


Next

Looping and Conditional Execution

Part 3 - Conditional Execution

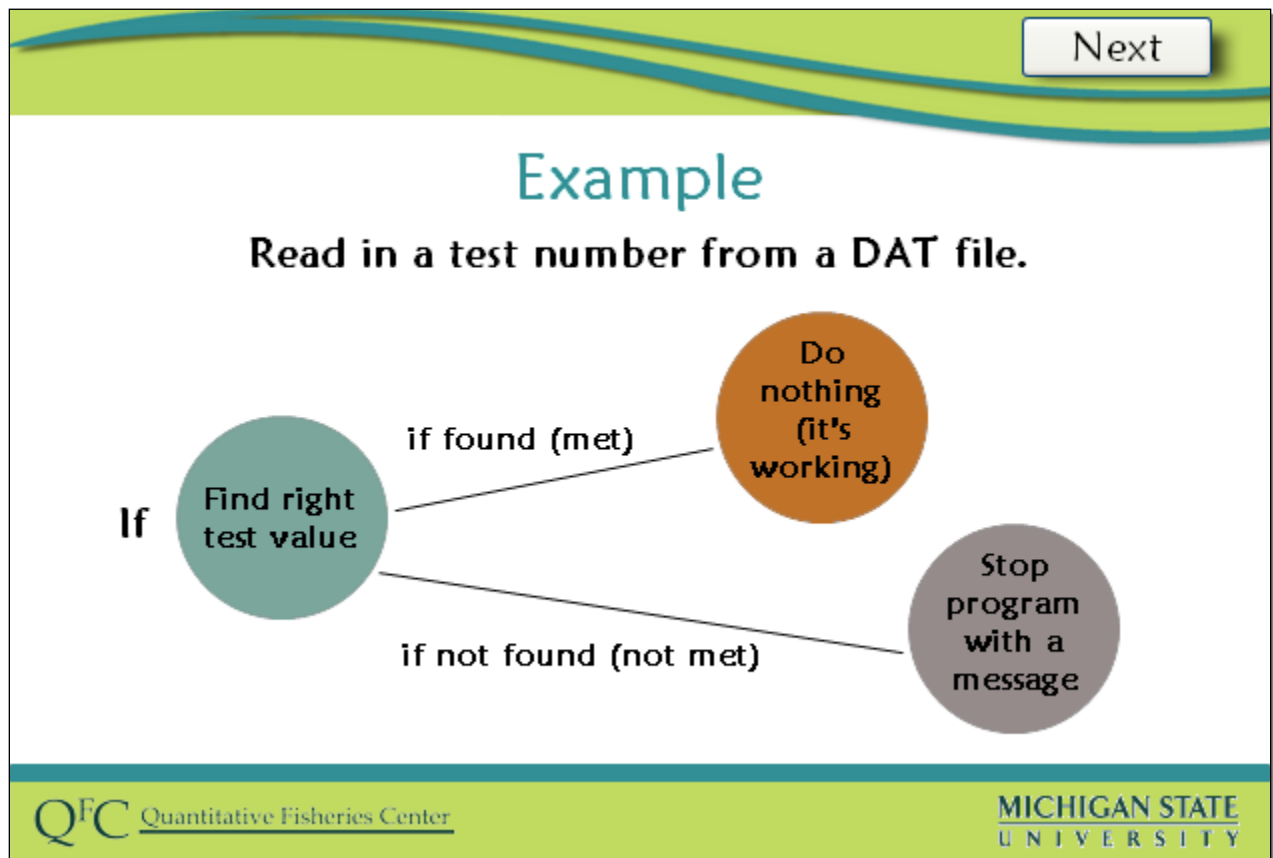


This video was created using ADMB-IDE release 4.5.0-1 (July 15, 2011)
You may notice some minor differences if using a different version.

Q^{FC} Quantitative Fisheries Center

MICHIGAN STATE
UNIVERSITY

We now turn to the use of conditional statements. Particularly the if statement. Our intent is to control what happens based on a condition that is tested. First determine if you would like a little background on the if statement before we proceed to the coding.



In our first example we will read in a test number from a DAT file and our program will stop with an appropriate message if the right test value is not found. We have advocated checking on a test value as a standard part of building a tpl file. However, even after an initial version is working there is a possibility that you make changes and forget to check that the data still read in correctly before proceeding. Our conditional approach will automate the check for us and not stop the program if things seem ok.

Next

Basic Syntax

Simple

Logical test

if (condition) Statement

What to execute if the statement is true

Compound

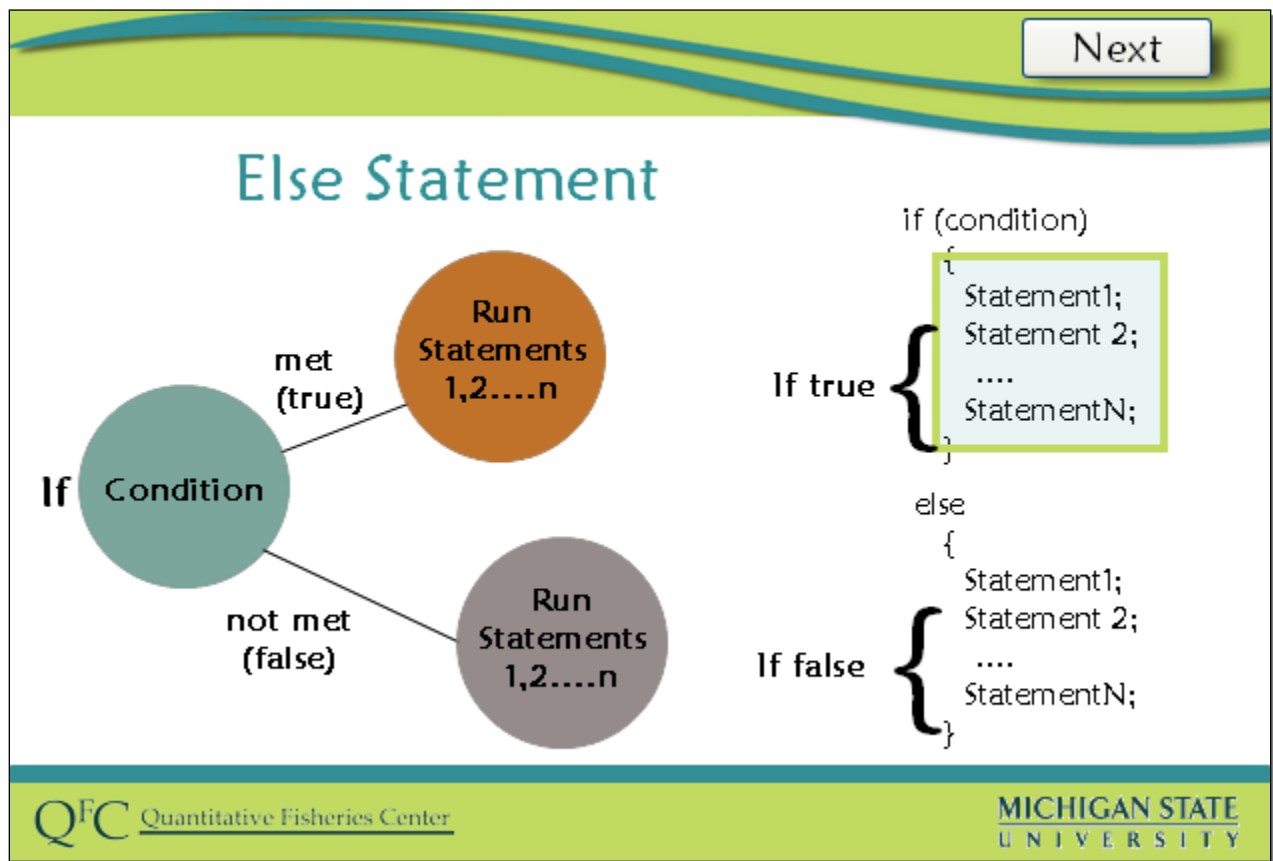
```
if (condition)
{
  Statement1;
  Statement 2;
  ....
  StatementN;
}
```

If true, do all of these

QFC Quantitative Fisheries Center

MICHIGAN STATE UNIVERSITY

The basic syntax for a simple if statement is shown here. Here condition is a logical test and statement is the statement to be executed if the statement condition is true. Statement can be a compound statement so the if statement could be displayed in this form where there could be multiple statements that get executed when the condition is true.



Optionally you can also specify using an else statement what should happen only if the condition is not true. For example, these statements will run if the condition is true, and the statements after the else will run if the statement is false. The else statement always comes immediately after an if statement.

