1.19 A Simple Program: Printing a Line of Text

- **`std::cout`**
  - Standard output stream object
  - “Connected” to the screen
  - **`std::`** specifies the "namespace" which **`cout`** belongs to
    - **`std::`** can be removed through the use of **`using`** statements
- **`<<`**
  - Stream insertion operator
  - Value to the right of the operator (right operand) inserted into output stream (which is connected to the screen)
  - **`std::cout << “Welcome to C++!\n”;`**
- **`\`**
  - Escape character
  - Indicates that a “special” character is to be output

1.19 A Simple Program: Printing a Line of Text

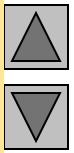
Escape Sequence	Description
`\n`	Newline. Position the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Used to print a backslash character.
`\"`	Double quote. Used to print a double quote character.

- There are multiple ways to print text
  - Following are more examples

```
1    // Fig. 1.4: fig01_04.cpp
2    // Printing a line with multiple statements
3    #include <iostream>
4
5    int main()
6    {
7       std::cout << "Welcome ";
8       std::cout << "to C++!\n";
9
10      return 0;   // indicate that program ended successfully
11   }
```

```
Welcome to C++!
```

Unless new line `'\n'` is specified, the text continues on the same line.



Outline

1. Load `<iostream>`
2. `main`
   - 2.1 Print `"Welcome"`
   - 2.2 Print `"to C++!"`
   - 2.3 newline
   - 2.4 exit (`return 0`)

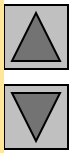
Program Output

```
1   // Fig. 1.5: fig01_05.cpp
2   // Printing multiple lines with a single statement
3   #include <iostream>
4
5   int main()
6   {
7      std::cout << "Welcome\nto\n\nC++!\n";
8
9      return 0;   // indicate that program ended successfully
10  }
```

```
Welcome
to

C++!
```

Multiple lines can be printed with one statement.



Outline

1. Load `<iostream>`
2. `main`
   - 2.1 Print `"Welcome"`
   - 2.2 newline
   - 2.3 Print `"to"`
   - 2.4 newline
   - 2.5 newline
   - 2.6 Print `"C++!"`
   - 2.7 newline
   - 2.8 exit (`return 0`)

Program Output

1.20 Another Simple Program: Adding Two Integers

- Variables
  - Location in memory where a value can be stored for use by a program
  - Must be declared with a name and a data type before they can be used
  - Some common data types are:
    - **`int`** - integer numbers
    - **`char`** - characters
    - **`double`** - floating point numbers
  - Example: **`int myvariable;`**
    - Declares a variable named **`myvariable`** of type **`int`**
  - Example: **`int variable1, variable2;`**
    - Declares two variables, each of type **`int`**

1.20 Another Simple Program: Adding Two Integers

- **>>** (stream extraction operator)
  - When used with **`std::cin`**, waits for the user to input a value and stores the value in the variable to the right of the operator
  - The user types a value, then presses the *Enter* (Return) key to send the data to the computer
  - Example:

```
int myVariable;
std::cin >> myVariable;
```

