**Уровень вложенности функции более чем один**

**function sbfun3(x,y)**

**% иллюстрация основной функции**

**% и нескольких подфункций в одном файле**

**s=10;**

**d=20;**

**fsum(x,y)**

**fdif(x,y)**

**function fsum(a,b)**

**fsum=(a+b)+(s+d)**

**end**

**function fdif(a,b)**

**fdif=(a-b)+(s-d)**

**fmul(x,y)**

**function fmul(a,b)**

**fmul=a\*b+s\*d**

**end**

**end**

**end**

**Свойства функций**

1. Функция может обратиться к вложенной функции, но не может использовать вложенную функцию более низкого уровня.
2. Функция может обратиться к функции того же уровня.
3. Функция нижнего уровня может вызвать функцию верхнего уровня, в которую она вложена, и все функции, доступные из нее.

**function sss(x,y)**

**a=x**

**ss1;**

**d=ss3(y)**

**% ошибка**

**%ss\_1\_1;**

**function ss1**

**b=a+5**

**ss\_1\_1;**

**function ss\_1\_1**

**h=ss2(y)**

**end**

**end**

**function [c]=ss2(z)**

**c=z-5;**

**end**

**function [d]=ss3(z)**

**d=y+10-ss2(z);**

**end**

**end**

**Замечание:** Переменные, определенные во внешней функции, доступны во вложенной функции и наоборот.

**function www**

**a=10**

**www1**

**function www1**

**www2**

**c=10**

**function www2**

**d=a**

**e=a\*a**

**end**

**end**

**b=a\*c\*e**

**end**

**Встраиваемые и анонимные функции**

Встраиваемая функция определяется с помощью функции **inline()** и представляет собой формулу для вычисления выражения:

**имя\_функции = inline('формула', 'список\_аргументов'),**

где формула – текстовая строка, каждый аргумент в апострофах.

**fun=inline('sin(x)+cos(x)','x')**

**y=fun(pi)**

Если во встраиваемой функции используется аргумент, который **явно не описан** в строке аргументов, но есть в рабочем пространстве, функция его **не «видит»!**

**x=5**

**y=10**

**% вариант с ошибкой**

**fun=inline('sin(x)+cos(y)','x')**

**z=fun(pi)**

**% Вариант без ошибки:**

**x=5**

**y=10**

**fun=inline('sin(x)+cos(y)','x','y')**

**z=fun(pi,2)**

Анонимная функция использует указатель **@**:

**имя\_функции = @(список аргументов) формула**

Анонимной функции доступны переменные рабочей среды, используемые в формуле. Но они являются **константами**, то есть имеют те значения, которые были при создании функции и далее не меняются.

**Пример:**

**x=1**

**y=pi**

**c=5**

**fun=@(x) c+sin(x)+cos(y)**

**z=fun(pi)**

**c =10**

**z=fun(pi)**

**Создание указателя на функции («именованные») и анонимные**

1. **h\_f=@NameFun;**

**h\_f=str2func('NameFun');**

1. **h\_f=@ x DefFun(x));**

**h\_f=str2func('@ x DefFun(x)');**

**Пример:**

% именованные функции

**h\_f=@ sin**

**h\_f(pi/2)**

**h\_f = @sin**

**ans = 1**

**h\_f=str2func('sin')**

**h\_f(pi/2)**

**h\_f = @sin**

**ans = 1**

% анонимные функции

**h\_f=@ (x) x\*2+3**

**h\_f(7)**

**h\_f = @(x)x\*2+3**

**ans = 17**

**h\_f=str2func('@ (x) x\*2+3')**

**h\_f(7)**

**ans = 17**

# Операторы отношения и операции

**Операторы отношения**

В системе MATLAB определено 6 операторов отношения:

|  |  |  |
| --- | --- | --- |
| **Операция** | **Знак** | **Функция** |
| Меньше | **<** | **lt()** |
| Меньше или равно | **<=** | **le()** |
| Больше | **>** | **gt()** |
| Больше или равно | **> =** | **ge()** |
| Равно | **==** | **eq()** |
| Не равно | **~ =** | **ne()** |

Операторы отношения выполняют поэлементное сравнение двух массивов равных размерностей. Для векторов и прямоугольных массивов, оба операнда должны быть одинакового размера, за исключением случая, когда один из них скаляр. В этом случае MATLAB сравнивает скаляр с каждым элементом другого операнда. Позиции, где это соотношение истинно, получают значение 1, где ложно - 0. В результате получаем ***логические массивы***.

