Лабораторная работа 4
Работа со строками и текстовыми файлами в MatLab

Работа со строками

Переменные, содержащие строки, будем называть строковыми переменными. Для ограничения строки используются апострофы, например, **оператор присваивания**

```
>> str='Hello, World!'
```

приводит к образованию строковой переменной

```
str =
Hello, World!
```

Убедитесь, что строковая переменная `str` действительно является массивом, обратившись к ее элементам: **`str(8), str(1:5).`** Команда `whos` позволяет получить подробную информацию о `str`, в частности, `str` хранится в виде вектор-строки, ее длина равна 13. Поскольку строковые переменные являются массивами, то к ним применимы некоторые функции и операции, рассмотренные нами ранее.

Длина строковой переменной, т.е. число символов в ней, находится при помощи функции **`length`**.

Допустимо **сцепление строк** как вектор-строк с использованием квадратных скобок. Создайте еще одну строковую переменную `str1`, содержащую текст ' My name is Igor.', и осуществите **сцепление**:

```
>> strnew=[str str1]
strnew =
Hello, World! My name is Igor.
```

Сцепление строк может быть проведено как с использованием квадратных скобок, так и при помощи функции **`strcat`**. Входными аргументами `strcat` являются сцепляемые строки, их число неограничено, а результат возвращается в выходном аргументе.

Функция `strcat` игнорирует пробелы в конце каждой строки. Для строк: `s1='abc  '` и `s2='def'` сцепления `s=[s1 s2]` и **`s=strcat(s1,s2)`** приведут к разным результатам.

Поиск позиций вхождения подстроки в строку производится при помощи функции **`findstr`**. Ее входными аргументами являются строка и подстрока, а выходным — вектор позиций, начиная с которых подстрока входит в строку, например:

```
>> s='abcbcddefbcc';
>> s1='bc';
>> p=findstr(s,s1)
p =
2 4 10
```

Порядок входных аргументов не важен, подстрокой всегда считается аргумент меньшего размера.

Функция **`strcmp`** предназначена для **сравнения двух строк**, которые указываются во входных аргументах: **`strcmp(s1,s2)`**. Результатом является логическая единица или ноль. Функция `strncmp` сравнивает только часть строк от первого символа до символа, номер которого указан в третьем входном аргументе: `strncmp(s1,s2,8)`. Данные функции имеют аналоги: `strcmpi` и `strncmpi`, которые проводят сравнение без учета регистра, т.е., например, символы `'A'` и `'a'` считаются одинаковыми.

Для **замены в строке** одной подстроки на другую служит функция **`strrep`**. Пусть, например, требуется заменить в строке `str` подстроку `s1` на `s2` и записать обновленную строку в `strnew`. Тогда вызов функции `strrep` должен выглядеть так: **`strnew=strrep(str,s1,s2).`** Здесь важен порядок аргументов.

Преобразование всех букв строки в строчные (прописные) производит функция **lower (upper),** например: `lower('MatLab')` приводит к `'matlab'`, а `upper('MatLab')` — к `'MATLAB'`.

То, что строки являются массивами символов, позволяет легко писать собственные файл-функции обработки строк. Предположим, что требуется переставить символы в строке в обратном порядке. Файл-функция, приведенная на листинге 1, решает поставленную задачу.

Листинг 1. Файл-функция для перестановки символов в строке

```
function sout=strinv(s)
L=length(s);
for k=1:L
sout(L-k+1)=s(k);
end
```

Перейдем теперь к изучению массивов строк. Можно считать, что массив строк является вектор-столбцом, каждый элемент которого есть строка, причем длины всех строк одинаковы. В результате получается прямоугольная матрица, состоящая из символов. Например, для переменных `s1='March'`, `s2='April'`, `s3='May'`, операция `S=[s1; s2]` является допустимой, и приводит к массиву строк:

```
S =
March
April
```

Аналогичное объединение трех строк `S=[s1; s2; s3]` вызовет вывод сообщения об ошибке. Можно, конечно, дополнить при помощи сцепления, каждую строку пробелами справа до длины наибольшей из строк и затем производить формирование массива.