    - Waits for user input, then stores input in **`myVariable`**
- **=** (assignment operator)
  - Assigns value to a variable
  - Binary operator (has two operands)
  - Example:

```
sum = variable1 + variable2;
```

```
1   // Fig. 1.6: fig01_06.cpp
2   // Addition program
3   #include <iostream>
4
5   int main()
6   {
7      int integer1, integer2, sum;          // declaration
8
9      std::cout << "Enter first integer\n"
10     std::cin >> integer1;
11     std::cout << "Enter second integer\n"
12     std::cin >> integer2;                 // read an integer
13     sum = integer1 + integer2;            // assignment of sum
14     std::cout << "Sum is " << sum << std::endl; // print sum
15
16     return 0;   // indicate that program ended s
17  }
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

Notice how **std::cin** is used to get user input.

std::endl flushes the buffer and prints a newline.

Variables can be output using **std::cout << variableName**.

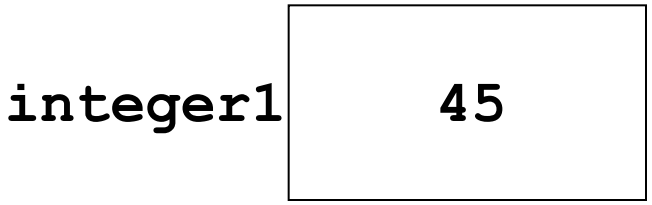
Outline

1. Load <iostream>
2. main
   - 2.1 Initialize variables integer1,
   - first integer"
   - 2.2.1 Get input
   - 2.3 Print "Enter second integer"
   - 2.4 Add variables and put result into sum
   - 2.5 Print "Sum is"
   - 2.5.1 Output sum
   - 2.6 exit (return 0)

Program Output

1.21 Memory Concepts

- Variable names
  - Correspond to locations in the computer's memory
  - Every variable has a name, a type, a size and a value
  - Whenever a new value is placed into a variable, it



- A visual representation

1.22 Arithmetic

- Arithmetic calculations
  - Use **`*`** for multiplication and **`/`** for division
  - Integer division truncates remainder
    - **`7 / 5`** evaluates to 1
  - Modulus operator returns the remainder
    - **`7 % 5`** evaluates to 2
- Operator precedence
  - Some arithmetic operators act before others (i.e., multiplication before addition)
    - Be sure to use parenthesis when needed
  - Example: Find the average of three variables a, b and c
    - Do not use: **`a + b + c / 3`**
    - Use: **`(a + b + c ) / 3`**

1.22 Arithmetic

- Ari