Операторы отношения всегда выполняются ***поэлементно*.**

**Пример:**

**A=[ 4 2 5 6; 3 5 2 7; 8 1 4 9]**

**B=[ 3 5 1 2; 4 2 3 7; 5 1 3 7]**

**C=A<B**

**C=A<=B**

**C=A>=B**

**C=A>=B**

**C=A==B**

**C=A~=B**

**Логические операторы**

В состав логических операторов системы MATLAB входят следующие операторы:

Пусть даны два вектора:

**A=[0 1 1 0 1];**

**B=[1 1 0 0 1];**

|  |  |  |  |
| --- | --- | --- | --- |
| **Оператор** | **Знак** | **Функция** | **Пример** |
| Поэлементное “И” | & | and() | A&B= 0 1 0 0 1 |
| Поэлементное “ИЛИ” | | | or() | A|B= 1 1 1 0 1 |
| Логическое отрицание | ~ | not() | ~A=1 0 0 1 0 |
| Исключающее “ИЛИ” |  | xor() | xor(A,B)=1 0 1 0 0 |

**Замечание**: В некоторых случаях значение логического выражения определяется значением первого операнда (для скаляров). Операции && и || работают именно таким образом.

**Пример**:

**a=0;**

**b=1;**

**c=a&&b** % проверяется только первый операнд

**d=a||b**

**Приоритет логических и арифметических операций**

|  |  |
| --- | --- |
| Приоритет | Операции |
| **1** | **and, or, not, xor** |
| **2** | **~** (отрицание) |
| **3** | **', ^, +,-** (транспонирование, возведение в степень, унарные +, -,) |
| **4** | **\***, / (умножение и деление) |
| **5** | **+, -** (сложение и вычитание) |
| **6** | **>, >=, <, <=, ==** (операции отношения) |
| **7** | **&** (логическое «и») |
| **8** | **|** (логическое «ИЛИ») |
| **9** | **&&** (логическое «и») |
| **10** | **||** (логическое «ИЛИ») |

**Замечание**: Для изменения приоритета надо применять круглые скобки.

**Пример**: % проверить работу и объяснить результат

**a=1**;

**b=0**;

**c=2**;

**d=and(a,b)+c**

**e=a&b+c**

**d =**

**2**

**e =**

**1**

**Организация в командной строке диалога с пользователем**

|  |  |
| --- | --- |
| **disp** | **Вывод значений переменных и текста на экран** |
| **input** | **Ввод пользователем информации (с клавиатуры)** |
| **keyboard** | **Переключение на работу с клавиатурой из М-файла** |
| **pause** | **Приостановка выполнения М-файла** |
| **menu** | **Создание меню** |

**Формат функции disp():**

**disp(<переменная>|<'текст'>)**

Рассмотрим стандартное использование функции disp()

**Пример:**

**disp(' Это переменная'); %** На экран будет выведено: Это переменная

**x=7;**

**disp(x) %** На экран будет выведено: 7

Чтобы вывести значения **нескольких переменных** в одну строку необходимо создать объект, который содержал бы все эти значения. Это можно сделать, объединив соответствующие переменные в вектор:   
**x = [x1 x2 ...xN].**   
Тогда вывод значений нескольких переменных в одну строку будет иметь вид:   
**disp ([x1 x2 ...xN]).**

**Пример**: % вывод вектора значений  
**x1=-3; x2=0.5; x3=7.7;**

**x=[x1 x2 x3];**

**disp([x1 x2 x3])**

**disp(x)**

Для одновременного вывода символьной и цифровой информации в командное окно надо использовать функцию **sprintf()**

**Пример**:  **>> disp(sprintf('X1 = %f X2= %f',x1,x2))**

**X1 = -3.000000 X2= 0.500000**

Аналогично работает функция **fprintf()**, где в качестве первого параметра для вывода информации в командное окно используется 1 (либо, первый параметр просто пропускается)

**Пример**:   
**>> fprintf(1,'X1 = %f X2= %f\n',x1,x2)**

**X1 = -3.000000 X2= 0.500000**

**>> fprintf('X1 = %f X2= %f\n',x1,x2)**

**X1 = -3.000000 X2= 0.500000**

Для интерактивного взаимодействия можно использовать функцию **input()**

**Формат** функции **input():**

1. x=input('Введите значение x '); % переменной **x** присваивается введенное значение
2. str=input(' Ввод строки: ','s'); % строковой переменной **str** присваивается **Ввод строки**
3. input(' 1-я строка\n 2-я строка \n'); % вывод многострочного текста

**1-я строка**

**2-я строка**

**М-файл** может передать управление клавиатуре по команде

**keyboard**

Для продолжения работы М-файла надо набрать в командном окне команду **dbcont**. Используется, как правило, при отладке файла.

