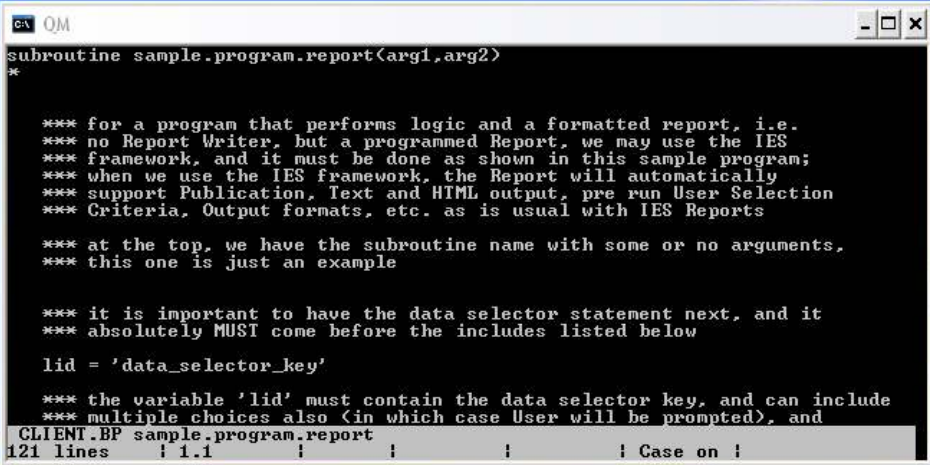



Slide 1 - Slide 1

```
QM
subroutine sample.program.report(arg1,arg2)
*
*** for a program that performs logic and a formatted report, i.e.
*** no Report Writer, but a programmed Report, we may use the IES
*** framework, and it must be done as shown in this sample program;
*** when we use the IES framework, the Report will automatically
*** support Publication, Text and HTML output, pre run User Selection
*** Criteria, Output formats, etc. as is usual with IES Reports
***
*** at the top, we have the subroutine name with some or no arguments,
*** this one is just an example
***
*** it is important to have the data selector statement next, and it
*** absolutely MUST come before the includes listed below
lid = 'data_selector_key'
*** the variable 'lid' must contain the data selector key, and can include
*** multiple choices also (in which case User will be prompted), and
CLIENT.BP sample.program.report
121 lines      | 1.1      |      |      |      | Case on |
```

Slide notes

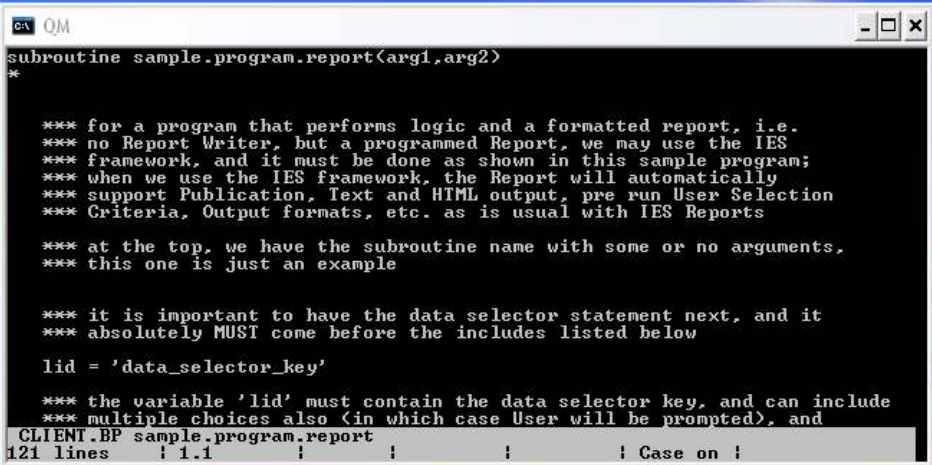
Despite all the IES Report Writers, there are occasions when Developers may prefer to write a specific Report by program. For example, all the Audit Trail programs in IES are programs, in other words, written by code logic rather than Report Writers. This can only be done when we have programming skills. When we choose to write a Report by program, then it is recommended that we use the IES framework that is provided for this purpose. In the file CLIENT.BP, we have a sample program that makes it quite clear how to do this. This Program is called "sample.program.report" and it may be copied to new names and then used as the basis of a new report program, by extending and adding to it.