Функция **char** как раз и решает эту задачу, создавая из строк разной длины массив строк:

```
>> S=char(s1,s2,s3)
S =
March
April
May
```

Проверьте при помощи команды `whos`, что `S` является массивом размера 3 на 5. Для определения размеров массива строк, так же, как и любых массивов, используется `size`.

Обращение к элементам массива строк производится аналогично обращению к элементам числового массива — при помощи индексирования числами или двоеточием, например: **`S(3,2), S(2,:), S(2:3,1:4)`**. При выделении строки из массива строк в конце могут остаться пробелы. Если они не нужны, то их следует удалить, воспользовавшись функцией **deblank**. Сравните, к чему приводят `S(3,:)` и `deblank(S(3,:))`.

Поиск подстроки в массиве строк выполняется функцией `strmatch`. Входными аргументами **strmatch** являются подстрока и массив строк, а выходным — вектор с номерами строк, содержащих подстроку:

```
>> p=strmatch('Ma',S)
p =
1
3
```

Если требуется искать номера строк, точно совпадающих с подстрокой, то следует задать третий дополнительный входной аргумент `'exact'`.

Некоторые функции обработки строк могут работать и с массивами строк. Например, при сцеплении массивов с одинаковым числом строк при помощи `strcat`, сцепление применяется к соответствующим строкам массивов, образуя новый массив

строк. Среди входных аргументов strcat может быть и строка. В этом случае она добавляется ко всем строкам массива.

Среди символов строки могут быть цифры и точка, т.е. строка может содержать числа. Важно понимать, что оператор присваивания b=10 приводит к образованию числовой переменной b, а оператор s='10' — строковой.

Преобразование целого числа в строку производится при помощи функции **int2str**, входным аргументом которой является число, а выходным — строка, например:

```
>> str=['May, ' int2str(10)]
str =
May, 10
```

Заметьте, что нецелые числа перед преобразованием округляются.

Для перевода нецелых чисел в строки служит функция **num2str**:

```
>> str=num2str(pi)
str =
3.1416
```

Дополнительный второй входной аргумент num2str предназначен для указания количества цифр в строке с результатом: **str=num2str(pi,11).**

Возможно более гибкое управление преобразованием при помощи строки специального вида, определяющей формат (экспоненциальный или с плавающей точкой) и количество позиций, отводимых под число. Строка с форматом задается в качестве второго входного аргумента num2str, начинается со знака процента и имеет вид **'%A.ax'**, где:

- A — количество позиций, отводимое под все число;
- a — количество цифр после десятичной точки;
- x — формат вывода, который может принимать, например, одно из следующих значений: f (с плавающей точкой), e (экспоненциальный) или g (автоматический подбор наилучшего представления).

Сравните результаты следующих преобразований числа в строку:

```
>> z=198.23981
>> sf=num2str(z, '%12.5f')
>> se=num2str(z, '%12.5e')
>> sg=num2str(z, '%12.5g')
```

Если входным аргументом функций int2str и num2str является матрица, то результатом будет массив строк. В этом случае в строке с форматом часто полезно указать разделитель, например запятую и пробел:

```
>> R=rand(5)
>> S=num2str(R,'%7.2e, ')
```

Обратная задача, а именно, преобразование строковой переменной, содержащей число, в числовую переменную решается при помощи функции **str2num**. Входным аргументом str2num может быть строка с представлением целого, вещественного или комплексного числа в соответствии с правилами MatLab, например:

str2num('2.9e-3'), str2num('0.1'), str2num('4.6+4i').

Работа с текстовыми файлами

MatLab предлагает достаточно универсальные способы считывания данных из текстовых файлов и записи данных в требуемом виде в текстовый файл.

Работа с текстовыми файлами состоит из трех этапов:

1. открытие файла;
2. считывание или запись данных;

3. закрытие файла.

Для **открытия файла** служит функция **fopen**, которая вызывается с двумя входными аргументами: именем файла и строкой, задающей способ доступа к файлу. **Выходным аргументом** fopen является идентификатор файла, т.е. переменная, которая впоследствии используется при любом обращении к файлу. Функция fopen возвращает –1, если при открытии файла возникла ошибка.