**x=10;**

**y=20;**

**z=x\*y**

**keyboard**

**% x=100**

**% dbcont**

**z=x\*y**

Функция **pause()** приостанавливает выполнение М-файла

1. **pause**  % ожидает нажатие любой клавиши
2. **pause(x)** %задержка выполнения на x секунд

Создание меню – диалогового окна с возможностью выбора варианта.

**Формат команды**:

**>> m=menu('Выбор варианта',' Вариант 1',' Вариант 2',' Вариант 3' )**

# Форматный ввод-вывод данных

Пусть данные для вектора **x** хранятся в файле \**work\x.txt**, а для **y** в файле **\work\y.txt**.

Данные записываются через пробел.

Для считывания из файла в MATLAB используется команда **load**, для записи команда – **save**

Используем функциональный способ вызова этих команд с выходными аргументами:

**x = load('x.txt');**

**y = load('y.txt');**

**save 'a.txt' a -ascii %** вывод в файл a.txt данных из переменной **a** в текстовом формате

**Спецификатор типов:**

|  |  |
| --- | --- |
| **Спецификатор** | **Форма вывода** |
| **%d** | **десятичное число со знаком** |
| **%i** | **десятичное число со знаком** |
| **%u** | **беззнаковое целое десятичное число** |
| **%f** | **вещественное число с плавающей точкой** |
| **%e** | **экспоненциальная форма** |
| **%c** | **cимвол** |
| **%s** | **cтрока символов** |
| **%o** | **восьмеричное число** |
| **%x** | **шестнадцатиричное число** |
| **%ld** | **long int – длинное целое** |
| **%lf** | **long double – вещественное число с двойной точностью** |

**Замечание**: Возможно задание в спецификаторе ширины поля:

**%nd %n.mf %ne %ns**

Запись в файл и считывание из файла осуществляется по строкам.

Для открытия файла используется **fopen().**

Формат **fopen():**

**Пример**:

**[F,mes]=fopen(filename,'rt');** % в F записывается идентификатор файла,

% строка с информацией об ошибке в mes

% disp(mes)

Функцией **fopen()** управляют три основных параметра:

Имя файла, который следует открыть (**filename**).

Описание того, как должен использоваться файл

“**r**” – чтение существующего файла;

“**w**’ – запись;

“**a**” – добавление в конец файла (если файла нет, он создается);

“**r+**” – чтение и запись в файл (файл должен существовать);

“**w+**” – чтение и запись в файл (если файл уже существует, то его содержимое уничтожается);

“**a+**” – чтение и добавление в файл.

Кроме этого различают еще текстовый (**t**) и двоичный (**b**) потоки. Если при открытии файла не указан тип потока, то по умолчанию файл обрабатывается системой как текстовый файл.

Указатель на файл (**F**). Его значение возвращается функцией и с этого момента программа ссылается на файл при помощи указателя, а не по имени файла.

**Пример:** %Вывод матрицы случайных чисел в текстовый файл

%Каждый раз при открытии файла производится удаление данных (флаг – w)

**[F,mes]=fopen('randmatr.txt','wt');**

**R=rand(3);**

**disp(R)**

**fprintf(F,'Вывод матрицы случайных чисел\n\n');**

**fprintf(F,'| %7.4f | %7.4f | %7.4f |\n',R);**

**fclose(F);**

**Задание:**

Проработать формат %[-]n.mf при разных значениях n

При открытии файла удаление данных не происходит (флаг – a)

К данным добавляются новые данные вывода

**[F,mes]=fopen('randmatr.txt','at');**

**Чтение двоичных данных из файла**

**Функции**:

**fopen()** – открыть файл;

**fclose()** – закрыть файл;

**fseek()** – установить позицию;

**freed()** – чтение бинарных данных из файла;

**fwrite()** – запись бинарных данных в файл;

**frewind()** – переход на начало файла;

**ftell()** – возвращает текущую позицию;

**fgetl()** – получение следующей строки текстового файла без символа перевода строки;

**fgets()** – получение следующей строки текстового файла c символом перевода строки;

**fprintf()** – форматный вывод в текстовый файл;

**fscanf()** – форматное чтение данных из текстового файла;

**feof()** – проверка достижения конца файла;

**sprintf()** – форматная запись данных в строку;

**scanf()** – форматное чтение данных из строки;

**dlmread()** – чтение числовых данных с разделителями из текстового файла в матрицу;

**dlmwrite()** – запись матрицы в тестовый файл с разделителями;

**textread()** – импорт данных из текстового файла, имеющего табличную структуру.

# Операторы языка MATLAB

Входной язык MATLAB насчитывает 9 операторов.

