```
   print "in foo1 with a= ",var1
End

Function foo2(a)
   Variable a

   print "in foo2 with a= ",a
End

Function foo3(a)
   Variable a

   print "in foo3 with a= ",a
End

Function badfoo(a,b)
   Variable a,b

   print "in badfoo with a= ",a
End

Function bar(a,b,fin)
   Variable a,b
   FUNCREF myprotofunc fin

   if( a==1 )
      FUNCREF myprotofunc f= foo1
   elseif( a==2 )
      FUNCREF myprotofunc f= $"foo2"
   elseif( a==3 )
      FUNCREF myprotofunc f= fin
   endif
   f(b)
End
```

For the above functions, the following table shows the results for various invocations of the bar function executed on the command line:

Command	Result
`bar(1,33,foo3)`	`in foo1 with a= 33`
`bar(2,44,foo3)`	`in foo2 with a= 44`
`bar(3,55,foo3)`	`in foo3 with a= 55`
`bar(4,55,foo3)`	`in myprotofunc with a= 55`

Executing `bar(3,55,badfoo)` generates a syntax error dialog that highlights “badfoo” in the command. This error results because the format of the badfoo function does not match the format of the prototype function, myprotofunc.

Conditional Compilation

Compiler directives can be used to conditionally include or exclude blocks of code. This is especially useful when an XOP may or may not be available. It is also convenient for supporting different versions of Igor with the same code and for testing and debugging code.

For example, to enable a block of procedure code depending on the presence or absence of an XOP, use

```
#if Exists("nameOfAnXopRoutine")
   <code using XOP routines>
#endif
```