Существует четыре основных способа открытия файла:

1. f=fopen('myfile.dat','rt') — открытие текстового файла myfile.dat только для чтения из него;
2. f=fopen('myfile.dat','rt+') — открытие текстового файла myfile.dat для чтения и записи данных;
3. f=fopen('myfile.dat','wt') — создание пустого текстового файла myfile.dat только для записи данных;
4. f=fopen('myfile.dat','wt+') — создание пустого текстового файла myfile.dat для записи и чтения данных.

При использовании двух последних вариантов следует соблюдать осторожность — если файл myfile.dat уже существует, то его содержимое будет уничтожено. После открытия файла появляется возможность считывать из него информацию, или заносить ее в файл.

По окончании работы с файлом необходимо закрыть его при помощи **fclose(f).**

Построчное считывание информации из текстового файла производится при помощи функции **fgetl**. Входным аргументом fgetl является идентификатор файла, а выходным — текущая строка. Каждый вызов fgetl приводит к считыванию одной строки и переводу текущей позиции в файле на начало следующей строки. Команды, приведенные на листинге 2, последовательно считывают из файла myfile.dat первые три строки в строковые переменные str1, str2, str3.

Листинг 2. Считывание трех первых строк из текстового файла

```
f=fopen('myfile.dat','rt');
str1=fgetl(f);
str2=fgetl(f);
str3=fgetl(f);
fclose(f);
```

При **последовательном считывании** рано или поздно будет достигнут конец файла, при этом функция fgetl вернет минус 1. Лучше всего перед считыванием проверять, не является ли текущая позиция в файле последней. Для этого предназначена функция **feof**, которая вызывается от идентификатора файла и возвращает единицу, если текущая позиция последняя и ноль, в противном случае. Обычно последовательное считывание организуется при помощи цикла while. Листинг 3 содержит файл-функцию viewfile, которая считывает строки файла, и выводит их в командное окно.

Листинг 3. Файл-функция viewfile

```
function viewfile(fname)
f=fopen(fname,'rt');
while feof(f)==0
s=fgetl(f);
disp(s)
end
fclose(f);
```

Вызов viewfile('viewfile.m') приводит к отображению текста самой файл-функции в командном окне.

Строки записываются в текстовый файл при помощи функции **fprintf**, ее первым входным аргументом является идентификатор файла, а вторым — добавляемая

строка. Символ **\n** служит для перевода строки. Если поместить его в конец добавляемой строки, то следующая команда `fprintf` будет осуществлять вывод в файл с новой строки, а если `\n` находится в начале, то текущая команда `fprintf` выведет текст с новой строки. Например, последовательность команд (листинг 4) приведет к появлению в файле my.txt текста из двух строк (листинг 5).

Листинг 4. Вывод строк в текстовый файл

```
f=fopen('my.txt','wt');
fprintf(f,'текст ');
fprintf(f,'еще текст\n');
fprintf(f,'а этот текст с новой строки');
fclose(f);
```

Листинг 5. Результат работы операторов листинга 9.3

```
текст еще текст
а этот текст с новой строки
```

Аналогичного результата можно добиться, переместив `\n` из конца строки второй функции `fprintf` в начало строки третьей функции `fprintf`. Аргументом `fprintf` может быть не только строка, но и строковая переменная. В этом случае для обеспечения вывода с новой строки ее следует сцепить со строкой `'\n'`.

Строка, предназначенная для вывода в текстовый файл, может содержать как текст, так и числа. Часто требуется выделить определенное количество позиций под число и вывести его в экспоненциальном виде или с плавающей точкой и с заданным количеством цифр после десятичной точки. Здесь не обойтись без форматного вывода при помощи `fprintf`, обращение к которой имеет вид:

```
fprintf(идентификатор файла, 'форматы', список переменных)
```

В списке переменных могут быть как числовые переменные (или числа), так и строковые переменные (или строки).

Второй аргумент является строкой специального вида с форматами, в которых будут выводиться все элементы из списка. Каждый **формат начинается со знака процента**. Для **вывода строк** используется формат `s`, а для **вывода чисел** — `f` (с плавающей точкой) или `e` (экспоненциальный).

Число перед `s` указывает количество позиций, отводимых под вывод строки.