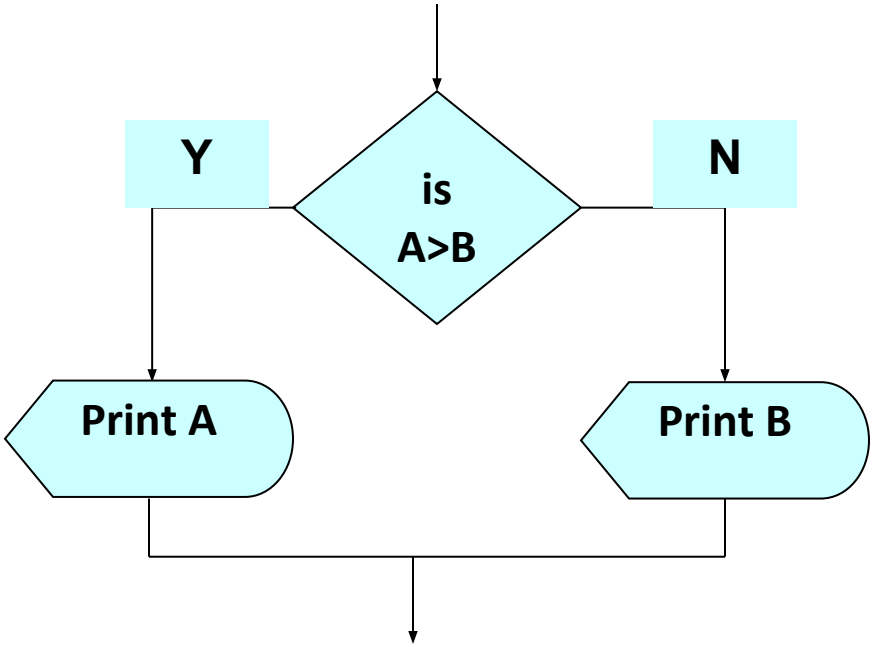
C++ operation	Arithmetic operator	Algebraic expression	C++ expression
Addition	+	*f + 7*	`f + 7`
Subtraction	-	*p – c*	`p - c`
Multiplication	*	*bm*	`b * m`
Division	/	*x / y*	`x / y`
Modulus	%	*r mod s*	`r % s`

Operator(s)	Operation(s)	Order of evaluation (precedence)
`()`	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
`*`, `/`, or `%`	Multiplication Division Modulus	Evaluated second. If there are several, they re evaluated left to right.
`+` or `-`	Addition Subtraction	Evaluated last. If there are several, they are evaluated left to right.

DECISION STRUCTURES

- The expression A>B is a logical expression
- *it describes a **condition** we want to test*
- ***if A>B is true (if A is greater than B)** we take the action on left*
- print the value of A
- ***if A>B is false (if A is not greater than B)** we take the action on right*
- print the value of B

DECISION STRUCTURES



IF–THEN–ELSE STRUCTURE

- The structure is as follows

If condition then

true alternative

else

false alternative

endif

IF–THEN–ELSE STRUCTURE

- The algorithm for the flowchart is as follows:

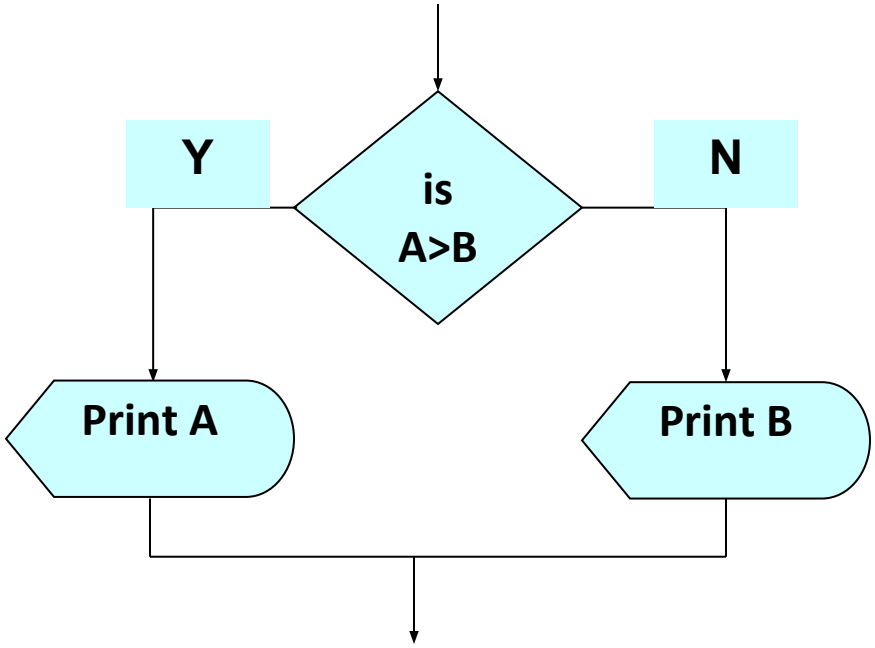
If A>B then

print A

else

print B

endif



Relational Operators

Relational Operators	
Operator	**Description**
>	**Greater than**
<	**Less than**
=	**Equal to**
≥	**Greater than or equal to**
≤	**Less than or equal to**
≠	**Not equal to**

Example 5

- Write an algorithm that reads two values, determines the largest value and prints the largest value with an identifying message.

ALGORITHM

Step 1: *Input* VALUE1, VALUE2

Step 2: *if (*VALUE1 > VALUE2) *then*

MAX = VALUE1

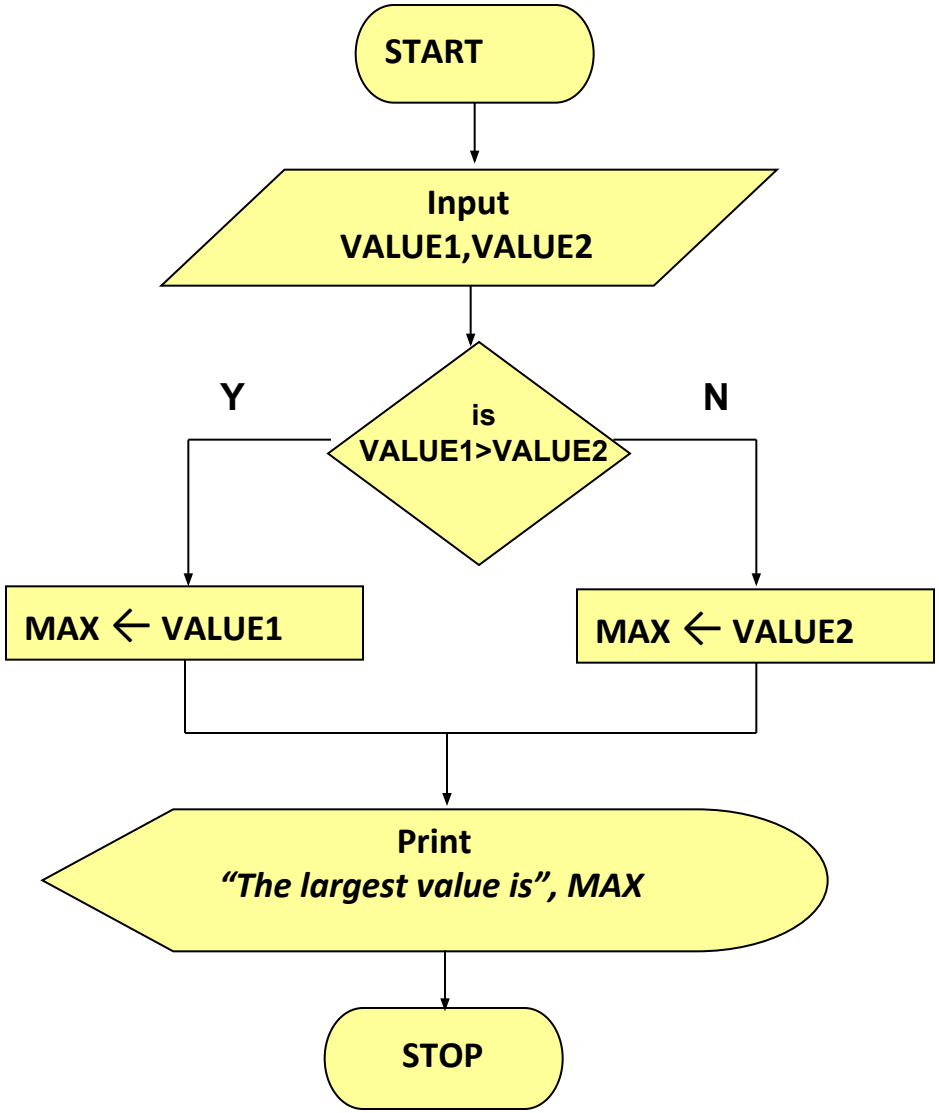
else

MAX = VALUE2

endif

Step 3: *Print "The largest value is", MAX*

Example 5


START
Input
VALUE1,VALUE2
Y
is
VALUE1>VALUE2
N
MAX ← VALUE1
MAX ← VALUE2
Print
"The largest value is", MAX
STOP

NESTED IFS

- One of the alternatives within an IF–THEN–ELSE statement
  - may involve further IF–THEN–ELSE statement

Pseudocode & Algorithm

- **Example 1:** Write an algorithm to determine a read student's marks in 4 subjects and compute the average marks. Check if the average marks are more than 50 then the students status is pass else fail.