|  |  |
| --- | --- |
| **Формат оператора** | **Выполнение** |
| 1. **var=expr** | Оператор присваивания |
| 1. **if условие\_1**   **операторы\_1**  **[elseif условие\_2**  **операторы\_2**  **elseif условие\_3**  **операторы\_3**  **………………….**  **else**  **операторы ]**  **end** | Условный оператор. Если справедливо условие\_1, то выполняются операторы\_1, Если справедливо условие\_2, то выполняются операторы\_2, … Если все указанные условия являются ложными, то выполняются операторы, расположенные между else и end. |
| 1. **switch expr**   **case val1**  **операторы\_1**  **case val2**  **операторы\_2**  **…………………..**  **[otherwise**  **операторы]**  **end** | Переключатель по значению выражения expr. |
| **4. for var=e1:[e2:]e3**  **операторы**  **end** | Цикл типа арифметической прогрессии, в котором переменная var при каждом повторении тела цикла изменяется от начального значения e1 с шагом e2 до конечного значения e3. |
| **5. while условие**  **операторы**  **end** | Цикл с предусловием, выполняется до тех пор, пока истинно указанное условие. |
| **6. try**  **операторы\_1**  **catch**  **операторы\_2**  **end** | Попытка выполнить группу операторы\_1. При условии, что в результате их выполнения возникает исключительная ситуация, управление передается группе операторы\_2. Если ошибка не возникла, то группа операторы\_2 не выполняется. |
| **7.break** | Досрочный выход из управляющих конструкций типа for, while, switch, try-catch |
| **8. function**  **[y1,y2,…]=f(x1,x2, …)** | Заголовок функции, x1,x2, … - входные параметры, y1,y2, … - выходные |
| **9. return** | Досрочный выход из тела функции |

**Обработка исключительных ситуаций**

% Обработка ввода правильности имени открываемого файла

i=1;

while (i==1)

try

i=0;

name\_file=input('Введете имя файла (прерывание Ctrl+c): ','s');

feval('type',name\_file);

catch

disp('Файл не найден!!!');

i=1;

end

end

**Замечание:**

1. Функция eval() выполняет команду, которую записывают в строку – аргумент функции eval()

**Пример**:

x=pi

y=eval('sin(x)+cos(x)')

x = 3.1416

y = -1.0000

1. Имя функции можно записать в строку и выполнить с помощью функции feval()

**Пример**:

y=feval('cos',pi)

y = -1

**Условный оператор** используется в следующем виде:

|  |  |  |
| --- | --- | --- |
| **if** <условие> | **if** <условие> | **if** <условие> |
| <команды> | <команды> | <команды> |
| **end** | **else** | **elseif** <условие> |
|  | <команды> | <команды> |
|  | **end** | **else** |
|  |  | <команды> |
|  |  | **end** |

**Пример**:% Формирование трехдиагональной матрицы

n=4;

for i=1:n

for j=1:n

if i==j

A(i,j)=2;

elseif abs(i-j)==1

A(i,j)=1;

else

A(i,j)=0;

end

end

end

A

**Пример**: % Ввод и вывод, оператор switch

input\_num=input('Введите значение, -1,0,1 или другое ')

switch input\_num

case -1

disp('минус один')

case 0

disp('ноль')

case 1

disp('плюс один')

otherwise

disp('другое значение')

end

**Пример:** % Вычисление значений функции sin(x) через разложение в ряд Тейлора(1685-1731)

% (Маклорена(1698-1746))



Рекуррентная формула:

, где .

**Замечание**: Радиус сходимости рассматриваемого ряда:, так как , а



**Рациональная техника программирования в Matlab**

Работа этой программы состоит из двух частей.

Первая часть начинается с вызова функции tic - таймера, затем вычисляются значения, и вызывается функция toc - вывести

на экран время в секундах, прошедшее с момента включения таймера.

Вторая часть функции начинается вновь с включения таймера. Затем в цикле for вычисляются значения. Обе части функции делают одно и тоже, но, используя разные операторы.

clear all

tic

x=rand(100000,1);

s=sum(x);

toc

tic

s=0;

for i=1:100000

c(i)=rand(1);

s=s+c(i);

end

toc