**51.Write a c program for Swap to numbers?**

C program to swap two numbers with and [without using third variable](https://www.programmingsimplified.com/c-program-swap-two-numbers#without), using [pointers](https://www.programmingsimplified.com/c-program-swap-two-numbers#pointers), [functions (Call by reference)](https://www.programmingsimplified.com/c-program-swap-two-numbers#functions) and using bitwise [XOR](https://www.programmingsimplified.com/c-program-swap-two-numbers#XOR) operator. Swapping means interchanging. For example, if in your C program you have taken two variables a and b where a = 4 and b = 5, then before swapping a = 4, b = 5 after swapping a = 5, b = 4. In the first C program to swap numbers we will use a temporary variable to swap two numbers.

## Swapping of two numbers in C

#include <stdio.h>

 int main()

{

int x, y, temp;

printf("Enter the value of x and y**\n**");

scanf("%d%d", &x, &y);

printf("Before Swapping**\n**x = %d**\n**y = %d**\n**",x,y);

temp = x;

x = y;

y = temp;

printf("After Swapping**\n**x = %d**\n**y = %d**\n**",x,y);

return 0;

}

**Enter the value of x and y**

**4**

**5**

**Before swaping**

**X = 4**

**Y = 5**

**After swaping**

**X =5**

**Y =4**

**52.Write a c program for Fibonacci series?**

#include <stdio.h>

int main()

{

int i, n, t1 = 0, t2 = 1, nextTerm;

printf("Enter the number of terms: ");

scanf("%d", &n);

printf("Fibonacci Series: ");

for (i = 1; i <= n; ++i)

{

printf("%d, ", t1);

nextTerm = t1 + t2;

t1 = t2;

t2 = nextTerm;

}

return 0;

}

**Output**

Enter the number of terms: 10

Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,

**Example #2: Fibonacci Sequence Up to a Certain Number**

#include <stdio.h>

int main()

{

int t1 = 0, t2 = 1, nextTerm = 0, n;

printf("Enter a positive number: ");

scanf("%d", &n);

// displays the first two terms which is always 0 and 1

printf("Fibonacci Series: %d, %d, ", t1, t2);

nextTerm = t1 + t2;

while(nextTerm <= n)

{

printf("%d, ",nextTerm);

t1 = t2;

t2 = nextTerm;

nextTerm = t1 + t2;

}

return 0;

}

**Output**

Enter a positive integer: 100

Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89,

**53.How to define in macro in c program?**

You can define a macro in C using #define preprocessor directive.

A macro is a fragment of code that is given a name. You can use that fragment of code in your program by using the name. For example,

#define c 299792458 // speed of light

Here, when we use c in our program, it's replaced with 299792458

**54.what are the workspace in matlab window?**

The base workspace stores variables that you create at the command line. This includes any variables that scripts create, assuming that you run the script from the command line or from the Editor. Variables in the base workspace exist until you clear them or end your MATLAB® session

Functions do not use the base workspace. Every function has its own function workspace. Each function workspace is separate from the base workspace and all other workspaces to protect the integrity of the data. Even local functions in a common file have their own workspaces. Variables specific to a function workspace are called **local**variables. Typically, local variables do not remain in memory from one function call to the next.

When you call a script from a function, the script uses the function workspace.

Like local functions, nested functions have their own workspaces. However, these workspaces are unique in two significant ways:

* Nested functions can access and modify variables in the workspaces of the functions that contain them.
* All of the variables in nested functions or the functions that contain them must be explicitly defined. That is, you cannot call a function or script that assigns values to variables unless those variables already exist in the function workspace

The Workspace browser shows the name of each variable, its value, its array size, its size in bytes, and the class. The icon for each variable denotes its class.

**55.what is workspace,why want use?**

The Workspace browser enables you to view and interactively manage the contents of the workspace in MATLAB®. For each variable or object in the workspace, the Workspace browser also can display statistics, when relevant, such as the minimum, maximum, and mean.

You can edit the contents of scalar (1-by-1) variables directly in the Workspace browser. Right-click the variable and select **Edit Value**. To edit other variables, double-click the variable name in the Workspace browser to open it in the Variables editor.