Pseudocode & Algorithm

Pseudocode:

- *Input a set of 4 marks*
- *Calculate their average by summing and dividing by 4*
- *if average is below 50*

  Print "FAIL"

  else

  Print "PASS"

Pseudocode & Algorithm

- Detailed Algorithm
- Step 1: Input M1,M2,M3,M4
  Step 2: GRADE =(M1+M2+M3+M4)/4
  Step 3: if (GRADE < 50) then
  Print “FAIL”
  else
  Print “PASS”
  endif

Assignment in computre science and equality in mathematics

a) The following instructions are the same in mathematics.

A=B B=A

not in computer science.

Let A=B is different from Let B=A

b) In mathematics we work with relations.

A relation B=A+1 means that it is true all the time

In computer science, we work with assignments. We can have:

A=5

B=A+1

A=2

The relation B=A+1 is true only after the second instruction and before the third one.

Assignment in computer science and equality in mathematics

c) The instruction A=A+3 is false in mathematics.
In computer science Let A=A+3 means: the new value of A is equal to the old one plus three.

d) The instruction A+5=3 is allowed in mathematics (it is an equation).
Let A+5=3 has no meaning in computer science (the left side must be a variable).

1.23 Decision Making: Equality and Relational Operators

- **`if`** structure
  - Test conditions truth or falsity. If condition met execute, otherwise ignore
- Equality and relational operators
  - Lower precedence than arithmetic operators
- Table of relational operators on next slide

1.23 Decision Making: Equality and Relational Operators

Standard algebraic equality operator or relational operator	C++ equality or relational operator	Example of C++ condition	Meaning of C++ condition
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

1.23 Decision Making: Equality and Relational Operators

- **`using`** statements
  - Eliminate the need to use the **`std::`** prefix
  - Allow us to write cout instead of **`std::cout`**
  - To use the following functions without the **`std::`** prefix, write the following at the top of the program