Slide 2 - Slide 2

```
QM
subroutine sample.program.report(arg1,arg2)
*
*** for a program that performs logic and a formatted report, i.e.
*** no Report Writer, but a programmed Report, we may use the IES
*** framework, and it must be done as shown in this sample program;
*** when we use the IES framework, the Report will automatically
*** support Publication, Text and HTML output, pre run User Selection
*** Criteria, Output formats, etc. as is usual with IES Reports
***
*** at the top, we have the subroutine name with some or no arguments,
*** this one is just an example
***
*** it is important to have the data selector statement next, and it
*** absolutely MUST come before the includes listed below
lid = 'data_selector_key'
*** the variable 'lid' must contain the data selector key, and can include
*** multiple choices also (in which case User will be prompted), and
CLIENT.BP sample.program.report
121 lines      | 1.1      |      |      |      | Case on |
```

Slide notes

Our program will always start with a subroutine name and possibly some arguments. This sample program includes many comment lines to explain what we should do.

Slide 3 - Slide 3

```
QM
subroutine sample.program.report(arg1,arg2)
*
*** for a program that performs logic and a formatted report, i.e.
*** no Report Writer, but a programmed Report, we may use the IES
*** framework, and it must be done as shown in this sample program;
*** when we use the IES framework, the Report will automatically
*** support Publication, Text and HTML output, pre run User Selection
*** Criteria, Output formats, etc. as is usual with IES Reports
***
*** at the top, we have the subroutine name with some or no arguments,
*** this one is just an example
***
*** it is important to have the data selector statement next, and it
*** absolutely MUST come before the includes listed below
lid = 'data_selector_key'
*** the variable 'lid' must contain the data selector key, and can include
*** multiple choices also (in which case User will be prompted), and
CLIENT.BP sample.program.report
121 lines | 1.1 | | | | Case on |
```

Slide notes

The data selector statement is always at the top, before the standard includes used in the framework.

Slide 4 - Slide 4

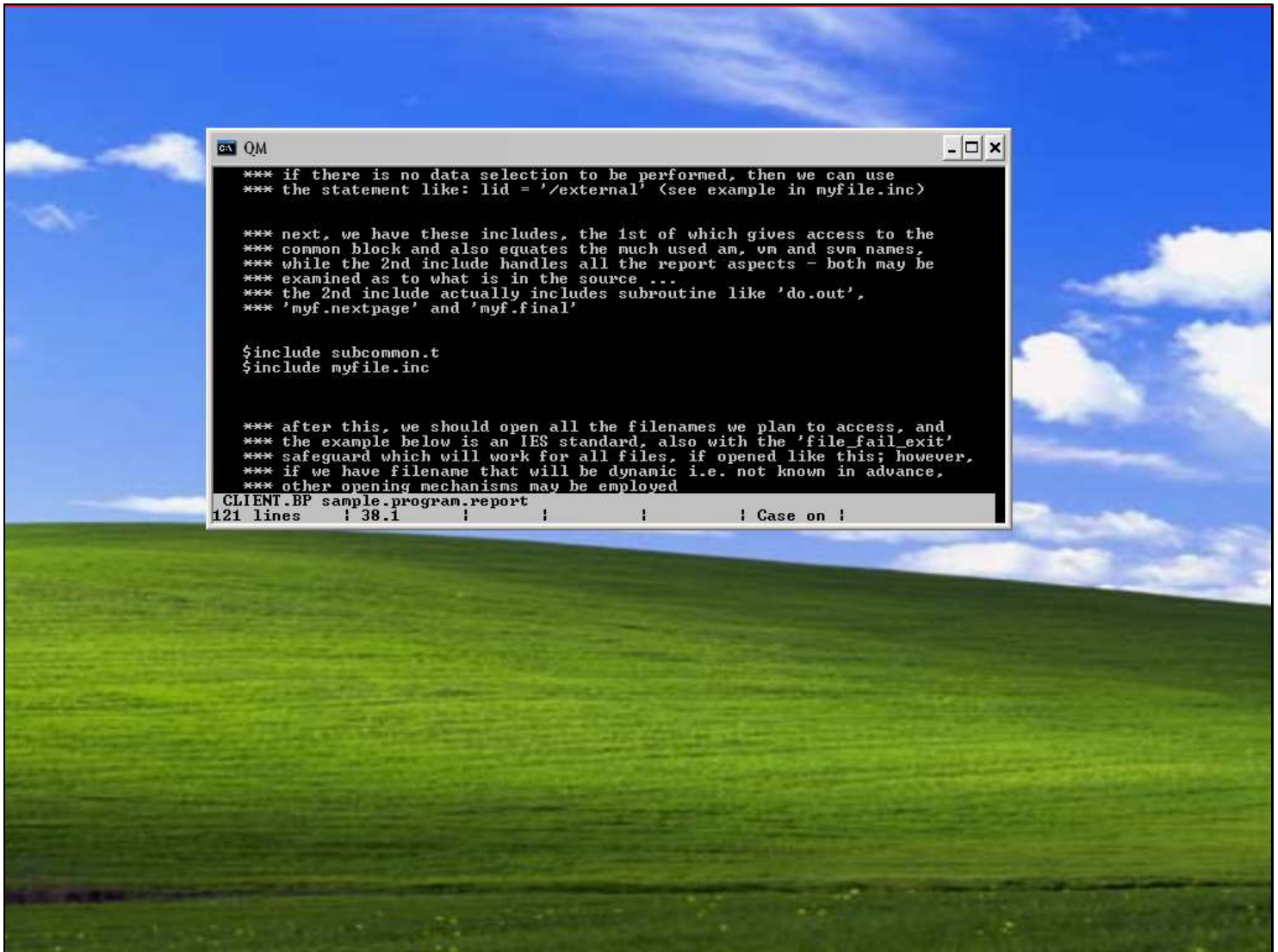


```
QM
subroutine sample.program.report(arg1,arg2)
*
*** for a program that performs logic and a formatted report, i.e.
*** no Report Writer, but a programmed Report, we may use the IES
*** framework, and it must be done as shown in this sample program;
*** when we use the IES framework, the Report will automatically
*** support Publication, Text and HTML output, pre run User Selection
*** Criteria, Output formats, etc. as is usual with IES Reports
***
*** at the top, we have the subroutine name with some or no arguments,
*** this one is just an example
***
*** it is important to have the data selector statement next, and it
*** absolutely MUST come before the includes listed below
- lid = 'data_selector_key'
*** the variable 'lid' must contain the data selector key, and can include
*** multiple choices also (in which case User will be prompted), and
CLIENT.BP sample.program.report
121 lines | 19.1 | | | | Case on |
```