**Open the Workspace Browser**

To open the Workspace browser if it is not currently visible, do one of the following:

* MATLAB Toolstrip: On the **Home** tab, in the **Environment** section, click **Layout**. Then, in the **Show** section, select **Workspace**.
* MATLAB command prompt: Enter workspace

**56.what is multidimensional array,why want use?**

Basically multi dimension arrays are used if you want to put arrays inside an array.

Say you got 10 students and each writes 3 tests. You can create an array like: arr\_name[10][3]

So, calling arr\_name[0][0] gives you the result of student 1 on lesson 1. Calling arr\_name[5][2] gives you the result of student 6 on test 3.

You can do this with a 30 position array,.

**1) easier to understand**

**2) easier to debug**.

**57.what is difference between in cell array and structure array?**

**Cell vs. Struct Arrays**

Try This Example

This example compares cell and structure arrays, and shows how to store data in each type of array. Both cell and structure arrays allow you to store data of different types and sizes.

**Structure Arrays**

Structure arrays contain data in fields that you access by name.

For example, store patient records in a structure array.

patient(1).name = 'John Doe';

patient(1).billing = 127.00;

patient(1).test = [79, 75, 73; 180, 178, 177.5; 220, 210, 205];

patient(2).name = 'Ann Lane';

patient(2).billing = 28.50;

patient(2).test = [68, 70, 68; 118, 118, 119; 172, 170, 169];

patient

patient = *1x2 struct array with fields:*

name

billing

test

Create a bar graph of the test results for each patient.

numPatients = numel(patient);

for p = 1:numPatients

figure

bar(patient(p).test)

title(patient(p).name)

xlabel('Test')

ylabel('Result')

end

**Cell Arrays**

Cell arrays contain data in cells that you access by numeric indexing. Common applications of cell arrays include storing separate pieces of text and storing heterogeneous data from spreadsheets.

For example, store temperature data for three cities over time in a cell array.

temperature(1,:) = {'2009-12-31', [45, 49, 0]};

temperature(2,:) = {'2010-04-03', [54, 68, 21]};

temperature(3,:) = {'2010-06-20', [72, 85, 53]};

temperature(4,:) = {'2010-09-15', [63, 81, 56]};

temperature(5,:) = {'2010-12-09', [38, 54, 18]};

temperature = *5x2 cell array*

{'2009-12-31'} {1x3 double}

{'2010-04-03'} {1x3 double}

{'2010-06-20'} {1x3 double}

{'2010-09-15'} {1x3 double}

{'2010-12-09'} {1x3 double}

**58.Simulink blocks support multidimensional array, if yes how to support?**

## Signal Dimensions

### About Signal Dimensions

Simulink® blocks can output one-dimensional, two-dimensional, or multidimensional signals. The Simulink user interface and documentation generally refer to 1-D signals as vectorsand 2-D or multidimensional signals as matrices. A one-element array is frequently referred to as a scalar. A row vector is a 2-D array that has one row. A column vector is a 2-D array that has one column.

* A one-dimensional (1-D) signal consists of a series of one-dimensional arrays output at a frequency of one array (vector) per simulation time step.
* A two-dimensional (2-D) signal consists of a series of two-dimensional arrays output at a frequency of one 2-D array (matrix) per block sample time.
* A multidimensional signal consists of a series of multidimensional (two or more dimensions) arrays output at a frequency of one array per block sample time. You can specify multidimensional arrays with any valid MATLAB® multidimensional expression, such as [4 3]. See [Multidimensional Arrays](https://ww2.mathworks.cn/help/matlab/math/multidimensional-arrays.html) (MATLAB) for information on multidimensional arrays.

Simulink blocks vary in the dimensionality of the signals they can accept or output. Some blocks can accept or output signals of any dimension. Some can accept or output only scalar or vector signals. To determine the signal dimensionality of a particular block, see the block documentation. See [Determine Output Signal Dimensions](https://ww2.mathworks.cn/help/simulink/ug/determining-output-signal-dimensions.html) for information on what determines the dimensions of output signals for blocks that can output nonscalar signals.

