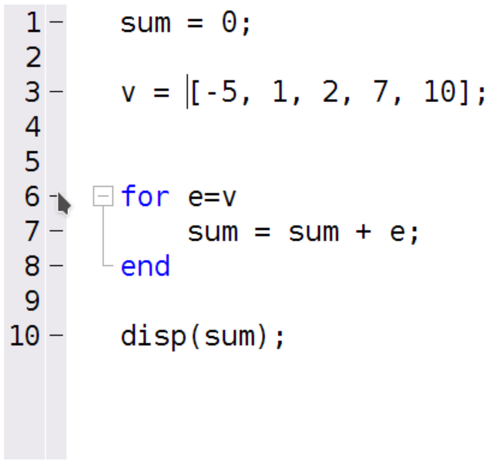
Part 1 – Debugging

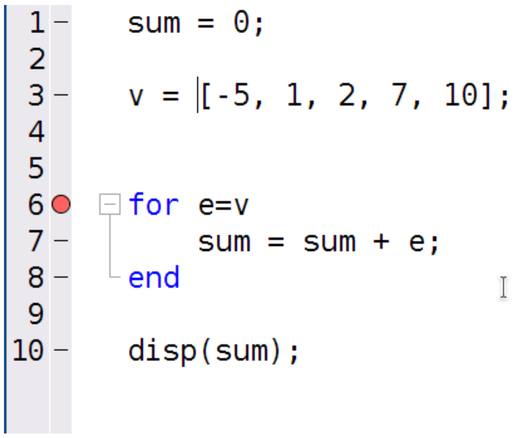
If you want to look at your workspace at some step of program execution, you always can use debugging tools. They allow you to pause program at any line and look at variables in the workspace.

1) In order to use debugging functional, you must put breakpoint. It can be done by clicking on the horizontal lines between code and line numbers. Please note that you cannot put a breakpoint at an empty line.



```
sum = 0;

v = [-5, 1, 2, 7, 10];


for e=v
    sum = sum + e;
end

disp(sum);
```

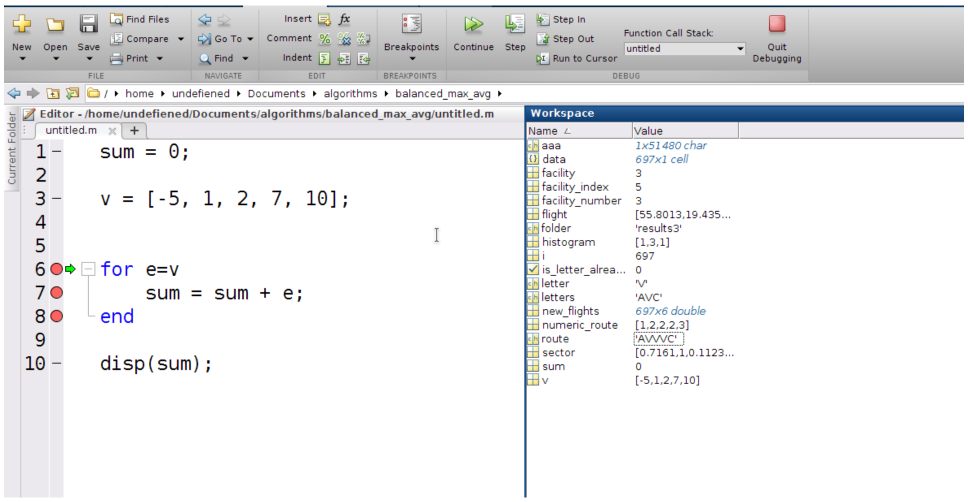
2) You should see red or gray circle after you clicked.



```
sum = 0;

v = [-5, 1, 2, 7, 10];


for e=v
    sum = sum + e;
end

disp(sum);
```

3) Now, after you click "Run" button, script execution will be paused at the line with a breakpoint. In the workspace window you can observe state of all variables.

Please note that you can use as many breakpoints as you want.



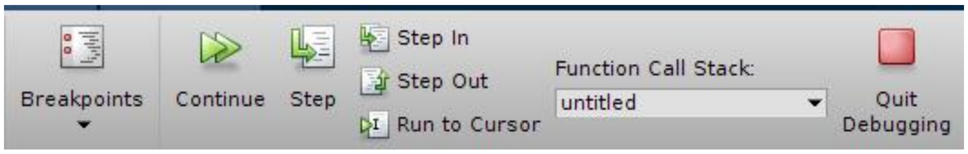
Editor - /home/undefiened/Documents/algorithms/balanced_max_avg/untitled.m

```
sum = 0;

v = [-5, 1, 2, 7, 10];


for e=v
    sum = sum + e;
end

disp(sum);
```

Workspace

Name	Value
aaa	*1x51480 char*
data	*697x1 cell*
facility	3
facility_index	5
facility_number	3
flight	[55.8013,19.435...
folder	'results3'
histogram	[1,3,1]
i	697
is_letter_alrea...	0
letter	'V'
letters	'AVC'
new_flights	*697x6 double*
numeric_route	[1,2,2,2,3]
route	'AVVVC'
sector	[0.7161,1,0.1123...
sum	0
v	[-5,1,2,7,10]

4) In order to continue code execution you should press "Continue" button.



5) Also, if you want to continue code execution line by line, instead of using "Continue" button, you can use "Step" button. It executes current line at stops at the next one.

Part 2 – The last part of the task 1

At this moment you should have function named **facility_histogram** in the file named "facility_histogram.m". You can find the function code in the folder called "Code".

Code for the last part you can find in the file named "check_routes.m" in the "Code" folder.

1) You can run **facility_histogram** for several routes (old and new one).

[letters1, histogram1] = facilities_histogram('ABCBAC');

[letters2, histogram2] = facilities_histogram('CCAABB');

2) Create new variable, and assign value **true** to it.

is_new_route_equivalent = true;

3) Now we must compare corresponding entries of **histogram1** and **histogram2**.

% In this for loop we iterate through all letters in "letters1"

for letter_index_1=1:length(letters1)

```
    letter_from_1 = letters1(letter_index_1);


    % In this loop we iterate through all letters in "letters2"
    for letter_index_2=1:length(letters2)
        letter_from_2 = letters2(letter_index_2);


        % If we have equivalent letters, then we should check corresponding
        % values of histogram
        if letter_from_1 == letter_from_2
            % If corresponding values of histogram are not equivalent, then
            % new route is bad
            if ~(histogram1(letter_index_1) == histogram2(letter_index_2))
                is_new_route_equivalent = false;
            end
        end
    end
end
```

4) In the final part we write to command window if new route is good or bad.

```
if is_new_route_equivalent
    disp('New route is good!')
else
    disp('New route is bad!')
end
```