Slide notes

The standard includes take care of the required subroutines for print lines, new pages and closing procedures for the report.

Slide 5 - Slide 5



```
*** if there is no data selection to be performed, then we can use
*** the statement like: lid = '/external' (see example in myfile.inc)

*** next, we have these includes, the 1st of which gives access to the
*** common block and also equates the much used an, vm and sum names,
*** while the 2nd include handles all the report aspects - both may be
*** examined as to what is in the source ...
*** the 2nd include actually includes subroutine like 'do.out',
*** 'myf.nextpage' and 'myf.final'

$include subcommon.t
$include myfile.inc

*** after this, we should open all the filenames we plan to access, and
*** the example below is an IES standard, also with the 'file fail_exit'
*** safeguard which will work for all files, if opened like this; however,
*** if we have filename that will be dynamic i.e. not known in advance,
*** other opening mechanisms may be employed
CLIENT.BP sample.program.report
121 lines      ! 38.1      !      !      !      ! Case on !
```

Slide notes

After this, we usually open the data files that we will access, and when done according to the IES standard as shown here then we have the fail-safe procedure included.

Slide 6 - Slide 6



```
QM
$include subcommon.t
$include myfile.inc

*** after this, we should open all the filenames we plan to access, and
*** the example below is an IES standard, also with the 'file_fail_exit'
*** safeguard which will work for all files, if opened like this; however,
*** if we have filename that will be dynamic i.e. not known in advance,
*** other opening mechanisms may be employed

open '','TRANSREAL' to fbxtransreal else
  vfbx = status() : ":" : "transreal"
  call file_fail_exit(vfbx)
end

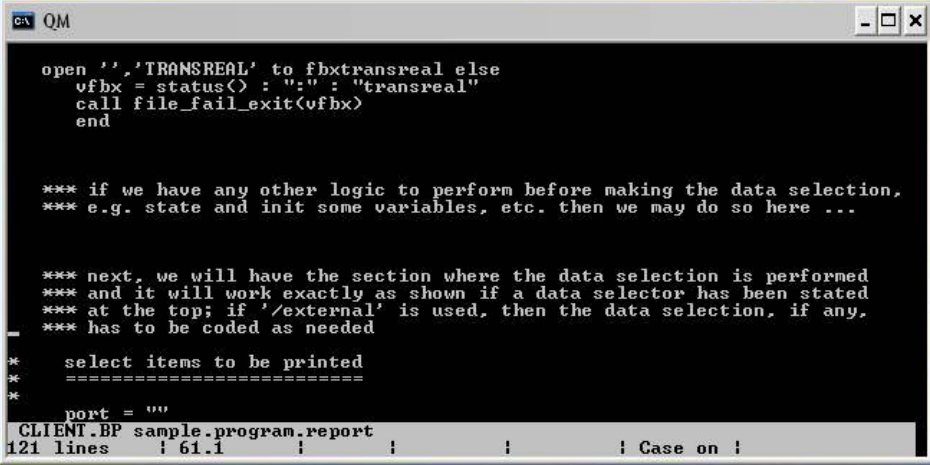
*** if we have any other logic to perform before making the data selection,
*** e.g. state and init some variables, etc. then we may do so here ...

CLIENT.BP sample.program.report
121 lines | 51.1 | | | | Case on |
```

Slide notes

After that, we have program code to retrieve the data selection which would have already been performed and saved by the system.

Slide 7 - Slide 7



```
open '','TRANSREAL' to fbxtransreal else
vfbx = status() : ":" : "transreal"
call file_fail_exit(vfbx)
end

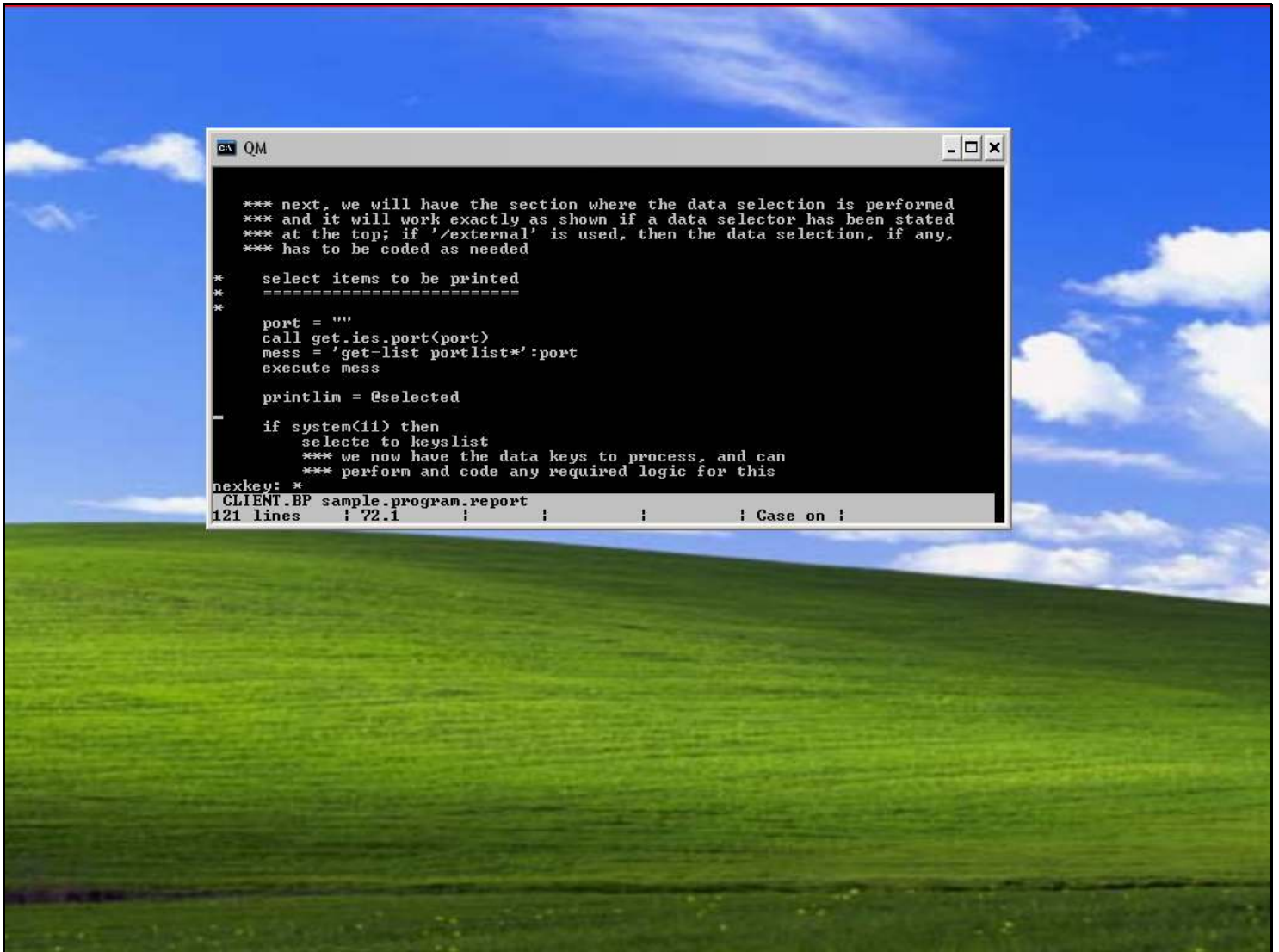
*** if we have any other logic to perform before making the data selection,
*** e.g. state and init some variables, etc. then we may do so here ...

*** next, we will have the section where the data selection is performed
*** and it will work exactly as shown if a data selector has been stated
*** at the top; if '/external' is used, then the data selection, if any,
*** has to be coded as needed

*   select items to be printed
*   =====
*
port = ""
CLIENT.BP sample.program.report
121 lines      | 61.1      |      |      |      | Case on |
```