### Note

Simulink does not support dynamic signal dimensions during a simulation. That is, the dimension of a signal must remain constant while a simulation is executing. However, you can change the size of a signal during a simulation. See [Variable-Size Signal Basics](https://ww2.mathworks.cn/help/simulink/ug/variable-size-signal-basics.html).

### Simulink Blocks that Support Multidimensional Signals

The Simulink Block Data Type Support table includes a column identifying the blocks with multi-dimension signal support.

1. In the Simulink editor, from the **Help** menu, click **Simulink** > **Block Data Types & Code Generation Support** > **All Tables**.

A separate window with the Simulink Block Data Type Support table opens.

1. In the Block column, locate the name of a Simulink block. Columns to the right are data types or features. An a **X** in a column indicates support for that feature.

Simulink supports signals with up to 32 dimensions. Do not use signals with more than 32 dimensions.

**59.what is difference between in m-file and normal file?**

An m-file, or script file, is a simple text file where you can place MATLAB commands. When the file is run, MATLAB reads the commands and executes them exactly as it would if you had typed each command sequentially at the MATLAB prompt. All m-file names must end with the extension '.m' (e.g. test.m). If you create a new m-file with the same name as an existing m-file, MATLAB will choose the one which appears first in the path order (type help path in the command window for more information). To make life easier, choose a name for your m-file which doesn't already exist. To see if a filename.m already exists, type help filename at the MATLAB prompt.

**60.why want use mex file?**

 MEX file is a function, created in MATLAB, that calls a C, C++, or Fortran subroutine. To call a MEX file, use the name of the file, without the file extension. The MEX file contains only one function or subroutine, and its name is the MEX file name. The file must be on your MATLAB path

**61.what is s-function, types?**

**S-Functions and Code Generation**

An S-function is a computer language description of a Simulink block written in MATLAB®, C, C++, or Fortran. C, C++, and Fortran S-functions are compiled as MEX files using the mex utility (see [Build MEX File](https://in.mathworks.com/help/matlab/matlab_external/build-an-executable-mex-file.html) (MATLAB)). As with other MEX files, S-functions are dynamically linked subroutines that the MATLAB execution engine can automatically load and execute.

S-functions use a special calling syntax called the S-function API that enables you to interact with the Simulink engine. This interaction is very similar to the interaction that takes place between the engine and built-in Simulink blocks.

S-functions follow a general form and can accommodate continuous, discrete, and hybrid systems. By following a set of simple rules, you can implement an algorithm in an S-function and use the S-Function block to add it to a Simulink model. After you write your S-function and place its name in an S-Function block (available in the User-Defined Functions block library), you can customize the user interface using masking (see [Create Block Masks](https://in.mathworks.com/help/simulink/block-masks.html)).

If you have Simulink Coder™, you can use S-functions with the software. You can also customize the code generated for S-functions by writing a Target Language Compiler (TLC) file. For more information, see [S-Functions and Code Generation](https://in.mathworks.com/help/rtw/ug/s-functions-and-code-generation.html) (Simulink Coder).

### How S-Functions Work

To create S-functions, you need to understand how S-functions work. Such knowledge requires an understanding of how the Simulink engine simulates a model, including the mathematics of blocks. This section begins by explaining the mathematical relationships between the inputs, states, and outputs of a block.

**Tasks performed by S-function callback methods include**:

* Initialization — Prior to the first simulation loop, the engine initializes the S-function, including:
  + Initializing the SimStruct, a simulation structure that contains information about the S-function
  + Setting the number and dimensions of input and output ports
  + Setting the block sample times
  + Allocating storage areas
* Calculation of next sample hit — If you created a variable sample time block, this stage calculates the time of the next sample hit; that is, it calculates the next step size.
* Calculation of outputs in the major time step — After this call is complete, all the block output ports are valid for the current time step.
* Update of discrete states in the major time step — In this call, the block performs once-per-time-step activities such as updating discrete states.
* Integration — This applies to models with continuous states and/or nonsampled zero crossings. If your S-function has continuous states, the engine calls the output and derivative portions of your S-function at minor time steps. This is so the solvers can compute the states for your S-function. If your S-function has nonsampled zero crossings, the engine also calls the output and zero-crossings portions of your S-function at minor time steps so that it can locate the zero crossings.