При выводе значений числовых переменных перед `f` или `e` ставится два числа, разделенных точкой. Первое из них означает количество позиций, выделяемых под все значение переменной, а второе — количество знаков после десятичной точки. Таким образом, под каждый из элементов списка отводится поле определенной длины, **выравнивание** в котором по умолчанию производится **по правому краю**. Для **выравнивания по левому краю** следует после знака процента поставить **знак минус**.

Ниже приведены варианты вызова `fprintf` и результаты, незаполненные позиции после вывода (пробелы) обозначены символом `o`.

```
fprintf(f,'%6.2f', pi) oo3.14
fprintf(f,'%-6.2f', pi) 3.14oo
fprintf(f,'%14.4e', exp(-5)) ooo6.7379e-003
fprintf(f,'%-14.4e', exp(-5)) 6.7379e-003ooo
fprintf(f,'%8s', 'текст') oootекст
fprintf(f,'%-8s', 'текст') текстooo
```

В случае, когда зарезервированного поля не хватает под выводимую строку или число, MatLab автоматически увеличивает длину поля, например:

```
fprintf(f,'%5.4e', exp(-5)) 6.7379e-003
```

При одновременном выводе чисел и текста пробелы могут появляться из-за неполностью заполненных полей, выделенных как под числа, так и под строки. Приведенные ниже операторы и результат их выполнения демонстрируют расположение

полей в строке текстового файла. Волнистой линией подчеркнуты поля, выделенные под числа, а прямой — под строки, символ ○ по-прежнему обозначает пробелы.

```
x=0.55;
fprintf(f,'%-5s%6.2f%6s%20.8e','x=',x,'y=',exp(x))
```

x=○○○○○**0.55**○○○○**y=**○○○○○**1.73325302e+000**

Форматный вывод удобен при формировании файла с таблицей результатов. Предположим, что необходимо записать в файл f.dat таблицу значений функции $f(x) = x^2 \sin x$ для заданного числа *n* значений $x \in [a,b]$, отстоящих друг от друга на одинаковое расстояние. Файл с таблицей значений должен иметь такую структуру, как показано на листинге 6.

Листинг 6. Текстовый файл с таблицей значений функции

```
f(0.65)=2.55691256e-001
f(0.75)=3.83421803e-001
f(0.85)=5.42800093e-001
f(0.95)=7.34107493e-001
f(1.05)=9.56334106e-001
```

Очевидно, что следует организовать цикл от начального значения аргумента до конечного с шагом, соответствующим заданному числу точек, а внутри цикла вызывать `fprintf` с подходящим списком вывода и форматами. Символ `\n`, предназначенный для перевода строки, помещается в конец строки с форматами (см. листинг 7) .

Листинг 7. Файл-функция tab, выводящая таблицу значений функции

```
function tab(a,b,n)
h=(b-a)/(n-1);
f=fopen('f.dat','wt');
for x=a:h:b
fprintf(f,'%2s%4.2f%2s%15.8e\n', 'f(',x,')=',x^2*sin(x))
end
fclose(f);
```

Обратимся теперь к **считыванию данных из текстового файла**. Функция **`fscanf`** осуществляет обратное действие по отношению к `fprintf`. Каждый вызов `fscanf` приводит к занесению данных, начинающихся с текущей позиции, в переменную. Тип переменной определяется заданным форматом. В общем случае, обращение к `fscanf` имеет вид:

```
a=fscanf(идентификатор файла, 'формат', число считываемых
элементов)
```

Для считывания строк используется формат `%s`, для целых чисел — `%d`, а для вещественных — `%g`.

Предположим, что файл exper.dat содержит текст, приведенный на листинге 7. Данные отделены друг от друга пробелами. Требуется считать дату проведения эксперимента (число, месяц и год) в подходящие переменные, а результаты занести в числовые массивы `TIME` и `DAT`.

Листинг 8. Файл с данными exper.dat

```
Результаты экспериментов 10 мая 2002
t= 0.1 0.2 0.3 0.4 0.5
G= 3.02 3.05 2.99 2.84 3.11
```

Считывание разнородных данных (числа, строки, массивы) требует контроля, который достигается применением форматов. Первые два элемента файла exper.dat являются строками, следовательно можно сразу занести их в одну строковую переменную `head`. Очевидно, надо указать формат `%s` и число 2 считываемых элементов. Далее требуется считать целое число 10, строку `'мая'` и снова целое число 2002. Обратите внимание, что перед заполнением массива `TIME` еще придется предусмотреть считывание

строки 't='. Теперь все готово для занесения пяти вещественных чисел в вектор-столбец TIME при помощи формата %g. Данные из последней строки файла exper.dat извлекаются аналогичным образом (листинг 9).

