External Function Interface
and
External Objects

Function Declaration

The structure of a typical function declaration is as follows:

```
function <functionname>
  input  TypeI1 in1;
  input  TypeI2 in2;
  input  TypeI3 in3;
  ...
  output TypeO1 out1;
  output TypeO2 out2;
  ...
protected
  <local variables>
  ...
algorithm
  ...
  <statements>
  ...
end <functionname>;
```

All internal parts of a function are optional, the following is also a legal function:

```
function <functionname>
end <functionname>;
```

Pure Modelica functions are *declarative mathematical functions*:

- Always return the same result(s) given the same input argument values

External Functions

It is possible to call functions defined outside the Modelica language, implemented in C or FORTRAN 77

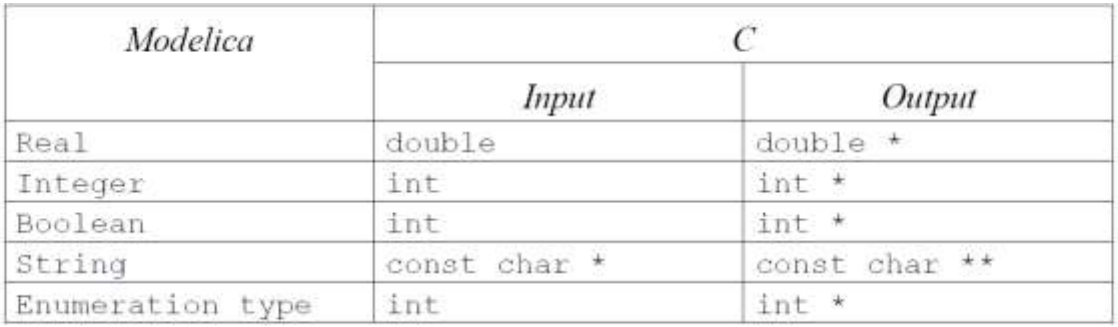
```
function polynomialMultiply
  input  Real a[:], b[:];
  output Real c[:] := zeros(size(a,1)+size(b, 1) - 1);
external
end polynomialMultiply;
```

The body of an external function is marked with the keyword **`external`**

If no language is specified, the implementation language for the external function is assumed to be C. The external function `polynomialMultiply` can also be specified, e.g. via a mapping to a FORTRAN 77 function:

```
function polynomialMultiply
  input  Real a[:], b[:];
  output Real c[:] := zeros(size(a,1)+size(b, 1) - 1);
external ”FORTRAN 77”
end polynomialMultiply;
```

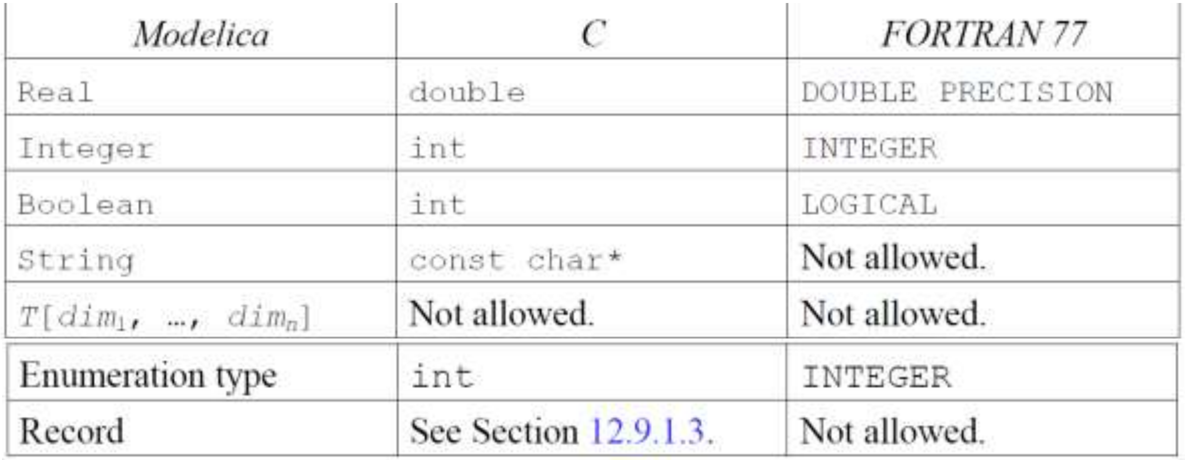
External Functions – Argument Type Mapping



Modelica	*C*	
	Input	*Output*
`Real`	`double`	`double *`
`Integer`	`int`	`int *`
`Boolean`	`int`	`int *`
`String`	`const char *`	`const char **`
`Enumeration type`	`int`	`int *`

```
record R               struct R {
  Real x;                double x;
  Real z;                double z;
end R;                 };
```

External Functions – Return Type Mapping



Modelica	*C*	*FORTRAN 77*
`Real`	`double`	`DOUBLE PRECISION`
`Integer`	`int`	`INTEGER`
`Boolean`	`int`	`LOGICAL`
`String`	`const char*`	Not allowed.
$T[dim_1, \ldots, dim_n]$	Not allowed.	Not allowed.
Enumeration type	`int`	`INTEGER`
Record	See Section 12.9.1.3.	Not allowed.