## S-Function Types

Consider the following questions if you are unclear about what type of S-function is best for your application.

| **If you...** | **Then use...** |
| --- | --- |
| Are a MATLAB® programmer with little or no C programming experience | A Level-2 MATLAB S-function, especially if you do not need to generate code for a model containing the S-function (see [Write Level-2 MATLAB S-Functions](https://in.mathworks.com/help/simulink/sfg/writing-level-2-matlab-s-functions.html)). |
| Need to generate code for a model containing the S-function | Either a Level-2 MATLAB S-function or a C MEX S-functions. Level-2 MATLAB S-functions require that you write a Target Language Compiler (TLC) file for your S-function, before generating code. C MEX S-functions, however, automatically support code generation. |
| Need the simulation to run faster | A C MEX S-function, even if you do not need to generate code. For complicated systems, Level-2 MATLAB S-functions simulate slower than C MEX S-functions because they call out to the MATLAB execution engine. |
| Need to implement the S-function in C, but have no previous experience writing C MEX S-functions | The [S-Function Builder](https://in.mathworks.com/help/simulink/slref/sfunctionbuilder.html). |
| Are incorporating legacy code into the model | Any S-function, with the exception of a Level-1 MATLAB S-function. Consider using the Legacy Code Tool if your legacy function calculates only outputs, not dynamic states (see [Integrate C Functions Using Legacy Code Tool](https://in.mathworks.com/help/simulink/sfg/integrating-existing-c-functions-into-simulink-models-with-the-legacy-code-tool.html)). Otherwise, consider using the S-Function Builder. If you need to call the legacy code during simulation, do not use a Level-2 MATLAB S-function because they call legacy code only through their TLC files. |
| Need to generate embeddable code for an S-function that incorporates legacy code | The Legacy Code Tool if your legacy function calculates only outputs. Otherwise, use a handwritten C MEX S-function or the S-Function Builder. |

* **62.simulink model are developed in ver. 2010b,it is support to ver. 2013 b mat lab ?**

**63.write a m-script to find all constant blocks sample time?**

# get\_param

Get parameter names and values

## Syntax

ParamValue = get\_param(Object,Parameter)

## Description

[ParamValue](https://in.mathworks.com/help/simulink/slref/get_param.html#btqy0ez-ParamValue) = get\_param([Object](https://in.mathworks.com/help/simulink/slref/get_param.html" \l "btqy0ez-Object),[Parameter](https://in.mathworks.com/help/simulink/slref/get_param.html#btqy0ez-Parameter)) returns the name or value of the specified parameter for the specified model or block object. Open or load the Simulink® model first.

**64.what is configuration setting ,why want use?**

* **Algebraic loop** is set to none
* **Minimize algebraic loop** is not set to error
* **Block Priority Violation** is not set to error

The check fix action modifies the **Minimize algebraic loop** and **Block Priority Violation** settings to error.

The check uses the ex\_DataFile.xml data file created in [Create Data File for Diagnostics Pane Configuration Parameter Check](https://ww2.mathworks.cn/help/slcheck/ug/create-check-for-model-configuration-parameters.html#bt6_kfi).

Close the Model Advisor and your model if either are open.

**65.what is difference between in ''ert.tlc and ''grt.tlc in code generation?**

for the first question: grt uses a larger memory model and is a target for generic use (as the name says) mostly on the host PC. ert (embedded real-time target) is using a smaller memory model, and optimizes better for speed and memory. To generate code using ert you might just choose the ert.tlc on the Simulink Parameters page.