Next

Prepare Data Files

<u>Option 1</u>	<u>Option 2</u>
<ol style="list-style-type: none">1. Open multiLinf.tpl that you worked with in the previous looping videos.	<ol style="list-style-type: none">1. Create a new folder.2. Save multiLinf.dat to the folder.3. Save multiLinf_Loop3_Start.tpl to the folder.4. Change the name of the tpl to multiLinf.tpl

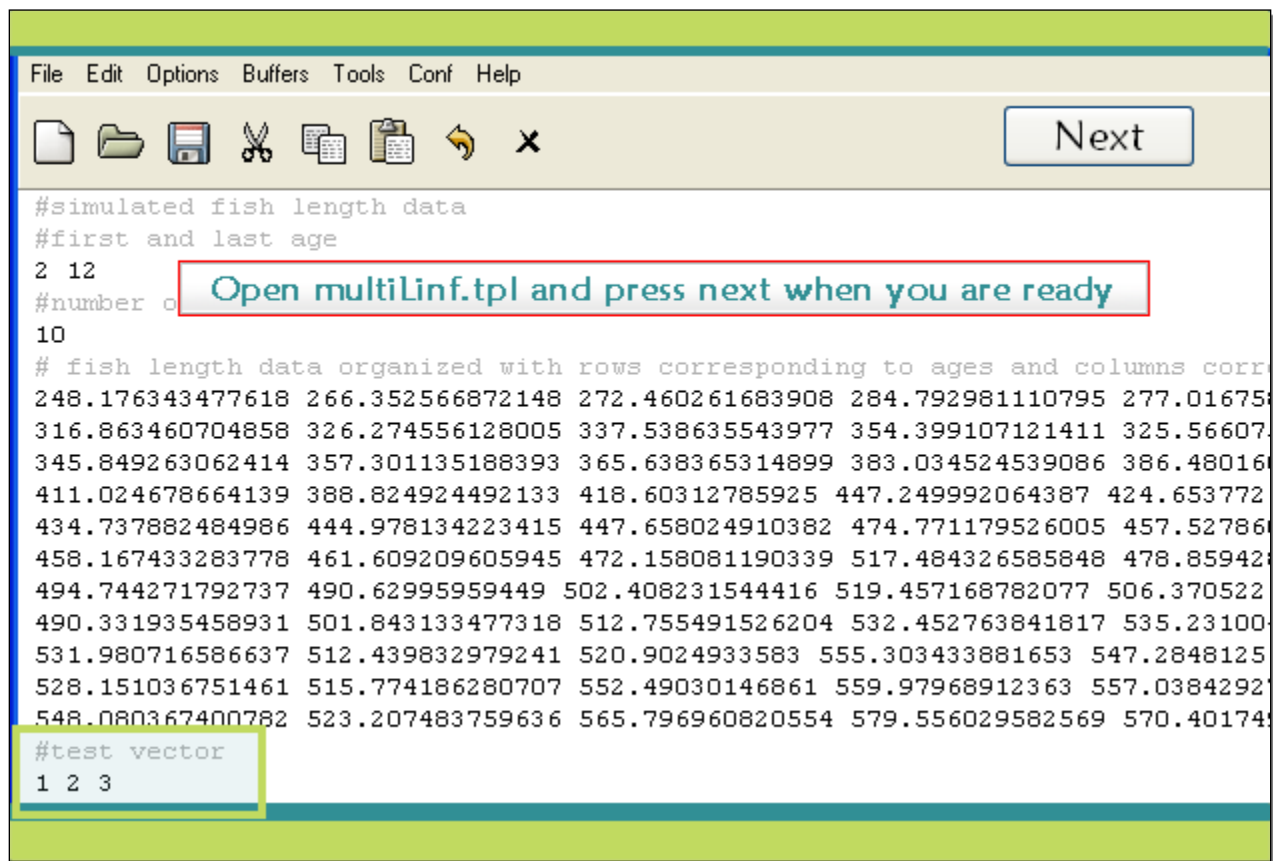
QFC Quantitative Fisheries Center

Prepare and open your files.

Either open multiLinf.tpl that you worked with in the previous looping videos or:

1. Create a new folder.
2. Save multiLinf.dat to the folder.
3. Save multiLinf_Loop3_Start.tpl to the folder.
4. Change the name of the tpl to multiLinf.tpl

Click next when you are ready.

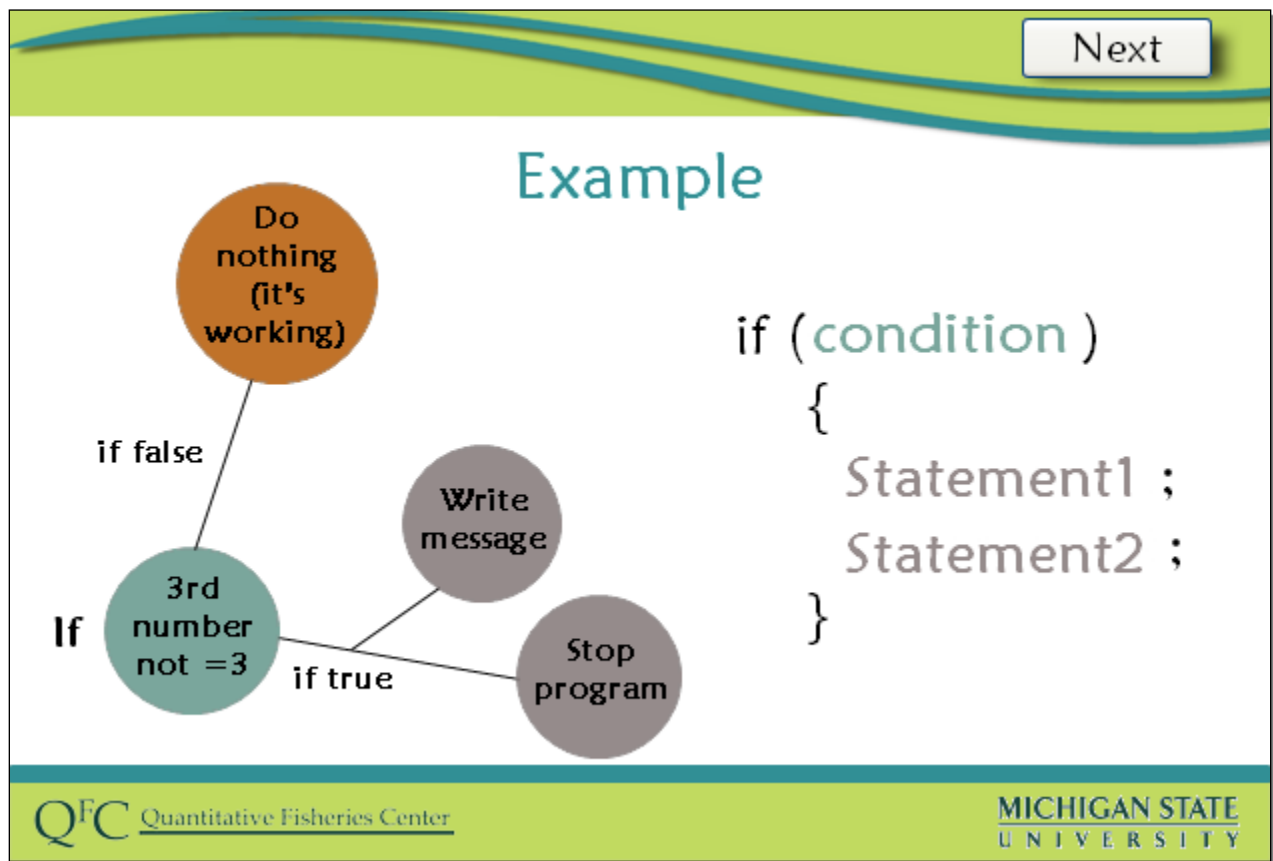


File Edit Options Buffers Tools Conf Help

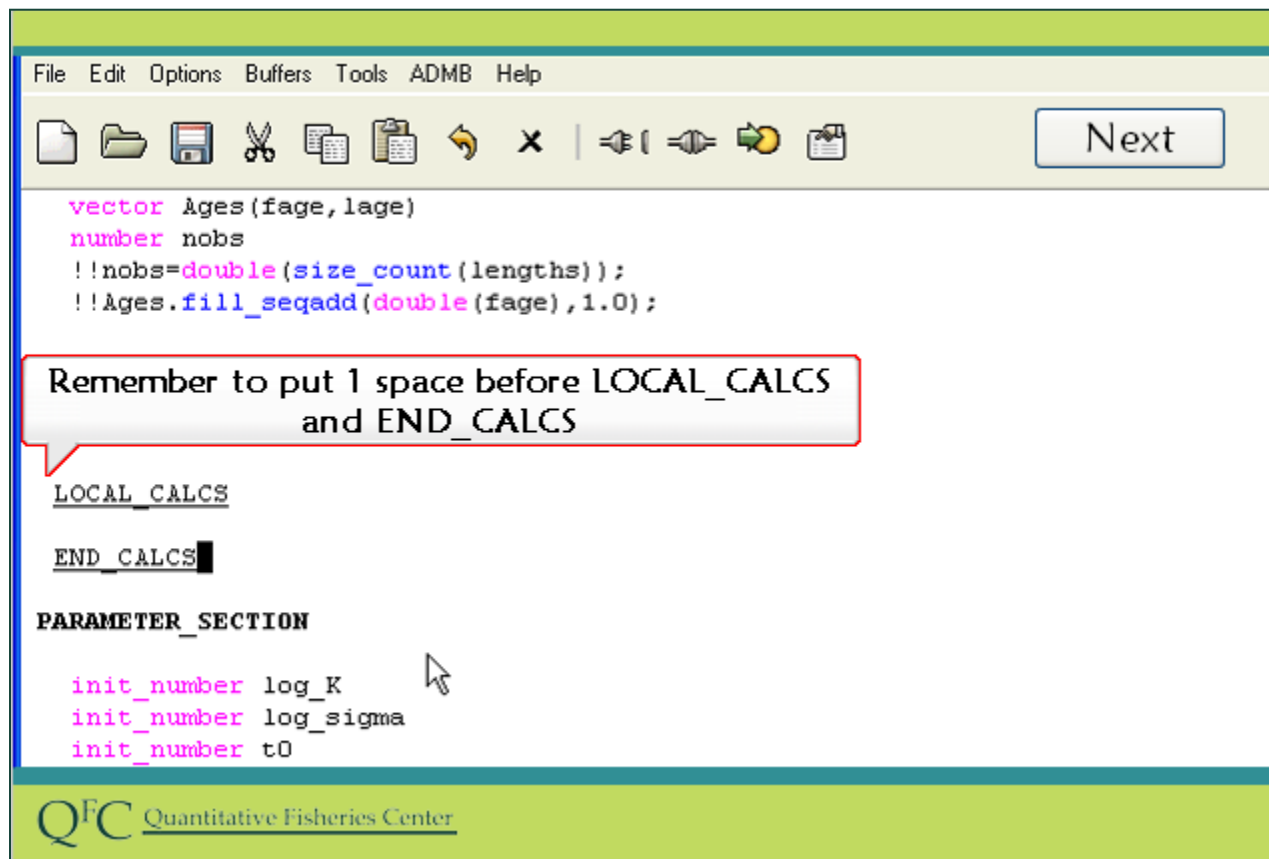
Next

```
#simulated fish length data
#first and last age
2 12
#number of fish length data organized with rows corresponding to ages and columns corresponding to
10
# fish length data organized with rows corresponding to ages and columns corresponding to
248.176343477618 266.352566872148 272.460261683908 284.792981110795 277.016751
316.863460704858 326.274556128005 337.538635543977 354.399107121411 325.566071
345.849263062414 357.301135188393 365.638365314899 383.034524539086 386.480161
411.024678664139 388.824924492133 418.60312785925 447.249992064387 424.653772
434.737882484986 444.978134223415 447.658024910382 474.771179526005 457.527861
458.167433283778 461.609209605945 472.158081190339 517.484326585848 478.859421
494.744271792737 490.62995959449 502.408231544416 519.457168782077 506.370522
490.331935458931 501.843133477318 512.755491526204 532.452763841817 535.231001
531.980716586637 512.439832979241 520.9024933583 555.303433881653 547.2848125
528.151036751461 515.774186280707 552.49030146861 559.97968912363 557.03842921
548.080367400782 523.207483759636 565.796960820554 579.556029582569 570.401741
#test vector
1 2 3
```

