

Flow of Control

- ❖ **Definition: The sequential execution of statements in a program**
 - ◆ **Sequential Control Structure (Top-Bottom)**
 - ◆ It is characterized by a flow chart construct without branches.
 - ◆ **Selection Control Structure (Branching)**
 - ◆ Decision making control
 - ◆ Tests an Assertion Statement
 - ▶ Evaluated as True or False (Humans)
 - ▶ Evaluated as Yes or No (Humans)
 - ▶ Evaluated as 1 or 0 (Computers)

Copyright © 2007 R.M. Laurie | 1

Relational Operators

- ❖ Relational operators are used to compare two data objects.
- ❖ The result of the comparison is either **true** or **false**.

== Equal to	!= Not Equal to
> Greater	>= Greater or Equal
< Less	<= Less or Equal
- ❖ Note the difference between **==** and **=** operator

Copyright © 2007 R.M. Laurie | 2

Arithmetic Operators Precedence

(Highest to Lowest)

()	Defines order of operation
-	Minus (unary)
* / %	Multiply, Division, Remainder
+ -	Addition, Subtraction
< <= > >=	} Relational Operators
== !=	
=	Assignment

Copyright © 2007 R.M. Laurie | 3

if Selection Control Structure

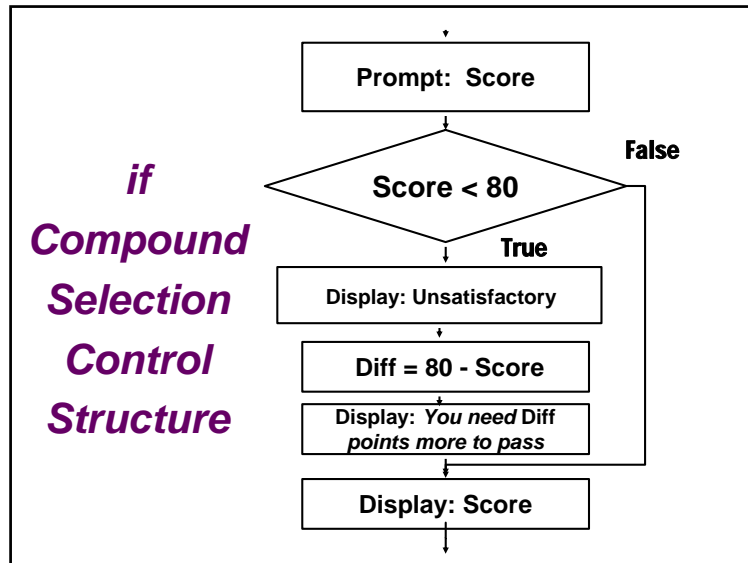
- ❖ Characterized by a diamond shaped flow chart construct, containing an assertions with two possible outcomes branches (True or False).

```

graph TD
    A[Prompt: Score] --> B{Score >= 90}
    B -- True --> C[Display: Grade= A]
    B -- False --> D[ ]
    C --> E[ ]
    D --> E
    style D fill:none,stroke:none
    style E fill:none,stroke:none
            
```

if(Score >= 90)
document.write("Grade = A");

Copyright © 2007 R.M. Laurie | 4

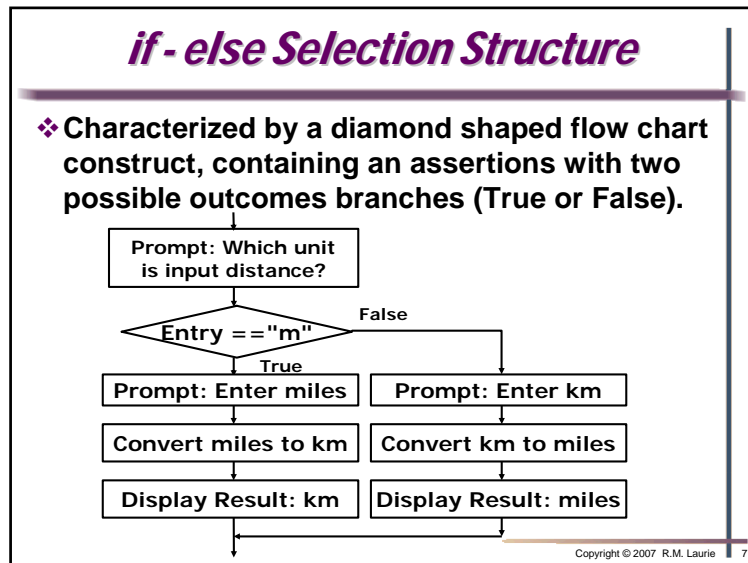


if Selection Control Structure
(Compound statement syntax)

```

Score = parseFloat(window.prompt( "Enter Score", "0" ));
if(Score < 80)
{
    document.writeln("<h2 style='color: #CC0000'\>"
        + "Exam Result Unsatisfactory</h2>");
    Diff = 80 - Score;
    document.writeln("<p>You need " + Diff
        + " to continue to next chapter</p>");
}
document.writeln("<p>You Exam Score was " + Score
    + "</p>");
    
```