Листинг 9. Применение fscanf для считывания данных

```
f=fopen('exper.dat','rt');
head=fscanf(f,'%s',2)
data=fscanf(f,'%d',1)
month=fscanf(f,'%s',1)
year=fscanf(f,'%d',1)
str1=fscanf(f,'%s',1)
TIME=fscanf(f,'%g',5)
str2=fscanf(f,'%s',1)
DAT=fscanf(f,'%g',5)
fclose(f);
```

Информация в текстовом файле может быть представлена в виде таблицы. Для считывания такой информации следует использовать массив структур с подходящим набором полей. Считывание всей информации реализуется в цикле while, в условии которого производится проверка на достижение конца файла при помощи функции feof. Функция fscanf предоставляет возможность считывать числа из текстового файла не только в вектор-столбец, но и массив заданных размеров. Расположение чисел по строкам в файле не имеет значения. Размеры формируемого массива указываются в векторе в третьем входном аргументе fscanf. Рассмотрим считывание числовых данных из файла matr.txt (листинг 10).

Листинг 10. Текстовый файл matr.txt с матрицей

```
1.1 2.2 3.3 4.4
5.5 6.6 7.7 8.8
0.2 0.4 0.6 0.8
```

Функция fscanf формирует столбец матрицы, последовательно считывая числа из файла (т. е. построчно). Следовательно, для заполнения матрицы с тремя строками и четырьмя столбцами, необходимо считать данные из файла в массив четыре на три, а затем транспонировать его (листинг 11).

Листинг 11. Считывание матрицы из текстового файла

```
f=fopen('matr.txt');
M=fscanf(f,'%g', [4 3]);
M=M'
fclose(f);
```

Обратите внимание, что массив может иметь размеры не только 3 на 4. Поскольку данные считываются подряд, то они могут быть занесены и в массив 2 на 6, и 4 на 3 и т. д.

Пример m-программы, создающей html-файл «00v01 отчет.htm» для отображения результатов расчетов

```
clear
close all
clc
% вычисление
tic  %включить таймер

%указать папку, где сохранять рисунки и html-файл
File1NameExp='v01';
PictHigh1='200';    %высота рисунка на html-странице
```

```
DirName1=['e:/' File1NameExp];
    %create new folder for данных
if ~exist(DirName1)
    mkdir(DirName1);
end;
cd(DirName1);

%открыть html-файл для создания отчета
FileHtml=fopen(['00' File1NameExp ' отчет.htm'],'wt');

   %Заголовок html-файла
   fprintf(FileHtml,['<HTML>\n']);
   fprintf(FileHtml,['<head>\n']);
   fprintf(FileHtml,['<title>Отчет по лабораторной работе \n']);
   fprintf(FileHtml,['</title>\n']);
   fprintf(FileHtml,['<META HTTP-EQUIV="Content-Type"
CONTENT="text/html; charset=windows">\n']);
   fprintf(FileHtml,['</head>\n']);
   fprintf(FileHtml,['<body>\n']);

   fprintf(FileHtml,['<center><H3>ОТЧЕТ<br>по лабораторной
работе</h3></center>\n']);

% листинг программы или другая информация
    n1=1;
    n2=2;
   fprintf(FileHtml,['<h3>Исходные данные</h3>\n']);

       fprintf(FileHtml,'<br>%-5.0f%-5.0f\n',n1,n2);

% листинг программы или другая информация

% создать графическое окно
h1=figure('Name','Модельное изображение (исходное Image0)
im3d');
hold on

%выполнить расчеты
x=[-2:0.01:2];
for beta=-0.5:0.1:0.5
y=exp(beta*x).*sin(x);
plot(x,y)
end
hold off

%сохранить построенные графики в файл
FileName000=[File1NameExp '_1.jpg'];
saveas(h1,FileName000);

% полученный графический файл необходимо разместить на html-
странице
```

```
    %создается html-файл
    fprintf(FileHtml,['<br><br> Графики \n']);
           fprintf(FileHtml,['<a href="' FileName000 '">']);
    fprintf(FileHtml,['<br><img src="'...
        FileName000 '" height="' PictHigh1 '" >' '\n']);
           %fprintf(FileHtml,['<a href="' FileName000 '">']);
           fprintf(FileHtml,['</a>\n']);
    fprintf(FileHtml,['</body>\n']);
    fprintf(FileHtml,['</HTML>\n']);
    fclose(FileHtml);

disp('Вычисления завершены');
toc %вывести показания таймера
```