Slide notes

This is how it's done in the framework context.

Slide 8 - Slide 8

```
*** next, we will have the section where the data selection is performed
*** and it will work exactly as shown if a data selector has been stated
*** at the top; if '/external' is used, then the data selection, if any,
*** has to be coded as needed

*   select items to be printed
*   =====
*
port = ""
call get.ies.port(port)
mess = 'get-list portlist*':port
execute mess

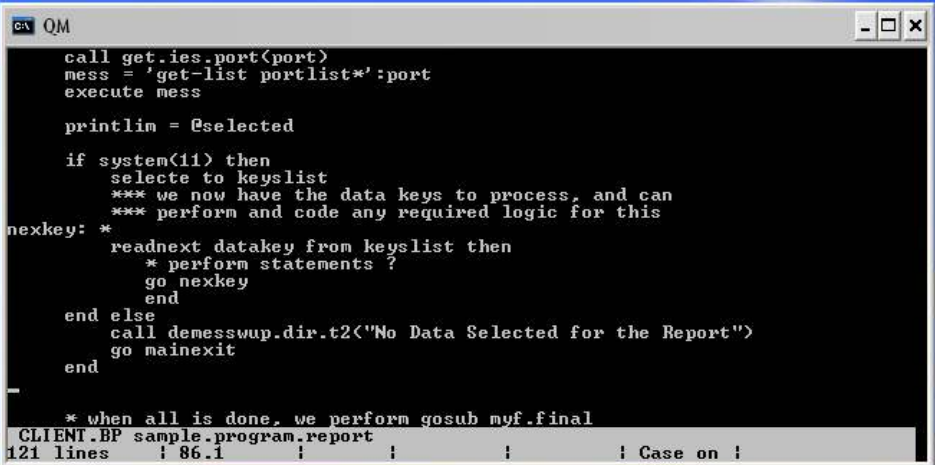
printlim = @selected

if system(11) then
  selecte to keylist
  *** we now have the data keys to process, and can
  *** perform and code any required logic for this
nextkey: *
CLIENT.BP sample.program.report
121 lines   | 72.1   |   |   |   | Case on |
```

Slide notes

And after this we have an execution loop where we perform the Data Record iterations. Of course, the program can also work differently if it is not based on a Data Record iteration, but it should still be clear how we will go about the logic. It is during this stage that we are very likely to perform subroutines that may be placed lower down in the subroutine section.

