STAT40780 Data Programming with C (online)

Lab Sheet 4 (Solutions)

Dr Marie Galligan

Summer 2015

This week's lab requires you to write some C++ functions and call them through the .C interface in R. This lab sheet will put into practice the lecture material from this week on operators in C++, as well as using the Rprintf() function.

1 Evaluating a relational expression

(a) Write a C++ function (callable from R through the .C interface) that receives two numeric scalar arguments from R, tests whether they are equal, and returns TRUE if the values are equal and FALSE if the values are not equal. Hint: the function will need to have a total of 3 input arguments - two numeric type arguments that are to be tested for equality, and an integer type argument (logical type values can be also be represented as integer types) through which a return a value of TRUE or FALSE can be passed

(b) Compile this function and call it from R through the .C interface

(c) Write a wrapper function in R for the compiled C++ function, that accepts two numeric scalars as input (and checks that the input is of the correct type), and outputs either TRUE (if its two input arguments are equal) and FALSE (if its input arguments are not equal).

Solution

The following is a possible solution to part (a):

isEqual.cpp

```cpp


extern "C" {

  void isEqual( double * x  , double * y, int * isEq)
  {

    *isEq = *x == *y ;

  }

}
```

A solution to part (b):

Compile this function. First open command prompt, and change current working directory to where the .cpp file is stored:

```
cd path/to/file
```

Compile isEqual.cpp as follows:

```
R CMD SHLIB isEqual.cpp
```

This should compile without errors or warnings, to produce either isEqual.so (OS X) or isEqual.dll (Windows).

To call this function from R through the .C interface:

Call isEqual() from R

```r
setwd("path/to/file/") #change working directory to location where
                                 #the compiled file is stored

dyn.load("isEqual.so") #load the compiled function  (for OS X)
dyn.load("isEqual.dll") #load the compiled function  (for Windows)


#call the function isEqual through the .C interface
#with equal arguments
.C("isEqual", x = as.numeric(1.1), y = as.numeric(1.1), result = TRUE)
#with non equal arguments
.C("isEqual", x = as.numeric(1.2), y = as.numeric(1.1), result = TRUE)


dyn.unload("isEqual.so") #unload compiled function (OS X)
dyn.unload("isEqual.dll") #unload compiled function (Windows)
```

A solution to part (c):

Wrapper function around isEqual() in R

```r
setwd("path/to/file/") #change working directory to location where
                                 #the compiled file is stored

dyn.load("isEqual.so") #unload the compiled function  (for OS X)
dyn.load("isEqual.dll") #unload the compiled function  (for Windows)


#write a wrapper function in R that takes two numeric input arguments
#returns a single value, TRUE or FALSE,
#depending on whether its input arguments are equal or not

isEqual <- function( x, y ){

  if( ! ( is.numeric(x) & is.numeric(y) ) ){
    stop("input arguments must be numeric")
  }

  if(  ( length(x) > 1) |  ( length(y) > 1 ) ){
    stop("input arguments must be scalars")
  }

  return( .C("isEqual", x = x, y = y, out = FALSE)$out )

}


#call C++ through the wrapper function
isEqual( 2.7, 2.7 )
isEqual( 2.9, 2.7 )


dyn.unload("isEqual.so") #unload compiled function (OS X)
dyn.unload("isEqual.dll") #unload compiled function (Windows)
```

2 Printing output to R

Write a C++ function (callable through the .C interface in R) that accepts as input an integer argument from R, divides it by 3 and prints the output in the R terminal. Compile this function, write a wrapper function in R, and call the function from R.

Solution

A possible solution for the C++ function that divides an integer argument (from R) by 3, and prints the result to the R terminal.

Note the use of the `static_cast<double>()` operator to convert the integer type to a double type.

divideBy3.cpp

```
extern "C" {

  void divideBy3( int * x ){

    Rprintf("x divided by 3 is equal to %f\r\n", static_cast<double>( *x )/3);

  }

}
```

Compile this function at the command line. Shown below is a possible solution for the task of calling the divideBy3() function from R through the .C interface, with the help of a wrapper function.

Call divideBy3() from R (through a wrapper function)

```
setwd("path/to/file/") #change working directory to location where
                                  #the compiled file is stored

dyn.load("divideBy3.so") #unload the compiled function  (for OS X)
dyn.load("divideBy3.dll") #unload the compiled function  (for Windows)

#write a wrapper function

divideBy3 <- function(x){

  #check that x is an integer scalar
  #if not, break out of function using stop()
  if(!is.integer(x) | ( length(x) != 1 ) ){
    stop("x must be an integer scalar")
  }

  result <- .C("divideBy3", as.integer(x))  #call compiled C++ function

}

#call the C++ function through the R wrapper
divideBy3(3)
divideBy3(4)

dyn.unload("divideBy3.so") #unload compiled function (OS X)
dyn.unload("divideBy3.dll") #unload compiled function (Windows)
```