Now let's try our first if statement. Recall that the dat file for our multiple pond length at age example contains a test vector with three values. Open multiLinef.tpl and press next when ready.



For our first if statement we will check if the third value is not equal to three, and if it isn't we will cause the program to exit. We are going to both write out an error statement and stop the program so our statement will be a compound one.

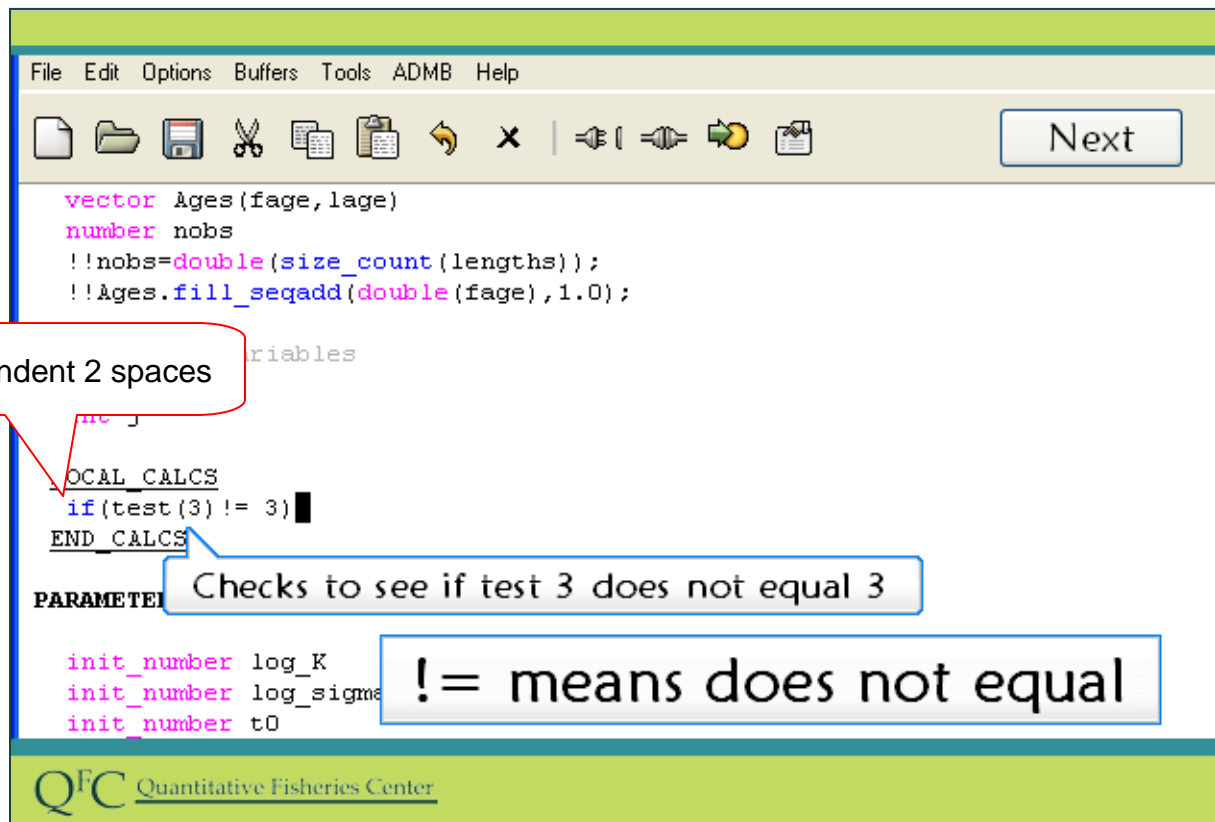


We will be adding the if statement to the bottom of the data section so it will need to go within local calcs blocks since this is C++ code.

Slide Code:

LOCAL_CALCS

END_CALCS



The screenshot shows a software window with a menu bar (File, Edit, Options, Buffers, Tools, ADBB, Help) and a toolbar. The code editor contains the following text:

```
vector Ages(fage, lage)
number nobs
!!nobs=double(size_count(lengths));
!!Ages.fill_seqadd(double(fage), 1.0);

LOCAL_CALCS
  if(test(3) != 3)
END_CALCS

PARAMETER
  init_number log_K
  init_number log_sigma
  init_number t0
```

Annotations include:

- A red callout bubble pointing to the indentation of the `if` statement: "Indent 2 spaces".
- A blue callout bubble pointing to the `if` statement: "Checks to see if test 3 does not equal 3".
- A large blue callout bubble explaining the operator: "`!=` means does not equal".
- A "Next" button in the top right corner.
- A footer with the "QFC Quantitative Fisheries Center" logo.

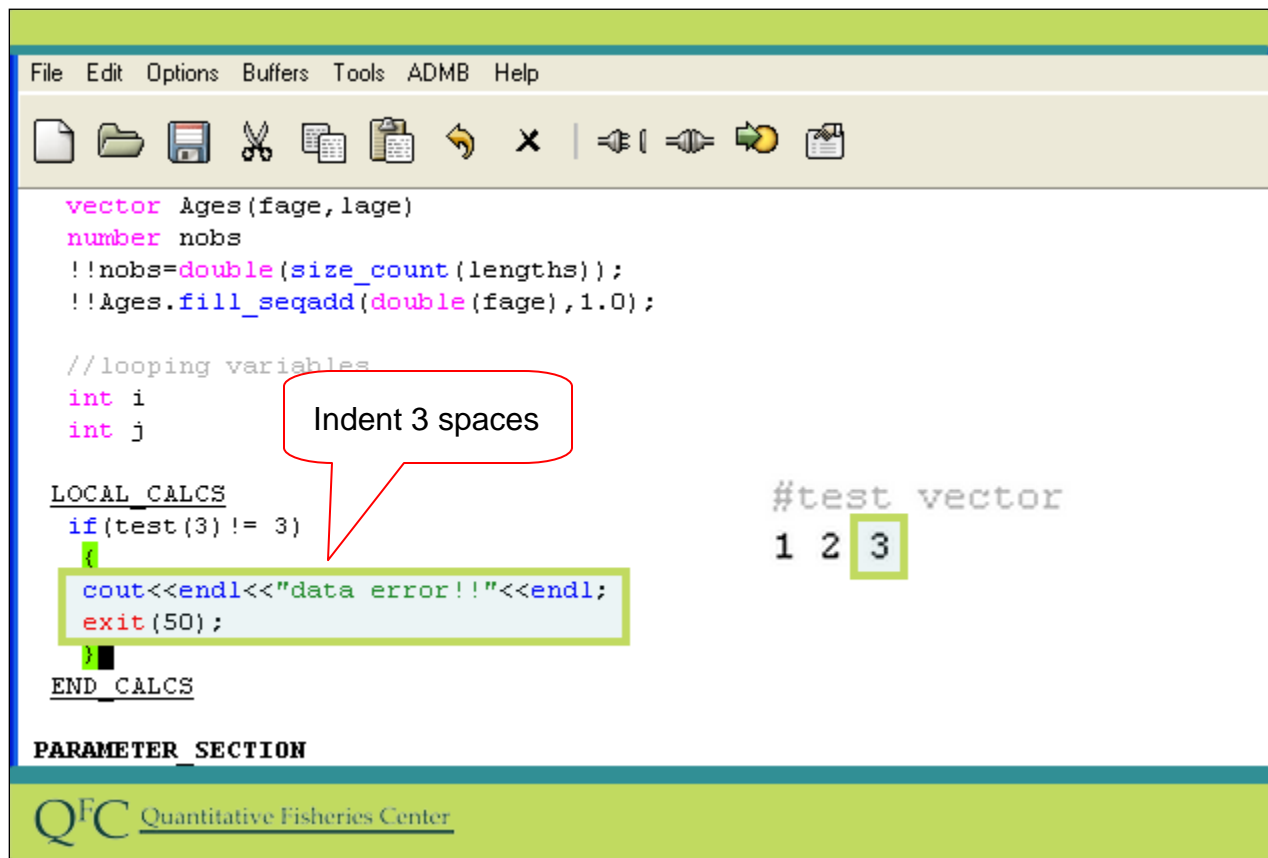
Following the word `if` we put our condition within parentheses. Here exclamation followed by an equals sign means not equal to. This should return true if test three does not equal 3 or false if it equals 3.