Copyright © 2007 R.M. Laurie 6

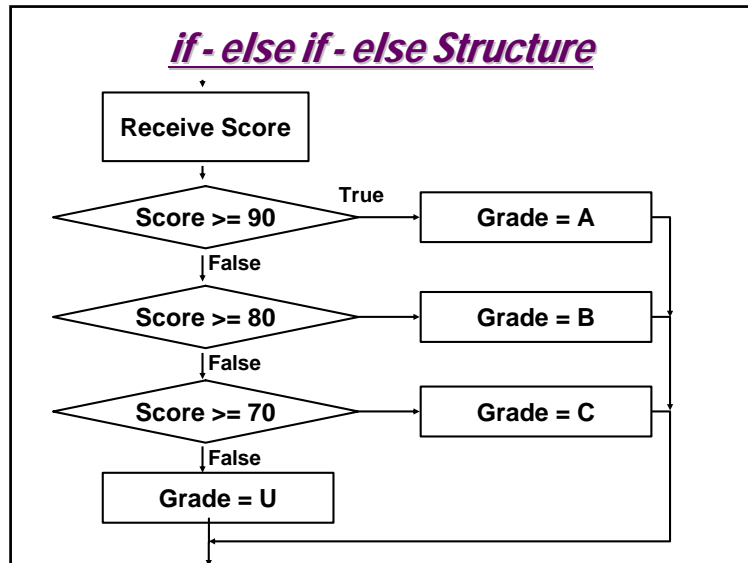


if - else Structure Selection

```

<head>
<script type="text/javascript">
var Entry, Result;
Entry = window.prompt("Is input distance miles or km? (m or k)", "m");
if(Entry == "m")
{
    Entry = parseFloat(window.prompt("Enter miles: ", "0"));
    Result = Entry * 1.609;
    document.writeln("<p>"+Entry+" miles = "+Result+" km</p>");
}
else
{
    Entry = parseFloat(window.prompt("Enter kilometers: ", "0"));
    Result = Entry / 1.609;
    document.writeln("<p>"+Entry+" km = "+Result+" miles</p>");
}
document.writeln("<p>Reload for another conversion</p>");
</script>
</head>
<body>
</body>
    
```

Copyright © 2007 R.M. Laurie 8



```

<head>
<title>Grade Determination</title>
<script type="text/javascript">
var Score, Grade;
Score = parseFloat(window.prompt( "Enter Score", "0" ));
if(Score >= 90)
    Grade = "A";
else if(Score >= 80)
    Grade = "B";
else if(Score >= 70)
    Grade = "C";
else
    Grade = "U";
document.writeln("<h2>For the score = " + Score
+ " <br/>Your letter grade is " + Grade + "</h2>" );
</script>
</head>
<body>
<p>Click Refresh (or Reload) to run the script again</p>
</body>
    
```

- ### *Design Phase*
- ❖ Write Program Specifications
 - ◆ Analysis of requirements
 - ◆ Program specifications description
 - ◆ Describe what the goals of the program
 - ◆ Describe appearance of input and output
 - ❖ Algorithm Design
 - ◆ Mathematical Analysis and Algorithm
 - ◆ Flow Chart to describe event sequencing
 - ❖ Verify algorithm
 - ◆ Test with known data
 - ◆ Solve manually
- Copyright © 2007 R.M. Laurie 11

Algorithm Design - Mathematical

- ❖ Mathematical Description
 - ◆ Boiling point
F = 212
C = 100
 - ◆ Freezing point
F = 32
C = 0

$$Y = MX + B$$

$$F = (180 / 100) C + 32$$

$$= (9/5) C + 32$$

$$= 1.8 C + 32$$

Copyright © 2007 R.M. Laurie 12

Verify Algorithm

- ❖ Testing with known data
 - ◆ Boiling point
F = 212 C = 100
 - ◆ Freezing point
F = 32 C = 0
 - ◆ Collect Data
 - ◆ Bank thermometer
 - ◆ Radio weather report
- ❖ Solve manually by hand using calculator

Copyright © 2007 R.M. Laurie 13

Implementation Phase

- ❖ Translate Algorithm into Code
 - ◆ Create source code file with syntax of JavaScript language and HTML
 - ◆ Run to detect *syntax errors*
- ❖ Test Program
 - ◆ Test with known data
 - ◆ Detects program *logic errors*
 - ◆ Often requires several iterations
 - ◆ May require re-evaluation of specifications and algorithms

Copyright © 2007 R.M. Laurie 14

Coding First Is No Shortcut?

The diagram illustrates a process flow starting with a yellow arrow labeled 'Shortcut?' pointing to the word 'CODE'. From 'CODE', the path goes down to 'THINKING', then up to 'DEBUG', then right to 'REVISE', then up to 'DEBUG', then right to 'DEBUG', then up to 'REVISE', then right to 'GOAL'. Below the 'GOAL' is the word 'TEST'. The word 'CODE' also appears at the bottom of the path. The path is represented by a jagged line, suggesting a non-linear or iterative process.

Copyright © 2007 R.M. Laurie 15

Conditional Exercises

- ❖ Create program that converts temperatures between Fahrenheit and Celsius number systems
 - ◆ Prompt for which Conversion to perform
 - ◆ Prompt for the temperature to convert
 - ◆ Convert and display the results
- ❖ Create an employee's pay program
 - ◆ Prompt for name, pay rate, and hours
 - ◆ Overtime rate is 1.5x normal pay rate
 - ◆ Subtract 15% withholding tax
 - ◆ Calculate pay check amount

Copyright © 2007 R.M. Laurie 16