После завершения работы m-программы будет создан следующий html-файл «00v01 отчет.htm»:

```
<HTML>
<head>
<title>Отчет по лабораторной работе
</title>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows">
</head>
<body>
<center><H3>ОТЧЕТ<br>по лабораторной работе</h3></center>
<h3>Исходные данные</h3>
<br>1    2
<br><br> Графики
<a href="v01_1.jpg"><br><img src="v01_1.jpg" height="200" >
</a>
</body>
</HTML>
```

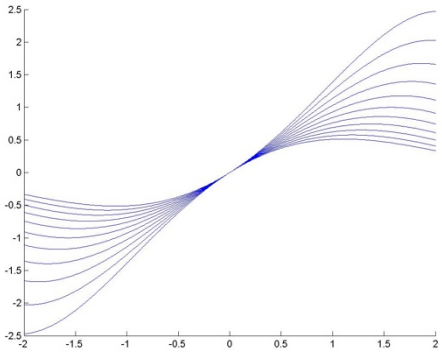
При открытии html-файла «00v01 отчет.htm» на экране отображается следующая информация

ОТЧЕТ
по лабораторной работе

Исходные данные

1 2

Графики



Замечание:
Миниатюрная копия графика является ссылкой на окно, в которой график отображается в масштабе 1:1.

Задания лабораторной работы

Написать файл-функцию для решения поставленной задачи.

Задачи:

1. Определить количество символов в первой строке варианта без учета пробелов.
2. Первая строка является предложением, в котором слова разделены пробелами. Переставить первое и последнее слово.
3. Заменить в первой строке цифры числительными (вместо 1, 2,… — один, два, три,…).
4. Задана строка (первая строка варианта), содержащая текст и числа, разделенные пробелами, выделить числа в числовой массив.

5. Записать данные, указанные в соответствующем варианте, в файл inN.txt, где N – номер варианта.
6. Считать матрицы и вектора из файла в подходящие по размеру массивы. Обратите внимание, что в файлах содержится рядом две или три матрицы или вектора, их следует занести в разные массивы.
7. Построить график функции $y = \sin(Nx)$, где N – номер варианта, $x = [1:0.1:5]$. Представить результаты вычислений и график в виде html-файла «lab3_N.html», график сохранить в файле с именем «varN», где N – номер варианта.

Варианты:

1	Алексеев Сергей 1980 5 4 4 5 3 5 0.1 0.2 0.3 9.91 1.9 0.4 0.1 8.01 4.7 5.1 3.9 7.16	2	23 марта 2002 0.36 0.32 0.28 0.25 1.399 2.001 9.921 3.21 0.12 0.129 1.865 8.341 9.33 8.01 9.136 8.401 7.133 3.12 3.22
3	Иванов Константин 1981 3 4 3 4 3 5 1 2 3 4 99 80 5 6 7 8 33 21 15 90	4	Time= 0.0 0.1 0.2 0.3 0.4 0.5 0.6 10 20 40 50 12 19 21 32 44 -1 -2 -3 -4 32 10
5	195251 СПб Политехническая 29 1 2 3 4 100	6	Петров Олег 1980 5 5 5 4 4 5 2.1 0.2 0.3 9.91

	6 7 8 9 0.1 0.2 0.3 0.4 200 0.5 0.6 0.7 0.8 300		1.9 0.4 0.1 8.01 7.7 5.1 3.9 5.16
7	195256 СПб Науки 49 4.79 2.001 9.921 3.21 0.25 1.129 1.865 8.341 9.33 8.01 8.136 8.401 7.133 3.12 3.44	8	Результаты 2.1 2.3 2.3 1.9 1.8 2.4 1 2 3 4 99 80 5 6 7 8 33 21 15 90
9	Сидоров Николай 101 521 899 10 20 40 50 12 19 21 32 44 -1 -2 -3 -4 32 10	10	Тимофеев Сергей 570 100 203 1 2 3 4 100 6 7 8 9 0.1 0.2 0.3 0.4 200 0.5 0.6 0.7 0.8 57

