператор условного выражения;
for (для) – оператор задания цикла с фиксированным числом повторений;
while (пока) – оператор задания цикла, типа «пока» (цикл выполняется, пока выполняется некоторое условие);
otherwise (иначе) – оператор иного выбора (обычно применяется с **if**);
break (прервать) – оператор прерывания;
continue (продолжить) – оператор продолжения;
return (возвратить) – оператор возврата;
on error (ошибка) – оператор обработки ошибок.

2. Оператор Add Line

Оператор **Add Line** выполняет функции расширения программного блока. Расширение фиксируется удлинённой вертикальной чертой программных блоков или их древовидным расширением. Благодаря этому, в принципе, можно создавать сколь угодно большие программы.

3. Оператор внутреннего присваивания ←

Оператор ← выполняет функции внутреннего локального присваивания. Например, выражение x ← 123 присваивает локальной переменной x значение 123. Локальный характер присваивания означает, что такое значение x сохраняет только в теле программного модуля. За пределами модуля значение переменной x может быть не определённым, либо равно значению, которое задаётся операторами присваивания := и ≡. В последнем случае x будет считаться глобальной переменной.

4. Оператор создания условных выражений if

Оператор **if** является оператором для создания условных выражений. Он задаётся в виде:

Выражение **if** *Условие*

Если *Условие* выполняется, то возвращается значение *Выражения*. Совместно с этим оператором часто используются операторы прерывания **break** или иного выбора **otherwise**. Например:

```
x ← 123
x ← 18 if x > 0
```

Здесь первоначально x = 123. Далее, согласно условию (x > 0), переменной x будет присвоено значение 18.

5. Оператор цикла for

Оператор **for** служит для организации циклов с заданным числом повторений. Он записывается в виде:

for *Var* ∈ *Nmin*, *Nmin* + *Step* . . *Nmax*

Эта запись означает, что если выражение, помещённое в шаблон, будет выполняться столько раз, сколько переменная *Var* изменяет своё значение от *Nmin* до *Nmax* с шагом *Step*. Если значение *Nmin* + *Step* не задано, то шаг изменения переменной, по умолчанию, принимается равным + 1. Переменную счётчика *Var* можно использовать в выражениях программы. Например:

```
for i ∈ 1, 3 .. 5
 xi←0
```

Здесь переменная i изменяется от 1 до 5 с шагом 2, т.е. принимает значения 1, 3, 5. Соответственно, в внутри оператора for $x_1 = 0$, $x_3 = 0$, $x_5 = 0$.

6. Оператор цикла while

Оператор **while** служит для организации циклов, действующих до тех пор, пока выполняется некоторое условие. Этот оператор записывается в виде:

while *Условие*

В шаблоне под оператором записывается выполняемое выражение. Например:

```
x←1
while x<5
 x←x + 1
```

Здесь в цикле будет выполняться присвоение переменной x значений 1, 2, 3, 4. Когда x = 5, то условие не выполняется и, соответственно, не выполняются операторы, следующие за **while**.

7. Оператор иного выбора otherwise

Оператор **otherwise** обычно используют совместно с оператором if. Его использование поясняет следующая программная конструкция:

```
f(x):=  1  if  x > 0        возвращает 1, если x > 0
       –1  otherwise        возвращает –1 во всех иных случаях
```

8. Оператор прерывания break

Оператор **break** вызывает прерывание работы программы. Чаще всего он используется совместно с оператором условного выражения **if** и операторами циклов **while** и **for**, обеспечивая переход в конец тела цикла. Например:

```
x←0
while 1
    x←x + 1
    break  if x > 9
```

Здесь выполнение цикла **while** прервется, когда x примет значение 10. (Конструкция **while** 1 обозначает бесконечный цикл).

9. Оператор продолжения continue

Оператор продолжения используется для продолжения работы программы после прерывания. Он также используется обычно совместно с операторами задания циклов **while** и **for**, обеспечивая после прерывания возврат в начало цикла. Например:

```
i←0
while i <5
    i←i + 1
    continue  if i = 2
    x← i
```

Здесь переменная x принимает в цикле **while** следующие значения: 1, 3, 4, 5. Когда i = 2 выполнение цикла будет прервано и произойдёт возврат в начало цикла.

10. Оператор возврата return

Оператор возврата **return** прерывает выполнение программы и возвращает значение, стоящее следом за ним. Например, в приведённом ниже случае:

return 0 **if** x < 0

будет возвращаться значение 0 при любом x < 0.