Slide Code:

LOCAL_CALCS

if (test (3) != 3)

END_CALCS



```
File Edit Options Buffers Tools ADMB Help

vector Ages(fage, lage)
number nobs
!!nobs=double(size_count(lengths));
!!Ages.fill_seqadd(double(fage), 1.0);

//looping variables
int i
int j

LOCAL_CALCS
if(test(3) != 3)
{
    cout<<endl<<"data error!!"<<endl;
    exit(50);
}
END_CALCS

PARAMETER_SECTION
```

Indent 3 spaces

#test vector
1 2 3

QFC Quantitative Fisheries Center

Within the curly braces we put two lines. The cout statement will just write out that there has been a data error.

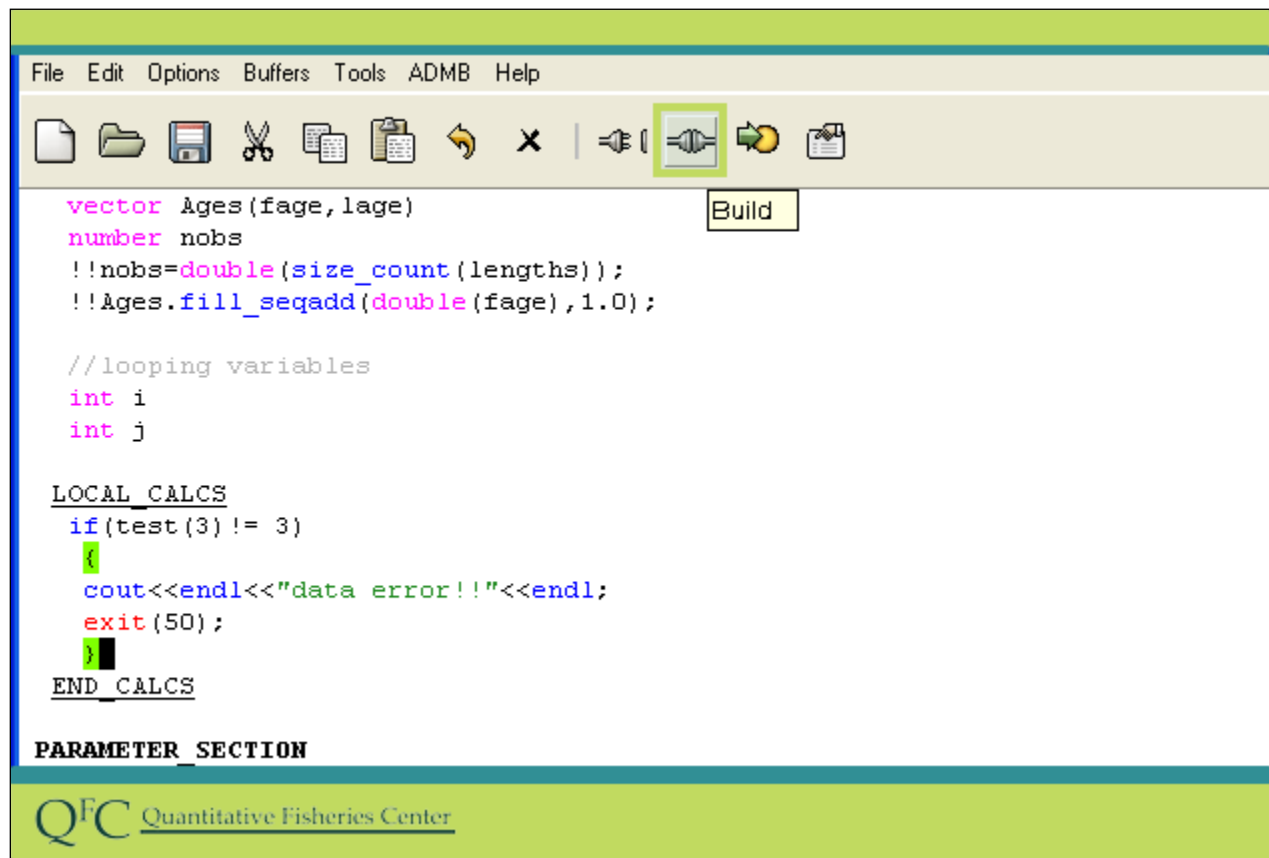
The exit statement will cause the program to quit at that point.

We close the loop.

Remember, these will only happen if our test 3 does not equal 3, which as you remember it does equal 3.

Slide Code:

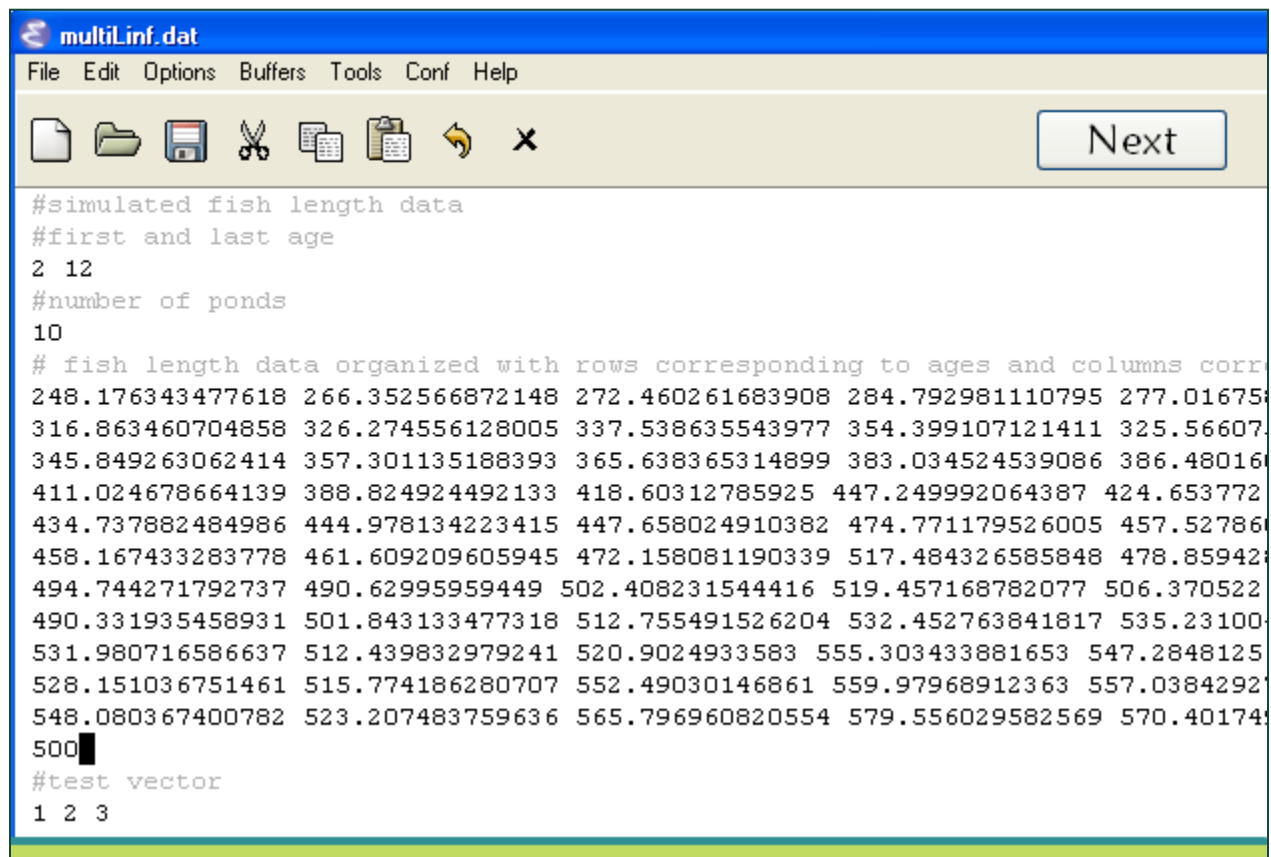
```
{
    cout<<endl<<"data error!!"<<endl;
    exit(50);
}
```



Now let's build and run our program. It still runs fine and we see no change because in fact test(3) does equal 3.

Slide Action:

Build and run the program



The screenshot shows a text editor window titled "multiLinf.dat". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Conf", and "Help". The toolbar contains icons for file operations (new, open, save, print, copy, paste, undo, redo) and a "Next" button. The text content is as follows:

```
#simulated fish length data
#first and last age
2 12
#number of ponds
10
# fish length data organized with rows corresponding to ages and columns corresponding to ponds
248.176343477618 266.352566872148 272.460261683908 284.792981110795 277.016754
316.863460704858 326.274556128005 337.538635543977 354.399107121411 325.56607
345.849263062414 357.301135188393 365.638365314899 383.034524539086 386.48016
411.024678664139 388.824924492133 418.60312785925 447.249992064387 424.653772
434.737882484986 444.978134223415 447.658024910382 474.771179526005 457.52786
458.167433283778 461.609209605945 472.158081190339 517.484326585848 478.85942
494.744271792737 490.62995959449 502.408231544416 519.457168782077 506.370522
490.331935458931 501.843133477318 512.755491526204 532.452763841817 535.23100
531.980716586637 512.439832979241 520.9024933583 555.303433881653 547.2848125
528.151036751461 515.774186280707 552.49030146861 559.97968912363 557.0384292
548.080367400782 523.207483759636 565.796960820554 579.556029582569 570.40174
500
#test vector
1 2 3
```

Now let's open up the dat file and add an extra length before proceeding to the test vector. Here we add 500.