External Functions – Annotations

- Include

  annotation(Include=“include directive")
  annotation(IncludeDirectory="modelica://*LibraryName*/Resources/Include")

- Linker

  annotation(Library={"libraryName1","libraryName2"})
  annotation(LibraryDirectory="modelica://*LibraryName*/Resources/Library")

- Standardized platform names
  - win32 *[Microsoft Windows 32 bit]*
  - win64 *[Microsoft Windows 64 bit]*
  - linux32 *[Linux Intel 32 bit]*
  - linux64 *[Linux Intel 64 bit]*

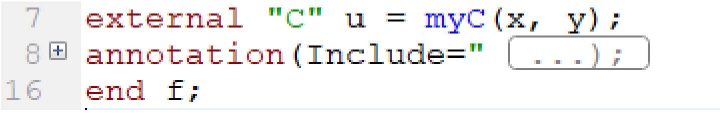
External Functions – Examples

```
// pass all outputs as arguments
function foo
  input Real x;
  input Integer y;
  output Real u1;
  output Integer u2;
  external "C" myfoo(x, u1, y, u2);
end foo;
// void myfoo(double, double *, int, int *);
```

```
// pass some outputs as arguments, return as value from function
function foo
  input Real x;
  input Integer y;
  output Real u1;
  output Integer u2;
  external "C" u1 = myfoo(x, y, u2);
end foo;
// double myfoo(double, int, int *);
```

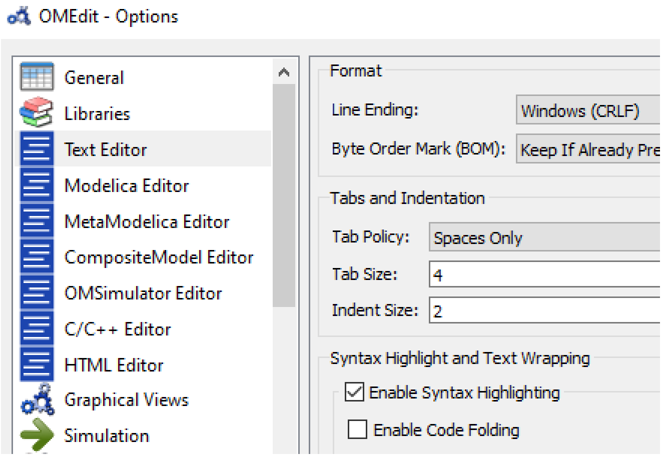
External Functions – Exercises

- Open ExternalFunctions.mo
- Complete the C definition of function f returning addition of inputs x and y
  Hint: add C code line: `return x+y;`



```
external "C" u = myC(x, y);
annotation(Include=" ...);
end f;
```

- Hint: to see the C code click on the +
  or in Tools->Options disable code folding



OMEdit - Options

Format

Line Ending: Windows (CRLF)

Byte Order Mark (BOM): Keep If Already Pre

Tabs and Indentation

Tab Policy: Spaces Only

Tab Size: 4

Indent Size: 2

Syntax Highlight and Text Wrapping

☑ Enable Syntax Highlighting

☐ Enable Code Folding

- Simulate the model that uses function f
- Optional change the external definition of function f to pass output as argument to the external C function

External Objects

- Used for C functions that need to **maintain an internal state** (opaque to Modelica)
- They are good for making sure **things are initialized once** (functions can be called multiple time during initialization and simulation). The **external object constructor is called only once** before the first use of the object

External Objects – Declaration

- Extend from ExternalObject and provide two functions: constructor and destructor

```
class MyTable
  extends ExternalObject;
  function constructor
    input String fileName := "";
    input String tableName := "";
    output MyTable table;
    external "C" table = initMyTable(fileName, tableName);
  end constructor;
  function destructor "Release storage of table"
    input MyTable table;
    external "C" closeMyTable(table);
  end destructor;
end MyTable;
```

External Objects – Exercises

- Open ExternalObjects.mo
- Fill in the external C code for `constructor` and `destructor` for the external object
- Simulate ExternalObjects.Test

```
int i=0; // Hint: you can Insert this code in Constructor

  double *extObj = (double*)malloc(size*sizeof(double));
  if(extObj == NULL)
    printf(\"Not enough memory\");

  for(i=0; i<size; i++)
    if(i < 2)
      extObj[i] = 1.0;
    else
      extObj[i] = extObj[i-1]+extObj[i-2];

  return (void*)extObj;
```

Hint, code
for constructor

Index 0 in C array
corresponds to index 1
in Modelica.