```
using std::cout;
using std::cin;
using std::endl;
```

```
1  // Fig. 2.14: fig02_14.cpp
2  // Using if statements, relational
3  // operators, and equality operators
4  #include <iostream>
5
6  using std::cout;  // program uses cout
7  using std::cin;   // program uses cin
8  using std::endl;  // program uses endl
9
10 int main()
11 {
12    int num1, num2;
13
14    cout << "Enter two integers, and I will tell you\n"
15         << "the relationships they satisfy: ";
16    cin >> num1 >> num2;   // read two integers
17
18    if ( num1 == num2 )
19       cout << num1 << " is equal to " << num2 << endl;
20
21    if ( num1 != num2 )
22       cout << num1 << " is not equal to " << num2 << e
23
24    if ( num1 < num2 )
25       cout << num1 << " is less than " << num2 << endl
26
27    if ( num1 > num2 )
28       cout << num1 << " is greater than " << num2 << e
29
30    if ( num1 <= num2 )
31       cout << num1 << " is less than or equal to "
32            << num2 << endl;
33
```

Notice the **using** statements.

```
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
```

```
3 is not equal to 7
```

```
3 is less than 7
```

```
3 is less than or equal to 7
```

The **if** statements test the truth of the condition. If it is executed. If not, body is skipped.

To include multiple statements in a body, delineate them with braces **{ }**.

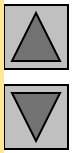
Outline

1. Load <iostream>

2.1 Initialize num1 and num2

2.1.1 Input data

```
34      if ( num1 >= num2 )
35         cout << num1 << " is greater than or equal to "
36              << num2 << endl;
37
38      return 0;   // indicate that program ended successfully
39   }
```



Outline

2.3 exit (`return 0`)

Program Output

```
Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 7 7
7 is equal to 7
7 is less than or equal to 7
7 is greater than or equal to 7
```

Example 7

- **Flowchart: Draw the flowchart of the above algorithm?**

Introduction

- This chapter covers the Visual Basic decision statements
  - **`If…Then`**
  - **`If…Then…Else`**
  - **`If…Then…ElseIf`**
  - **`Select Case`**
- It also discusses the use of
  - Radio Buttons
  - Message Boxes

If...Then Selection Statement

- A selection statement chooses among alternative courses of action in an application.

```
If student's grade is greater than or equal to 60
        Display "Passed"
```

- The preceding pseudocode *If* statement may be written in Visual Basic as

```
If studentGrade >= 60 Then
        displayLabel.Text = "Passed"
```

Use of goto

```
/ This program calculates the average of positive numbers entered by user.
 # include <iostream>
using namespace std;
int main()
{
   float  average;
    int i, n, num , sum = 0;
  cout << "Maximum number of inputs: ";
  cin >> n;
    i = 0;
  jump:  cout << "Enter n" << i+1 << ": ";
    cin >> num;
    if(num  >0) sum += num;
     i=i+1;
     if (i<n) goto jump;
   average = sum / i;
   cout << "\n Average = " << average;
   return 0;
}
```

```
/ This program calculates the average of positive numbers entered by user.
 # include <iostream>
using namespace std;
int main()
{
   float  average;
    int i, n, num , sum = 0;
  cout << "Maximum number of inputs: ";
  cin >> n;
    i = 1;
  jump:  cout << "Enter n" << i << ": ";
    cin >> num;
    if(num  >0) sum += num;
     i=i+1;
     if (i<=n) goto jump;
   average = sum / (i-1);
   cout << "\n Average = " << average;
   return 0;
}
```

Flow Control

- Simple Branching