11. Оператор обработки ошибок on error и функция error

Оператор обработки ошибок позволяет создавать конструкции обработчиков ошибок. Этот оператор задаётся в виде:

Выражение № 1 **on error** *Выражение № 2*

Если при выполнении *Выражения № 2* возникает ошибка, то выполняется *Выражение № 1*. Для обработки ошибок полезна также функция **error(S)**, которая, будучи в программном модуле возвращает окошко с надписью, хранящейся в символьной переменной **S** или в символьной константе (любой фразе в кавычках). Например:

$y(x):= 1$ **on error** $\frac{1}{x}$

Здесь функция y(x) возвратит значение 1 при x = 0.

12. Практические примеры программирования

Программный модуль, в сущности, является функцией, но созданной с применением упомянутых сугубо программных средств. Она может воз-

вращать значение, определённое последним оператором. Это значит, что после такого модуля, выделенного как целый блок, можно поставить знак равенства для вывода значения функции. 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 идёт знак присваивания :=. Переменные в списке являются локальными и им можно присваивать значения при вызове функции, заданной модулем. Локальный характер таких переменных позволяет использовать для их имён (идентификаторов) те же имена, что и у глобальных переменных документа.

На рисунке 3.2. показаны примеры программирования в среде **MathCAD**. Обратите внимание на последний пример, в котором определяется количество элементов одномерного массива, которые больше 5. Здесь буква “T” справа вверху от массива обозначает транспонирование. **MathCAD** корректно производит операции только с теми одномерными массивами, которые представлены в виде вектора-столбца. Для транспонирования массива необходимо выделить его правую часть (после знака присваивания) и нажать комбинацию клавиш <**Ctrl + 1**>. В случае, если массив не содержит элементов, больших 5, то будет выведено сообщение “No”. В программных блоках переменным можно присваивать текстовые значения, написанные только латинским шрифтом. Результат работы программного блока выведен справа от него (в данном случае в одномерном массиве содержится три элемента, которые больше 5).

Задание программных модулей позволяет реализовать любые специальные приёмы программирования. Оно может служить мощным средством расширения системы путём задания новых функций.

Задание для самостоятельной работы

Создать программный модуль. Задание взять из табл. 3.1. согласно порядковому номеру студента по журналу группы.

Таблица 3.1

Варианты заданий для самостоятельной работы

№ вар.	Задание	№ вар.	Задание
1.	Создать функцию, позволяющую суммировать положительные элементы одномерного массива.	2.	Написать функцию нахождения минимального элемента массива.

№ вар.	Задание	№ вар.	Задание
3.	Создать функцию, позволяющую суммировать элементы одномерного массива, которые больше заданного числа.	4.	Написать функцию, которая находит количество отрицательных элементов одномерного массива.
5.	Создать функцию, проверяющую все ли элементы одномерного массива > 0. Функция должна выводить соответствующее сообщение.	6.	Написать функцию, которая создавала бы одномерный массив, элементами которого являлись бы положительные элементы исходного массива.
7.	Написать функцию нахождения минимального элемента массива.	8.	Создать функцию, позволяющую находить произведение элементов одномерного массива, которые меньше заданного числа.
9.	Создать функцию, позволяющую суммировать положительные элементы одномерного массива.	10.	Создать функцию, проверяющую все ли элементы одномерного массива > 0. Функция должна выводить соответствующее сообщение.
11.	Создать функцию, проверяющую все ли элементы одномерного массива < 0. Функция должна выводить соответствующее сообщение.	12.	Написать функцию, которая находит количество положительных элементов одномерного массива.
13.	Написать функцию, которая находит количество отрицательных элементов одномерного массива.	14.	Создать функцию, проверяющую все ли элементы одномерного массива > 0. Функция должна выводить соответствующее сообщение.
15.	Написать функцию нахождения максимального элемента массива.	16.	Создать функцию, позволяющую суммировать отрицательные элементы одномерного массива.

Требования к отчёту

Отчет о лабораторной работе должен включать цель работы, кратко оформленный реферат первого раздела, описание команд меню **Format** из **Приложения** и протокол действий, самостоятельно выполняемых студен-

том на компьютере. Рабочий документ выполнения лабораторной работы должен быть сохранён на ПЭВМ в личной папке студента.