Slide 9 - Slide 9



```
call get.ies.port(port)
mess = 'get-list portlist*':port
execute mess

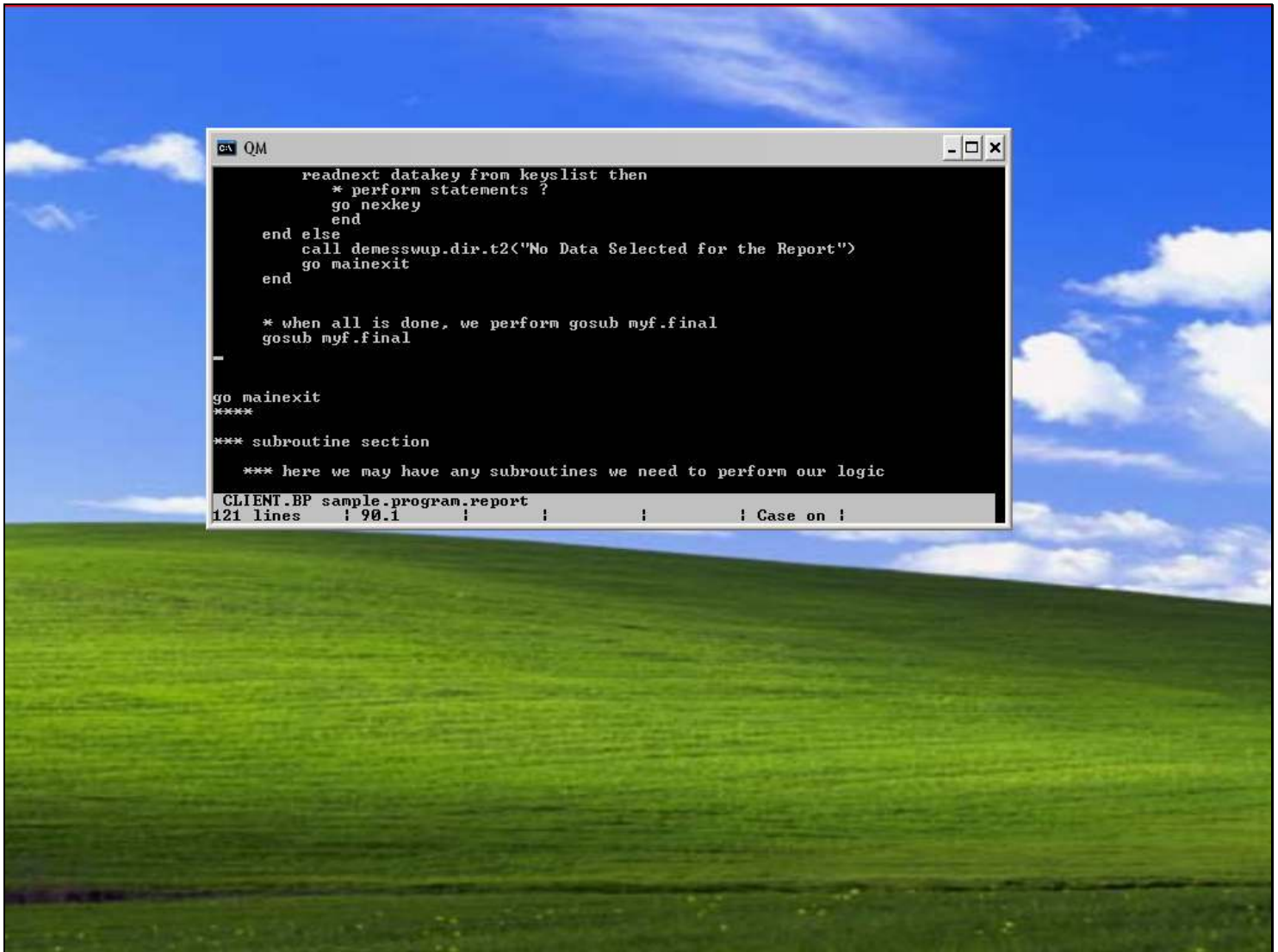
printlim = @selected

if system(11) then
  selecte to keylist
  *** we now have the data keys to process, and can
  *** perform and code any required logic for this
nexuskey: *
  readnext datakey from keylist then
    * perform statements ?
    go nexuskey
  end
end else
  call demesswup.dir.t2("No Data Selected for the Report")
  go mainexit
end

* when all is done, we perform gosub myf.final
CLIENT.BP sample.program.report
121 lines      ! 86.1      !      !      ! Case on !
```

Slide notes

When all is said and done, we perform "myf.final" to wrap up the results and present the Report Output to the User.