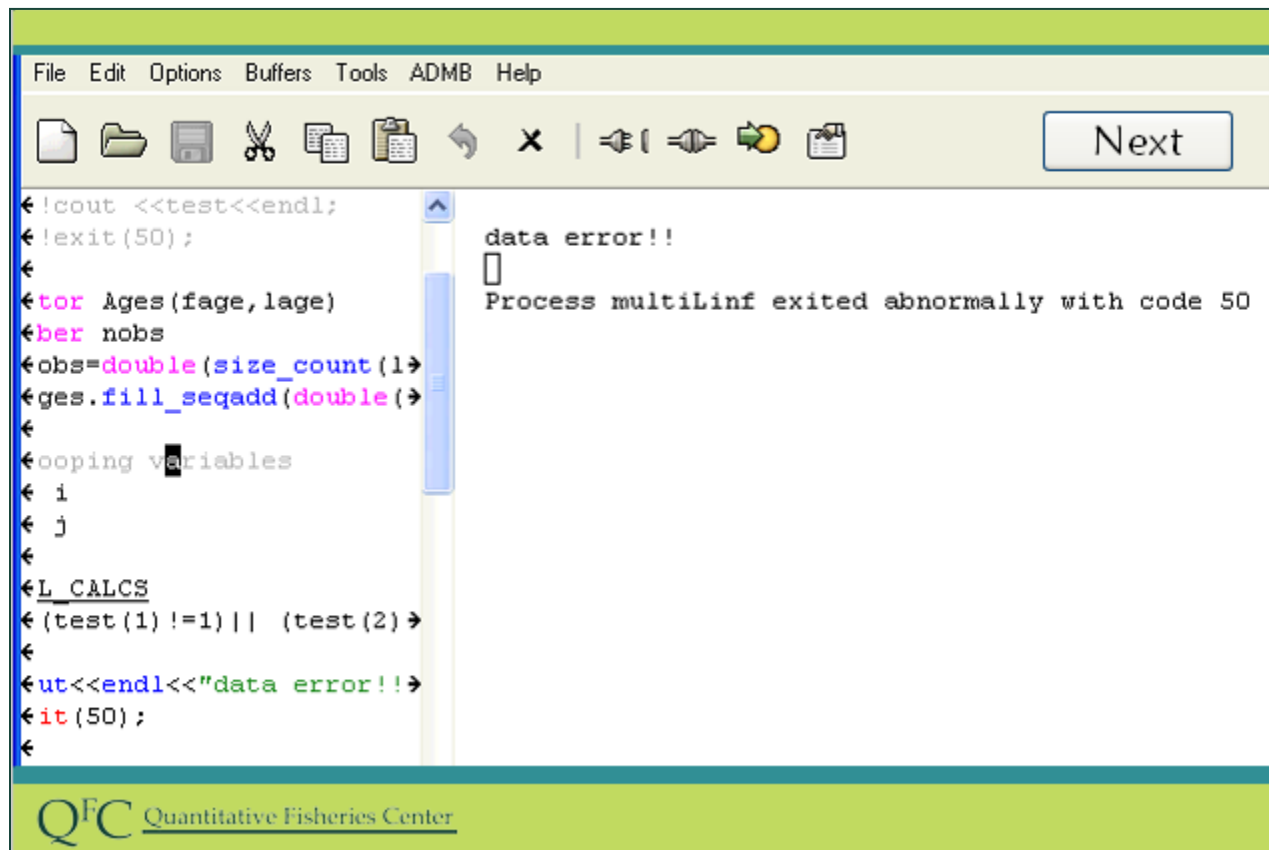
Save the changes to our dat file.

Slide Action:

Open multiLinf.dat

Type 500 in as an additional fish length.

Save changes



The screenshot shows a C++ IDE window with a menu bar (File, Edit, Options, Buffers, Tools, ADBM, Help) and a toolbar. The main editor area contains C++ code. The code defines a function `test` that takes two arguments, `fage` and `lage`, and returns a `bool`. The function body includes a loop that iterates over a range of values, calculating the square of each value and adding it to a running total. The function returns `true` if the total is greater than 100, and `false` otherwise. The `main` function calls `test` with arguments `1` and `2`, and prints the result. The output window shows the message "data error!!" and "Process multiLinf exited abnormally with code 50".

```
!cout <<test<<endl;
!exit(50);
tor Ages(fage, lage)
ber nob
obs=double(size_count(1)
ges.fill_seqadd(double(
oping variables
i
j
L_CALCS
(test(1)!=1) || (test(2)
ut<<endl<<"data error!!"
it(50);
```

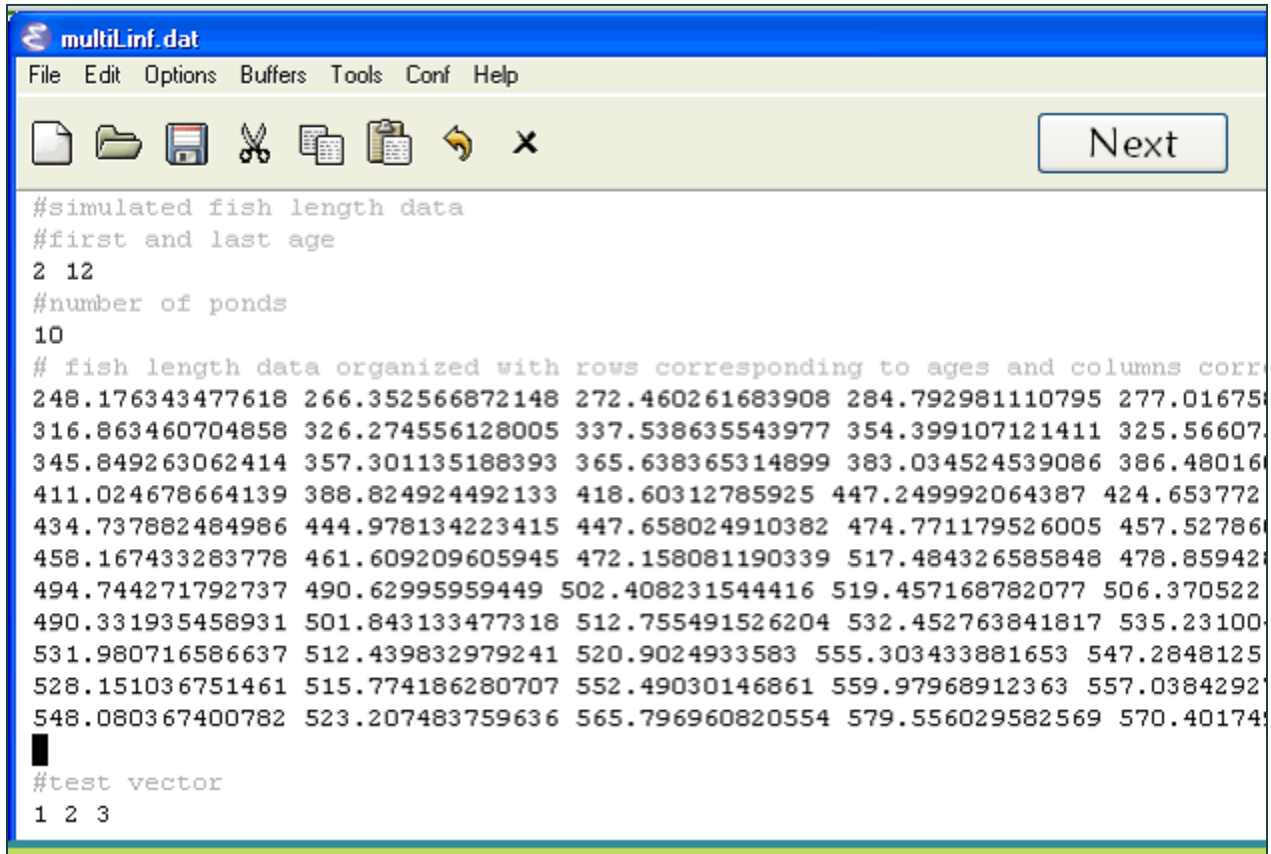
data error!!
Process multiLinf exited abnormally with code 50

QFC Quantitative Fisheries Center

Now if we run our program again it prints the error message "data error" that we specified in our cout statement and stops abnormally with exit code 50. It worked.