Пример m-программы, создающей html-файл «00v01 отчет.htm» для отображения результатов расчетов

```
clear
close all
clc
% вычисление
tic  %включить таймер

%указать папку, где сохранять рисунки и html-файл
File1NameExp='v01';
PictHigh1='200';    %высота рисунка на html-странице

DirName1=['e:/' File1NameExp];
    %create new folder for данных
if ~exist(DirName1)
    mkdir(DirName1);
end;
cd(DirName1);

%открыть html-файл для создания отчета
FileHtml=fopen(['00' File1NameExp ' отчет.htm'],'wt');

   %Заголовок html-файла
   fprintf(FileHtml,['<HTML>\n']);
   fprintf(FileHtml,['<head>\n']);
   fprintf(FileHtml,['<title>Отчет по лабораторной работе \n']);
   fprintf(FileHtml,['</title>\n']);
   fprintf(FileHtml,['<META HTTP-EQUIV="Content-Type"
CONTENT="text/html; charset=windows">\n']);
   fprintf(FileHtml,['</head>\n']);
   fprintf(FileHtml,['<body>\n']);

   fprintf(FileHtml,['<center><H3>ОТЧЕТ<br>по лабораторной
работе</h3></center>\n']);

% листинг программы или другая информация
    n1=1;
    n2=2;
   fprintf(FileHtml,['<h3>Исходные данные</h3>\n']);
```

```
        fprintf(FileHtml,'<br>%-5.0f%-5.0f\n',n1,n2);

% листинг программы или другая информация

% создать графическое окно
h1=figure('Name','Модельное изображение (исходное Image0)
im3d');
hold on

%выполнить расчеты
x=[-2:0.01:2];
for beta=-0.5:0.1:0.5
y=exp(beta*x).*sin(x);
plot(x,y)
end
hold off

%сохранить построенные графики в файл
FileName000=[File1NameExp '_1.jpg'];
saveas(h1,FileName000);

% полученный графический файл необходимо разместить на html-
странице

    %создается html-файл
    fprintf(FileHtml,['<br><br> Графики \n']);
           fprintf(FileHtml,['<a href="' FileName000 '">']);
    fprintf(FileHtml,['<br><img src="'...
        FileName000 '" height="' PictHigh1 '" >' '\n']);
           %fprintf(FileHtml,['<a href="' FileName000 '">']);
           fprintf(FileHtml,['</a>\n']);
    fprintf(FileHtml,['</body>\n']);
    fprintf(FileHtml,['</HTML>\n']);
    fclose(FileHtml);

disp('Вычисления завершены');
toc %вывести показания таймера
```

После завершения работы m-программы будет создан следующий html-файл «00v01 отчет.htm»:

```
<HTML>
<head>
<title>Отчет по лабораторной работе
</title>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows">
</head>
<body>
<center><H3>ОТЧЕТ<br>по лабораторной работе</h3></center>
<h3>Исходные данные</h3>
<br>1    2
<br><br> Графики
<a href="v01_1.jpg"><br><img src="v01_1.jpg" height="200" >
```


</body>
</HTML>

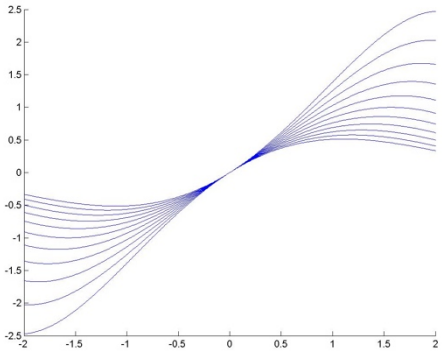
При открытии html-файла «00v01 отчет.htm» на экране отображается следующая информация

ОТЧЕТ
по лабораторной работе

Исходные данные

1 2

Графики



Замечание:
Миниатюрная копия графика является ссылкой на окно, в которой график отображается в масштабе 1:1.