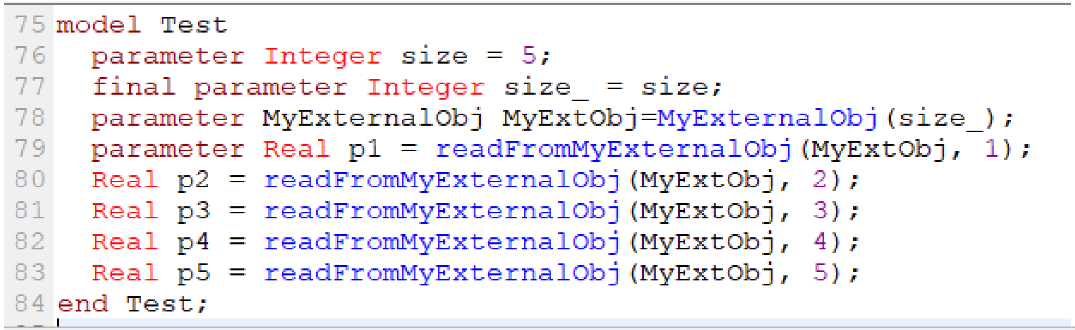
```
/* Release storage */
         double *extObj = (double*)object;
         if (object == NULL)
           return;

         free(extObj);
```

Hint, code
for destructor

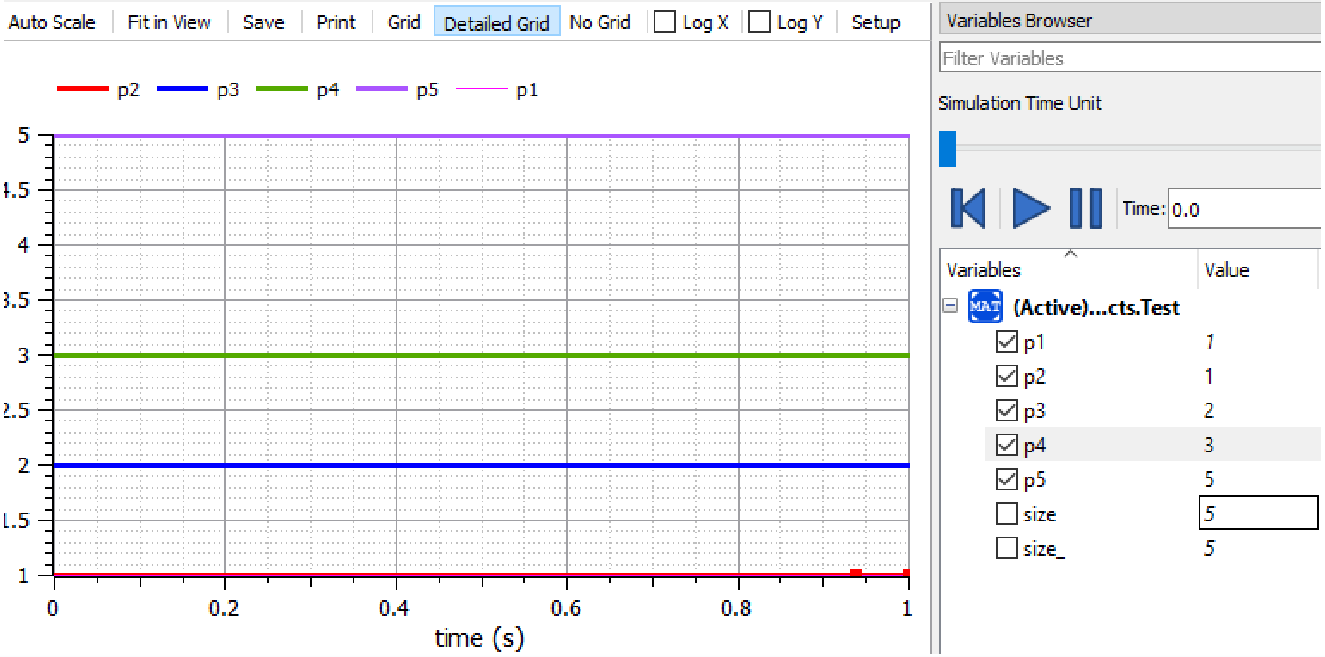
Test simulation of External Objects Example

- Test model simulation and plots. Change plot line width via plot Setup.



```
model Test
  parameter Integer size = 5;
  final parameter Integer size_ = size;
  parameter MyExternalObj MyExtObj=MyExternalObj(size_);
  parameter Real p1 = readFromMyExternalObj(MyExtObj, 1);
  Real p2 = readFromMyExternalObj(MyExtObj, 2);
  Real p3 = readFromMyExternalObj(MyExtObj, 3);
  Real p4 = readFromMyExternalObj(MyExtObj, 4);
  Real p5 = readFromMyExternalObj(MyExtObj, 5);
end Test;
```

According to the algorithm the first two elements are 1
The next are the sum of the two previous elements



Time: 0.0

Variables	Value
⊟ (Active)...cts.Test	
☑ p1	1
☑ p2	1
☑ p3	2
☑ p4	3
☑ p5	5
☐ size	5
☐ size_	5