Slide Action:

Run program

A screenshot of a software window titled "multiInf.dat". The window has a menu bar with "File", "Edit", "Options", "Buffers", "Tools", "Conf", and "Help". Below the menu bar is a toolbar with icons for file operations (new, open, save, cut, copy, paste, undo, redo) and a "Next" button. The main text area contains the following text:

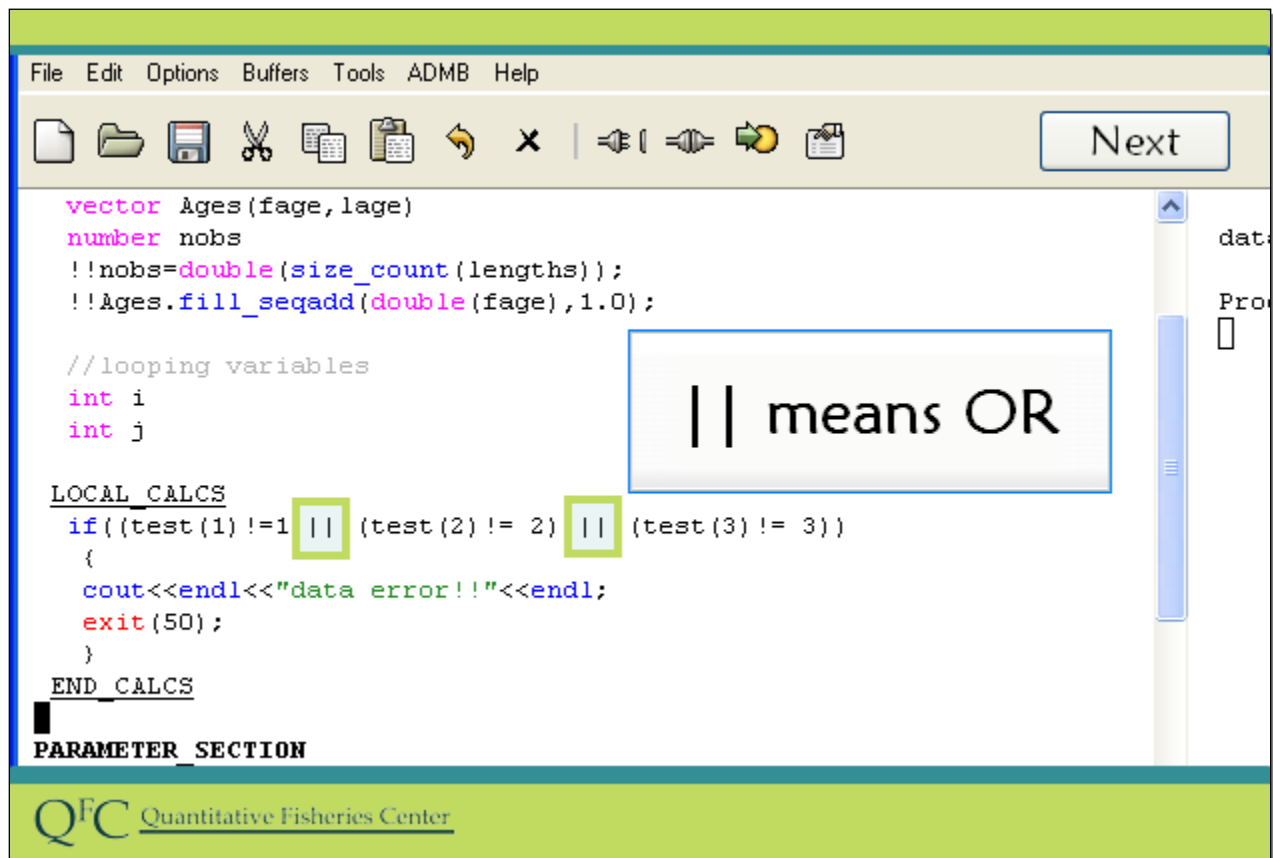
```
#simulated fish length data
#first and last age
2 12
#number of ponds
10
# fish length data organized with rows corresponding to ages and columns corresponding to ponds
248.176343477618 266.352566872148 272.460261683908 284.792981110795 277.016751
316.863460704858 326.274556128005 337.538635543977 354.399107121411 325.566071
345.849263062414 357.301135188393 365.638365314899 383.034524539086 386.480161
411.024678664139 388.824924492133 418.60312785925 447.249992064387 424.653772
434.737882484986 444.978134223415 447.658024910382 474.771179526005 457.527861
458.167433283778 461.609209605945 472.158081190339 517.484326585848 478.859421
494.744271792737 490.62995959449 502.408231544416 519.457168782077 506.370522
490.331935458931 501.843133477318 512.755491526204 532.452763841817 535.231001
531.980716586637 512.439832979241 520.9024933583 555.303433881653 547.28481251
528.151036751461 515.774186280707 552.49030146861 559.97968912363 557.03842921
548.080367400782 523.207483759636 565.796960820554 579.556029582569 570.401741
#
#test vector
1 2 3
```

Before we forget let's go back and delete the extra observation and re-save the dat file.

Slide Action:

Remove the new length (500) from the dat file

Re-save the dat file



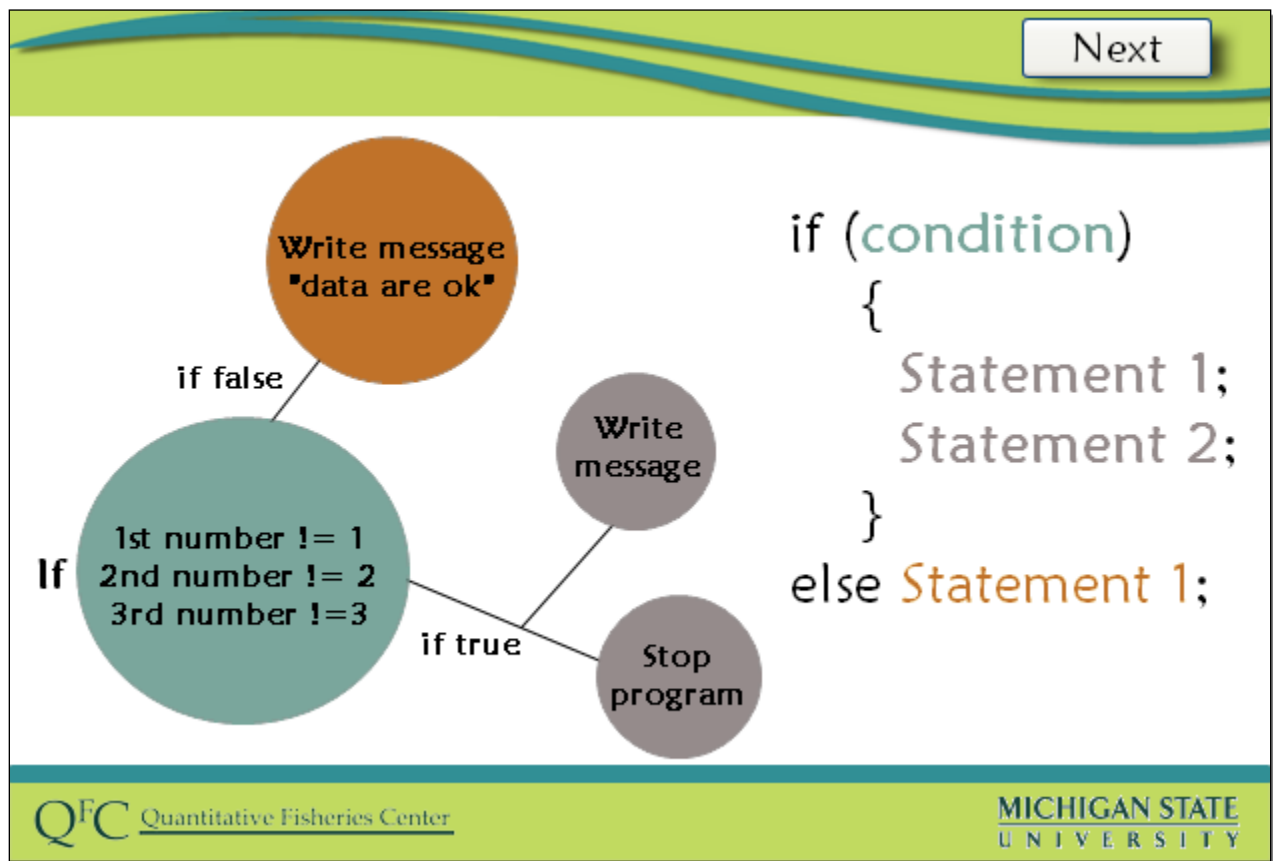
You may have noticed that our if statement is not a really rigorous test because we might get the number 3 as the value by accident if there happens to be two threes in a row and we got the wrong one. Ideally we would want to check that test one equals one, test two equals two and test three equals three and quit if any of them do not.

We can do this because our logical test can be a compound logical evaluation checking if test 1 does not equal one, test 2 does not equal 2 along with the test 3 check.