Slide 10 - Slide 10

```
readnext datakey from keylist then
  * perform statements ?
  go nexkey
end
end else
  call demesswup.dir.t2<"No Data Selected for the Report">
  go mainexit
end

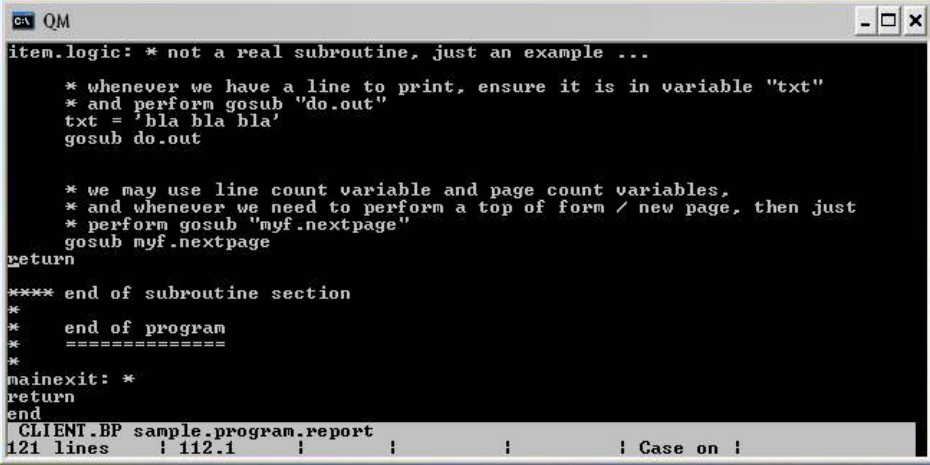
* when all is done, we perform gosub myf.final
gosub myf.final

-

go mainexit
****
*** subroutine section
*** here we may have any subroutines we need to perform our logic
CLIENT.BP sample.program.report
121 lines      ! 90.1      !      !      !      ! Case on !
```

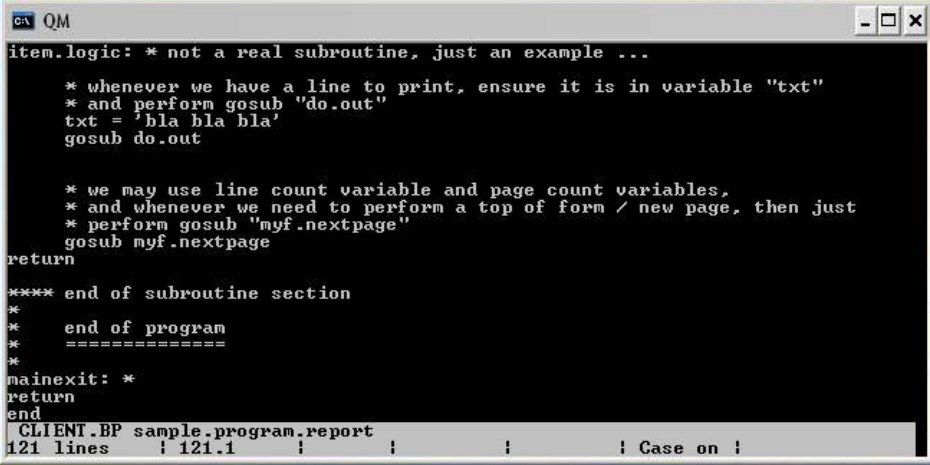
Slide notes

And here we have the sample subroutine section, with examples of how we will deal with print lines and new pages.

Slide 11 - Slide 11

```
item.logic: * not a real subroutine, just an example ...
* whenever we have a line to print, ensure it is in variable "txt"
* and perform gosub "do.out"
txt = 'bla bla bla'
gosub do.out
* we may use line count variable and page count variables,
* and whenever we need to perform a top of form / new page, then just
* perform gosub "myf.nextpage"
gosub myf.nextpage
return
**** end of subroutine section
*
* end of program
* =====
*
mainexit: *
return
end
CLIENT.BP sample.program.report
121 lines      | 112.1      |      |      |      | Case on |
```

Slide notes

Slide 12 - Slide 12

```
item.logic: * not a real subroutine, just an example ...
* whenever we have a line to print, ensure it is in variable "txt"
* and perform gosub "do.out"
txt = 'bla bla bla'
gosub do.out
* we may use line count variable and page count variables,
* and whenever we need to perform a top of form / new page, then just
* perform gosub "myf.nextpage"
gosub myf.nextpage
return
**** end of subroutine section
*
* end of program
* =====
*
mainexit: *
return
end
CLIENT.BP sample.program.report
121 lines      | 121.1      |      |      |      | Case on |
```

Slide notes

And there we have the end of the program. So by using this basis, we can use the IES framework productively, and add to and pad the program with our own required logic.