**66.what is wrapper function?**

A wrapper is a function or script whose implementation is only to call another function. 

|  |  |
| --- | --- |
|  | myMean = 2; % RV mean |
|  | myVariance = 3; % RV variance |
|  | myLength = 100; % Length of random vector |
|  |  |
|  | myOutput = myMean + sqrt(myVariance)\*randn(1,myLength); |

**67.what is difference between in fixed point and floating point?**

The main difference is in calculation error. In fixed point arithmetic you can put an upper margin on your absolute error, while in floating point arithmetic you can put an upper margin on your relative error.

**68.Floating point minimum and maximums value?**

|  |  |  |
| --- | --- | --- |
| FLT\_MAX | Maximum value of **float** | 3.40282347e+38F |
| FLT\_MIN | Minimum normalized positive floating-point number value of **float** | 1.175494351e-38F |

**69.difference between in switch and multiport switch,after code generation how look like?**

**70.After code generation rate transition and product blocks how look like?**

**71.which condition your using in integrated blocks?**

**72.what is difference between in delay unit delay and tapped delay blocks?**

**Tapped Delay block** Specify the initial output of the simulation. The **Initial condition** parameter is converted from a double to the input data type offline using round-to-nearest and saturation. Simulink software does not allow you to set the initial condition of this block to inf or NaN.

| **Capability** | **Memory** | **Unit Delay** | **Zero-Order Hold** |
| --- | --- | --- | --- |
| Specification of initial condition | Yes | Yes | No, because the block output at time t = 0 must match the input value. |
| Specification of sample time | No, because the block can only inherit sample time from the driving block or the solver used for the entire model. | Yes | Yes |
| Support for frame-based signals | No | Yes | Yes |
| Support for state logging | No | Yes | No |

**73.while running model it's possible to change parameter values, if yes how to change?**

yes, that should be possible as long the the parameter is marked as tunable (which it is by default, I think). You will need to run Ctrl+D on the model or set\_param('modelname', 'SimulationCommand', 'update') after changing the workspace value to ensure that Simulink notices the new value

**74.How to create array and structure in simulink model?**

Here is the workflow for creating a structure in a MATLAB Function block:

1. Decide on the type (or scope) of the structure (see [Types of Structures in MATLAB Function Blocks](https://in.mathworks.com/help/simulink/ug/types-of-structures-in-matlab-function-blocks.html)).
2. Based on the scope, follow these guidelines for creating the structure:

| **For Structure Scope:** | **Follow These Steps:** |
| --- | --- |
| Input | * 1. Create a Simulink.Bus object in the base workspace to define the structure input.   2. Add data to the MATLAB Function block, as described in [Adding Data to a MATLAB Function Block](https://in.mathworks.com/help/simulink/ug/_bqlblru.html). The data should have the following properties      + **Scope** = Input      + **Type** = Bus: <object name>   For <object name>, enter the name of the Simulink.Bus object that defines the structure input  See [Rules for Defining Structures in MATLAB Function Blocks](https://in.mathworks.com/help/simulink/ug/rules-for-defining-structures-in-matlab-function-blocks.html). |
| Output | * 1. Create a Simulink.Bus object in the base workspace to define the structure output.   2. Add data to the MATLAB Function block with the following properties:      + **Scope** = Output      + **Type** = Bus: <object name>   For <object name>, enter the name of the Simulink.Bus object that defines the structure output   * 1. Define and initialize the output structure implicitly as a variable in the MATLAB® function, as described in [Structure Definition for Code Generation](https://in.mathworks.com/help/simulink/ug/how-working-with-structures-is-different-for-code-generation.html).   2. Make sure the number, type, and size of fields in the output structure variable definition match the properties of the Simulink.Bus object. |
| Local | Define the structure implicitly as a local variable in the MATLAB function, as described in [Structure Definition for Code Generation](https://in.mathworks.com/help/simulink/ug/how-working-with-structures-is-different-for-code-generation.html). By default, local variables in MATLAB Functionblocks are temporary. |
| Persistent | Define the structure implicitly as a persistent variable in the MATLAB function. |
| Parameter | * 1. Create a structure variable in the base workspace.   2. Add data to the MATLAB Function block with the following properties:      + **Name** = same name as the structure variable you created in step 1.      + **Scope** = Parameter |

**75.What is difference between in normal signal and bus signal ?**