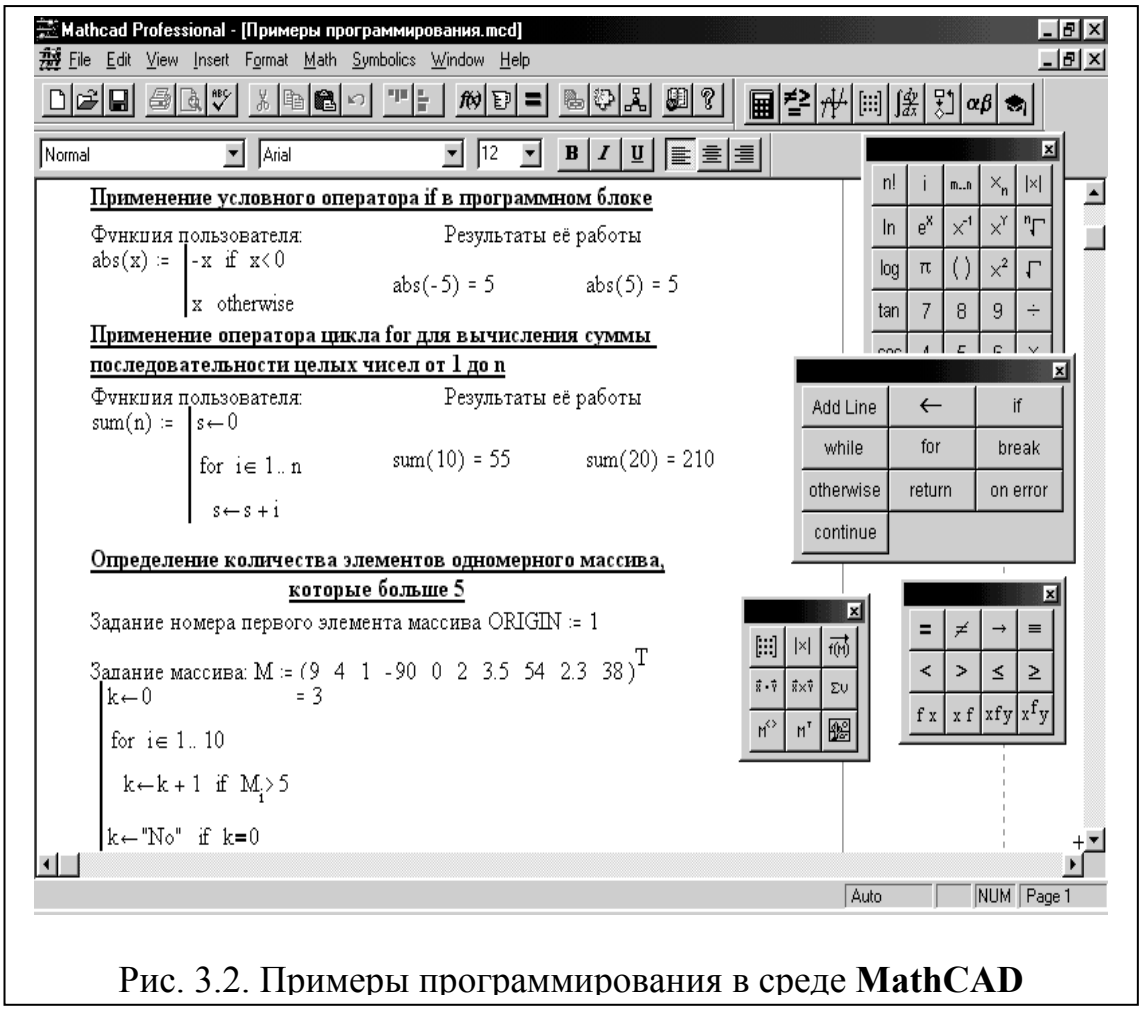

Рис. 3.2. Примеры программирования в среде **MathCAD**

При сдаче работы студент должен продемонстрировать практическое умение программировать в среде **MathCAD** и ответить на следующие контрольные вопросы:

1. Что представляет собой программный модуль в системе **MathCAD**?
2. Какие функции выполняет программный модуль?
3. Какие кнопки находятся на панели программных элементов?
4. Какие операторы цикла реализованы в **MathCAD**?
5. Что такое локальные и глобальные переменные?
6. Для чего необходим оператор **for**? Дать пример использования.
7. Какую функцию выполняет оператор **otherwise**? Дать пример использования.
8. Какую функцию выполняет оператор **if**? Дать пример использования.
9. Для чего необходим оператор **while**? Дать пример использования.
10. Как расширить программный блок?
11. Как установить формат чисел, например, вместо трёх чисел после запятой выводить пять?