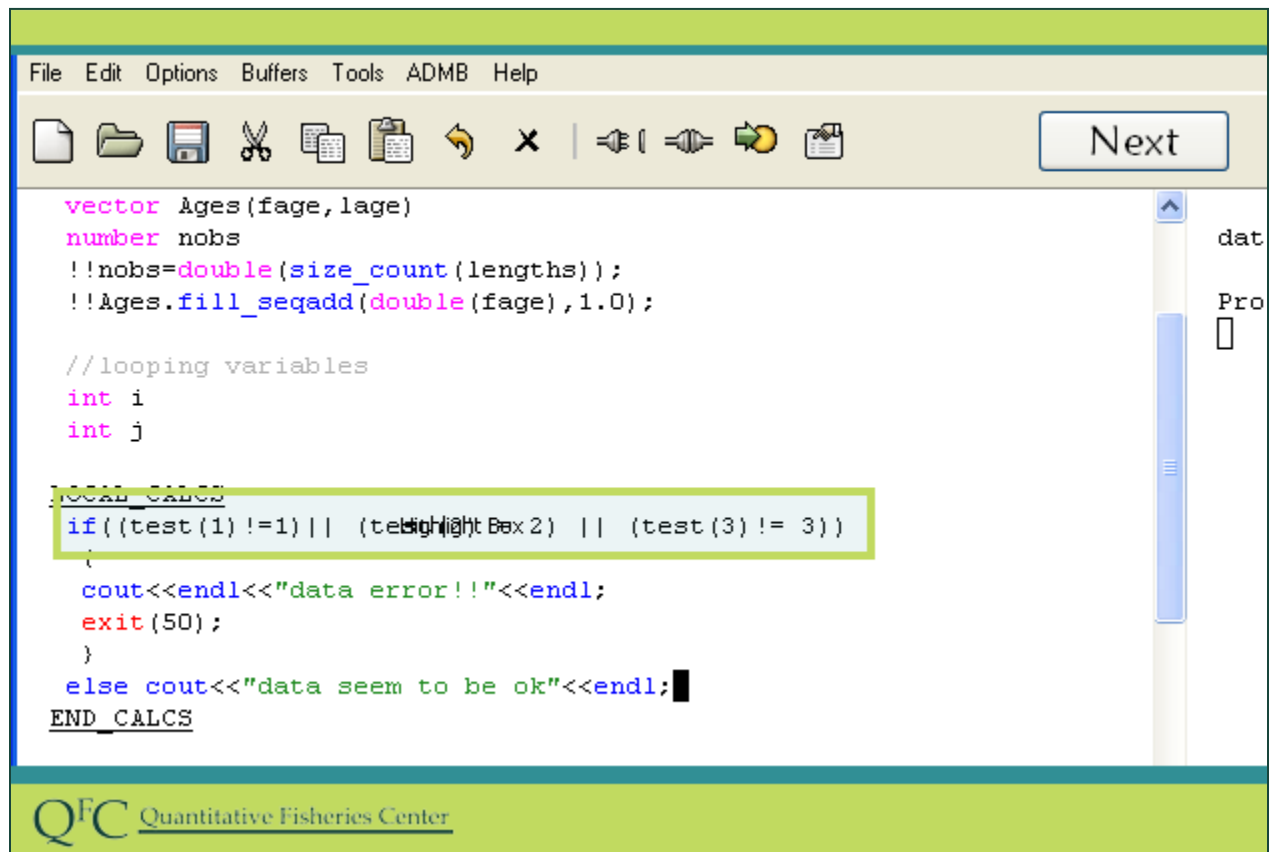
We can use two vertical bars to indicate logical OR in C++ and formally in this case we want to quit if test one is not equal to 1 OR test two is not equal to two OR test 3 is not equal to 3

Slide Code:

```
if ((test (1) != 1 || (test (2) != 2 || (test (3) != 3))
```



Notice that our if statement currently takes an action if the condition is true. We can write it to take an action if the condition is false. For example, we could have the program write that the data are ok. We can take this alternative action with an else statement that follows immediately after the if statement;



```
vector Ages(fage, lage)
number nobs
!!nobs=double(size_count(lengths));
!!Ages.fill_seqadd(double(fage),1.0);

//looping variables
int i
int j

LOCAL_CALCS
if((test(1) !=1) || (test(2) != 2) || (test(3) != 3))
{
    cout<<endl<<"data error!!"<<endl;
    exit(50);
}
else cout<<"data seem to be ok"<<endl;
END_CALCS
```

Here we add a message that the data seem to be ok if the tests turn out ok.

Slide Code:

```
else cout<<"data seem to be ok"<<endl;
```

[Next](#)

Relational Operators

<u>Operator</u>	<u>Definition</u>
==	is equal to
!=	is not equal to
<	is greater than
>	is less than
<=	is equal to or less than
>=	is equal to or greater than

In addition to the not equal to relational operator there other relational operators commonly used in if statement conditions including equal to, less than, greater than, less than or equal to, and greater than or equal to. Notice that the relational operator for equal to is two equal sign. If you use just one, your code will set what is on the left of the equal sign to what was on the right rather than evaluating the condition, which can often cause troublesome errors if you intended to evaluate equality rather than do an assignment.

[Next](#)

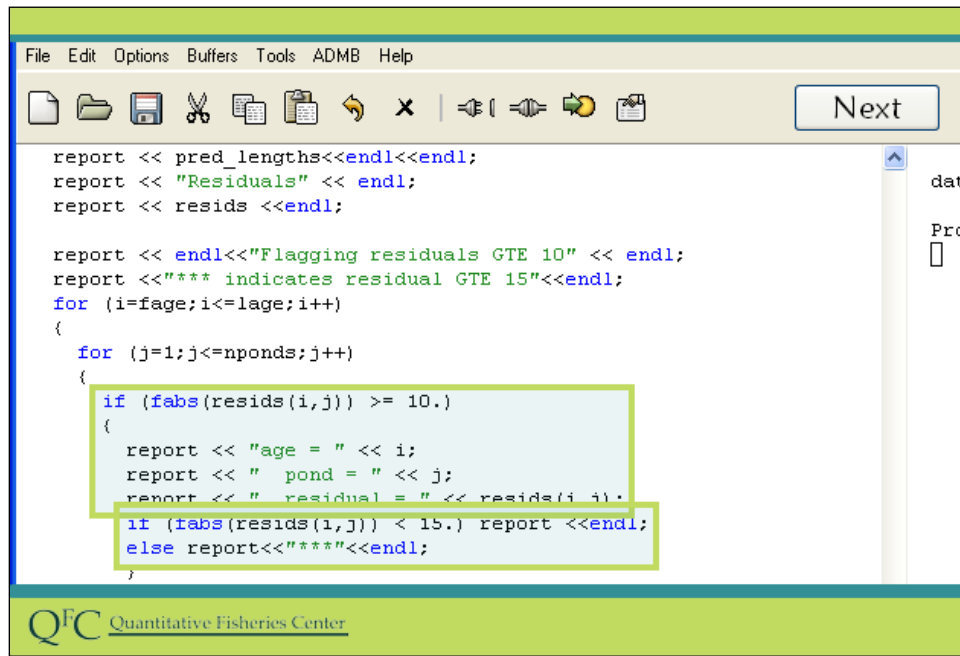
Logical Operators

<u>Operator</u>	<u>Definition</u>
	or
&&	and
!	not

If $!(a > 4)$ $y = a * 2$;

The following statement
would only be executed
only if a were not greater
than 4.

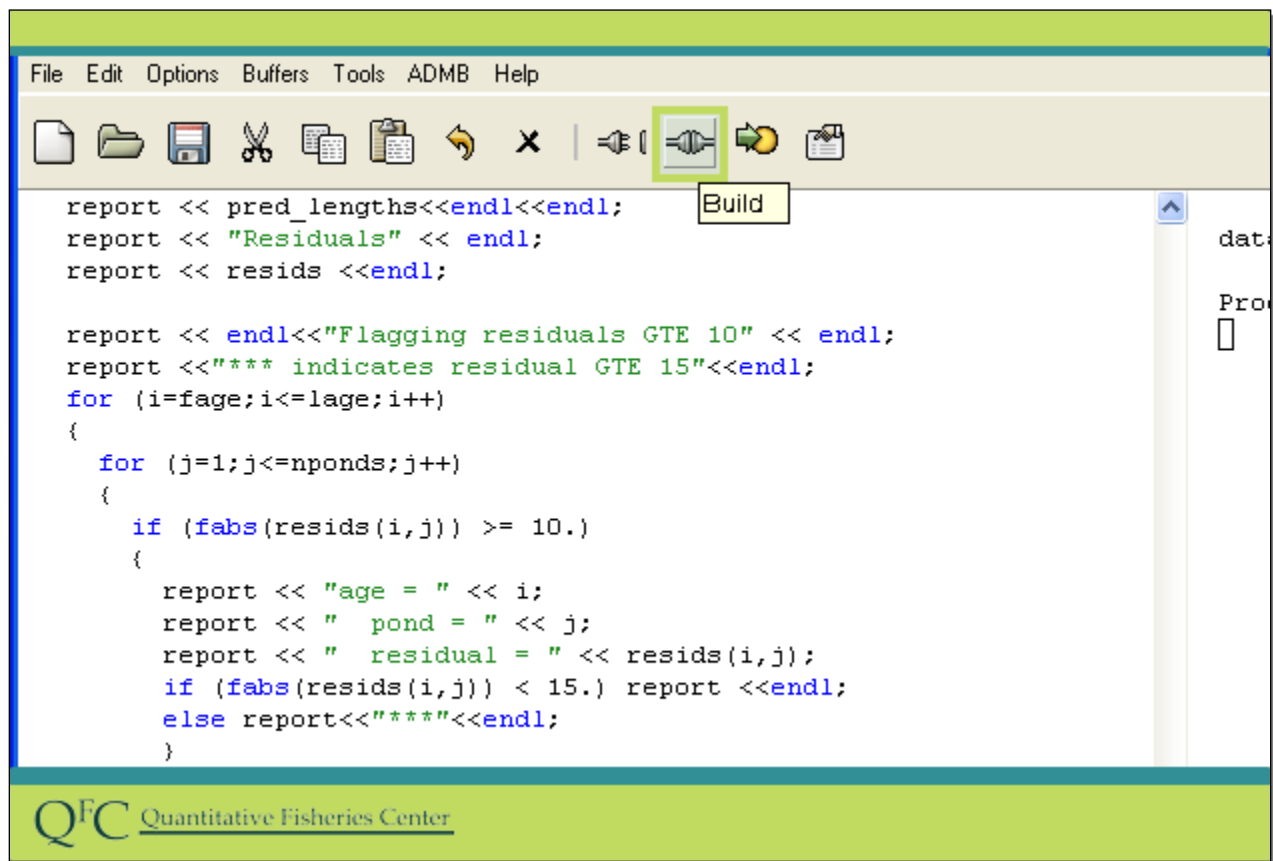
In addition to the OR logical operator, logical AND and logical NOT operators can sometimes be useful. The AND operator requires that each clause connected by the AND are true for the entire statement to be true. The NOT operator takes only one argument which immediately follows the symbol. For example the statement: If $!(a > 4)$ would cause the following statement to be executed only if a were not greater than 4.



