Nested `if` Statements

Example (in pseudocode)

IF today is Saturday or Sunday
    IF it is raining
        Sleep late
    ELSE
        Get up and go outside
ELSE
    Go to work

`if` Alternatives

```
if (month == 1)
    cout << “January”;
if (month == 2)
    cout << “February”;
if (month == 3)
    cout << “March”;
    .
    .
    .
if (month == 12)
    cout << “December”;
```

```
if (month == 1)
        cout << “January”;
else
  if (month == 2)
    cout << “February”;
  else
    if (month == 3)
      cout << “March”;
    else
      if (month == 4)
```


Multiple Alternatives

```
cout << “The recommended activity is ”;
if (temperature > 85)
    cout << “swimming.” << endl;
else if (temperature > 70)
    cout << “tennis.” << endl;
else if (temperature > 32)
    cout << “golf.” << endl;
else if (temperature > 0)
    cout << “skiing.” << endl;
else
    cout << “dancing.” << endl;
```


Beware of the dangling `else`

```
if (average >= 60.0)
   if (average < 70.0)
      cout << “Passing but marginal”;
else
   cout << “Failing”;
```


Beware of the dangling `else`

```
if (average >= 60.0)                    // Incorrect
   if (average < 70.0)
      cout << “Passing but marginal”;
else
   cout << “Failing”;
```


Beware of the dangling `else`

```
if (average >= 60.0)                    // Incorrect
   if (average < 70.0)
      cout << “Passing but marginal”;
else
   cout << “Failing”;

if (average >= 60.0)                    // Correct
{
   if (average < 70.0)
      cout << “Passing but marginal”;
}
else
   cout << “Failing”;
```

Testing the State of an I/O Stream

- C++ provides a way to check whether a stream is in the fail state. You simply use the name of the stream object as if it were a Boolean variable.

I/O Stream State Test Example

```
int main()
{
    int height, width;
    ifstream inFile;
    inFile.open("mydata.dat");
    if ( !inFile )
    {
         cout << "Can't open the input file.";
         return 1;
    }
    inFile >> height >> width;
    return 0;
}
```


Problem-Solving Case Study-Warning Notices

- Problem: In order to warn freshmen who are in danger of failing a class, your program should calculate the average of three tests. Then, the program prints out the student id, average, and either pass, fail, or marginal.
- Input: Student ID number (long) followed by three test grades.
- Output: A prompt for input, echo print input, id, average, message with possible error message for invalid data (<0).
- Discussion: Passing is >= 70.0, marginal is < 70.0 and >= 60.0, failing is < 60.0.

Main Module (Level 0)

```
Get data
Test data
IF data OK
    Calculate average
    Print message indicating status
ELSE
    Print "Invalid Data: Score(s) less than zero."
```


Print Message Indicating Status (Level 1)

```
Print average
IF average >= 60.0
    Print "Passing"
    IF average < 70.0
        Print "but marginal"
    Print '.'
ELSE
    Print "Failing"
```


Notices Program

```
//*************************************************************
// Notices program
// This program determines (1) a student's average based on
// three test scores and (2) the student's passing/failing
// status
//*************************************************************

#include <iostream>
#include <iomanip>    // For setprecision()
using namespace std;

int main()
{
    float average;       // Average of three test scores
    long  studentID;     // Student's identification number
    int   test1;         // Score for first test
    int   test2;         // Score for second test
```

More Notices Program

```
int   test3;          // Score for third test
bool  dataOK;         // True if data is correct

cout << fixed << showpoint;         // Set up floating pt.
                                    //   output format
// Get data
cout << "Enter a Student ID number and three test scores:"
     << endl;
cin >> studentID >> test1 >> test2 >> test3;
cout << "Student number: " << studentID << "  Test Scores: "
     << test1 << ", " << test2 << ", " << test3 << endl;

// Test data
if (test1 < 0 || test2 < 0 || test3 < 0)
    dataOK = false;
else
    dataOK = true;
```


Notices Program (continued)

```
if (dataOK)
{
    // Calculate average
    average = float(test1 + test2 + test3) / 3.0;

    // Print message
    cout << "Average score is " << setprecision(2)
         << average << "--";

    if (average >= 60.0)
    {
        cout << "Passing";            // Student is passing
        if (average < 70.0)
            cout << " but marginal";  // But marginal
        cout << '.' << endl;
    }
```


Notices Program (The End)

```
    else                        // Student is failing
        cout << "Failing." << endl;
}
else                            // Invalid data
    cout << "Invalid Data:  Score(s) less than zero."
         << endl;
return 0;
}
```

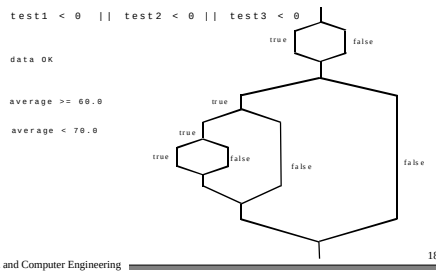

Testing and Debugging Phases

- Problem-Solving Phase
  - Algorithm Walk-Through
    - Precondition - an assertion which must be true before a module is executed in order for the module to execute correctly.
    - Postcondition - an assertion which will be true after the module has executed, if it has done its job properly.

Testing and Debugging Phases

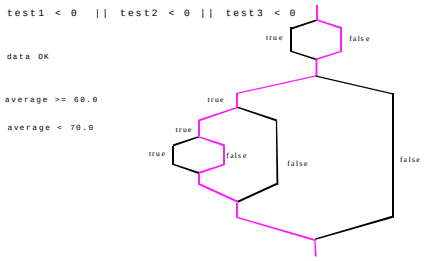
- Implementation Phase
  - Code Walk-Through - make sure that you've faithfully reproduced the algorithm.
  - Execution Trace - play computer and execute each statement of a program by hand.
  - Testing Selection Control Structures - you need to execute each branch at least once and verify the results.

Testing Selection Control Structures



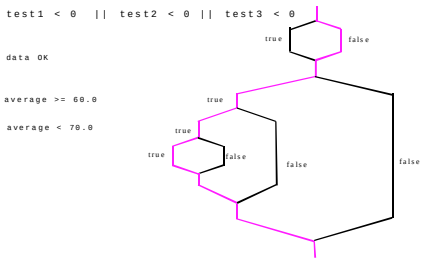
UAH CPE 112

Test Set 1
test1 = 100, test2 =100, test3 = 100



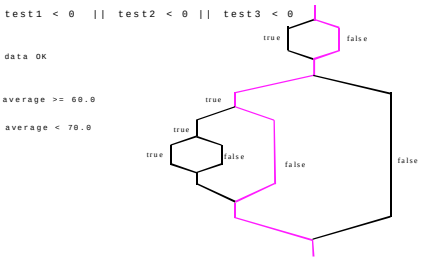
UAH CPE 112

Test Set 2
test1 = 60, test2 = 60, test3 = 63



UAH CPE 112

Test Set 3
test1 = 50, test2 = 50, test3 = 50



UAH CPE 112

Test Set 4
test1 = -50, test2 = 50, test3 = 50