We conclude this section on looping and conditional statements by showing one more example in the report section of our application. You can follow along or just watch if you wish. Here we are flagging large residuals and writing them out to the report file. This example illustrates that if statements can be nested within loops and that if statements can also be nested within other if statements.

Slide Code:

```
report << endl<<"Flagging residuals GTE 10" << endl;
report <<"*** indicates residual GTE 15"<<endl;
for (i=fage;i<=lage;i++)
{
  for (j=1;j<=nponds;j++)
  {
    if (fabs(resids(i,j)) >= 10.)
    {
      report << "age = " << i;
      report << " pond = " << j;
      report << " residual = " << resids(i,j);
      if (fabs(resids(i,j)) < 15.) report <<endl;
      else report<<"***"<<endl;
    }
  }
}
```

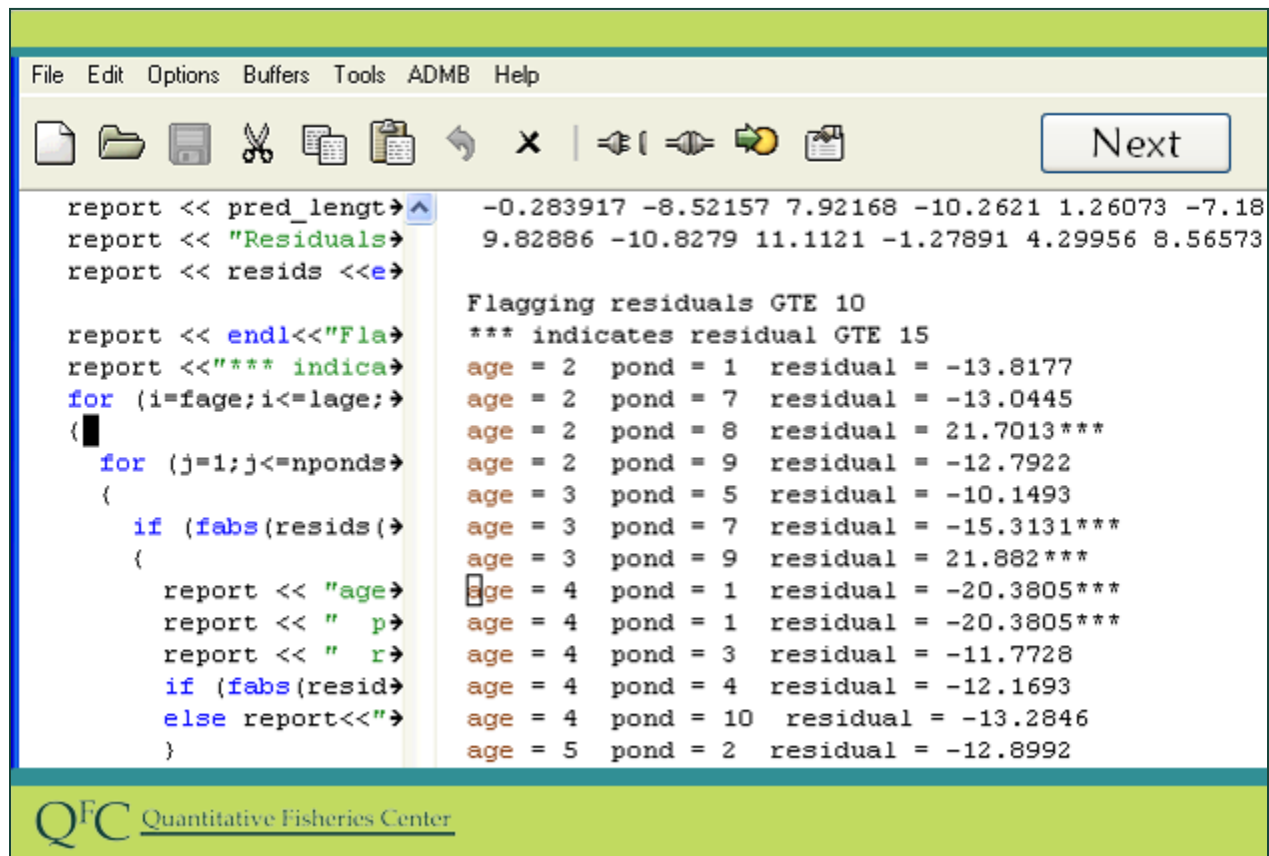


Here we are looping over ages and within ages over ponds. For each pond age combination we check if a residual is larger than or equal to 10. If it is we write it out and then check to see if its less than 15. If it's less than 15 we print a line feed and move on. If its not less than 15 we print three asterisk and a line feed and then move on.

Let's build and run the model with this addition to see what we get.

Slide Action:

Build and run the model if you added the additional code



```
File Edit Options Buffers Tools ADMB Help

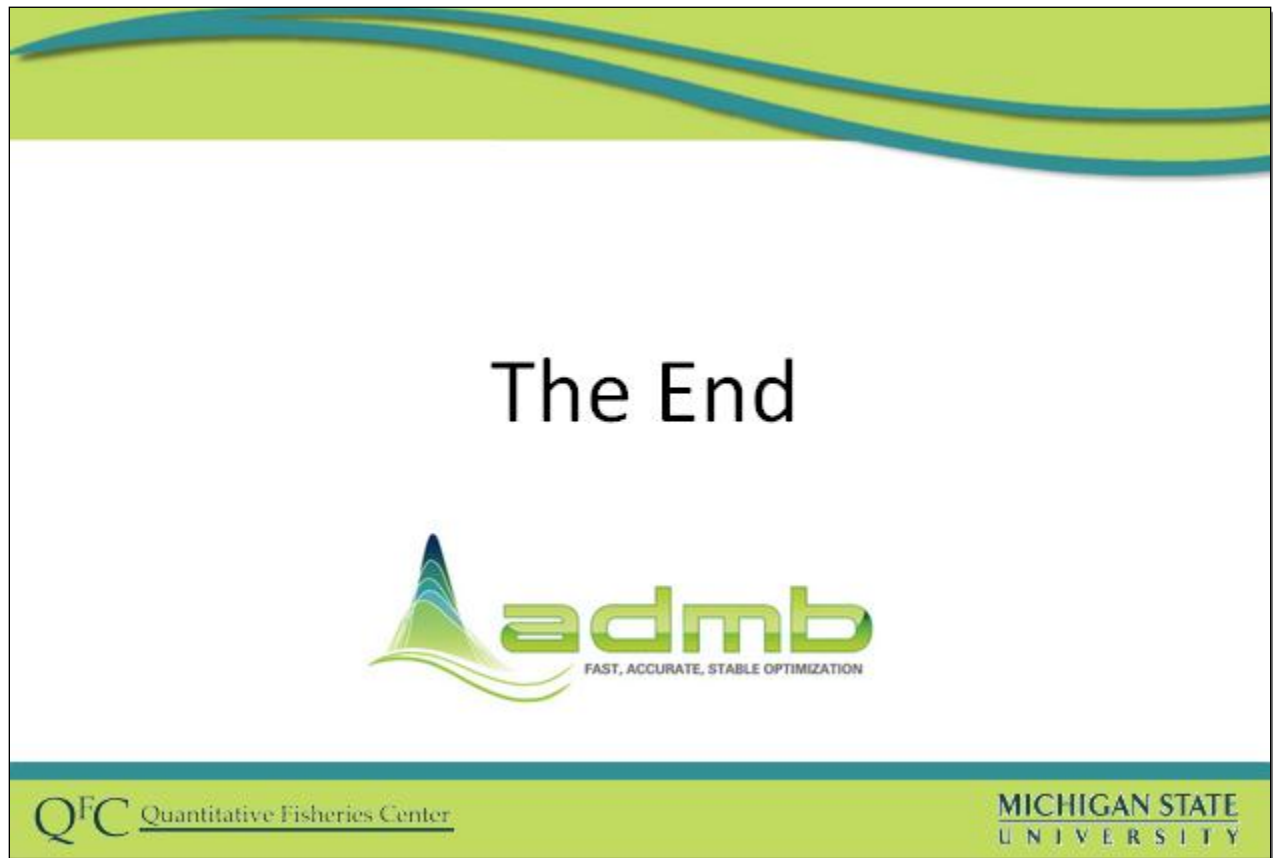
report << pred_length -0.283917 -8.52157 7.92168 -10.2621 1.26073 -7.18
report << "Residuals" 9.82886 -10.8279 11.1121 -1.27891 4.29956 8.56573
report << resid <<e>

report << endl<<"Flagging residuals GTE 10
report <<"*** indicates residual GTE 15
for (i=fage;i<=lage;i++)
{
  for (j=1;j<=nponds;j++)
  {
    if (fabs(resid[i,j]) >= 10)
    {
      report << "age = " i
      report << " pond = " j
      report << " residual = " resid[i,j]
      if (fabs(resid[i,j]) < 15) report<<"***"
    }
  }
}
```

age	pond	residual
2	1	-13.8177
2	7	-13.0445
2	8	21.7013***
2	9	-12.7922
3	5	-10.1493
3	7	-15.3131***
3	9	21.882***
4	1	-20.3805***
4	1	-20.3805***
4	3	-11.7728
4	4	-12.1693
4	10	-13.2846
5	2	-12.8992

QFC Quantitative Fisheries Center

The results are a list of the residuals of 10 or larger with those less than 15 flagged with three asterisks.



Obviously there is lots more that can be done with loops and conditional statements. Hopefully this video has provided you the basic ideas.