```
if (value < 0)
   {
        cout << " Number is negative ";
   }
else
   {
        cout << " Number is positive ";
   }
```

Use of for loop

```
#include <iostream>
using namespace std;
int main ()
{int i , n, num, sum=0;
cin>>n;
for (int i=1; i<=n; i++)
 { cin>>num;
    If(num>0)
    sum=sum+num;}
cout << sum;
return 0;}
```

Example 6

- Write an algorithm that reads **three** numbers and prints the value of the largest number.

Example 6

Step 1: ***Input*** **N1, N2, N3**
Step 2: ***if (*****N1>N2) *then***
if (**N1>N3) *then***
MAX = N1 [N1>N2, N1>N3]
else
MAX = N3 [N3>N1>N2]
endif
else
if (**N2>N3) *then***
MAX = N2 [N2>N1, N2>N3]
else
MAX = N3 [N3>N2>N1]
endif
endif
Step 3: ***Print "The largest number is", MAX***

Example 6

Step 1: ***Input*** **N1, N2, N3**

Step 2: ***if (*****N1>N2)**

```
{
    if (N1>N3)
      MAX = N1    //[N1>N2, N1>N3]
    else
      MAX = N3
}                          //[N3>N1>N2]
else
  {
      if (N2>N3)
      MAX = N2    //[N2>N1, N2>N3]
    else
      MAX = N3    //[N3>N2>N1]
  }
```

Step 3: ***Print "The largest number is", MAX***

Alternative soln

Begin

Input N1,N2,N3

Max=N1;

If max<N2 then max=N2

If max<N3 then max=N3

Print max

Stop

Example 6

- **Flowchart: Draw the flowchart of the above Algorithm.**

Example 7

- Write and algorithm and draw a flowchart to

a) read an employee name (NAME), overtime hours worked (OVERTIME), hours absent (ABSENT) and

b) determine the bonus payment (PAYMENT).

amount >30 , x=5000

amount>20 and <=30 , x =2000

amount<=20 , x=1000

If >30 amount =5000
If <=30 and >20 amount=2000
If <=20 amount =1000

If(x<=20) amt=1000
If (x >20) and (x<=30) amt=2000
If (x>30) amt=5000

If (x>30) amt=5000
If (x<=30) and (x>20) amt=2000
If(x<=20) amt=1000

If(x<=20) amt=1000
Else
If (x >20) and (x<=30) amt=2000
Else
amt=5000

If (x>30) amt=5000
Else if(x>20) amt=2000
Else amt=1000

Example 7

Bonus Schedule	
OVERTIME – (2/3)*ABSENT	Bonus Paid
>40 hours >30 but ≤ 40 hours >20 but ≤ 30 hours >10 but ≤ 20 hours ≤ 10 hours	$50 $40 $30 $20 $10

Step 1: *Input* NAME,OVERTIME,ABSENT
Step 2: *if* (OVERTIME–(2/3)*ABSENT > 40) *then*
PAYMENT=50
*else if (*OVERTIME–(2/3)*ABSENT > 30) *then*
PAYMENT = 40
*else if (*OVERTIME–(2/3)*ABSENT > 20) *then*
PAYMENT =30
*else if (*OVERTIME–(2/3)*ABSENT > 10) *then*
PAYMENT =20
else
PAYMENT =10
end if
Step 3: *Print* “Bonus for”, NAME “is $”, PAYMENT

If...Then...Else Selection Statement and Conditional If Expressions

- **Nested If...Then...Else statements** test for multiple conditions by placing If...Then...Else statements inside other If...Then...Else statements.

```
If student's grade is greater than or equal to 90
      Display "A"
Else
      If student's grade is greater than or equal to 80
        Display "B"
    Else
       If student's grade is greater than or equal to 70
                Display "C"
     Else
                If student's grade is greater than or equal to 60
                  Display "D"
            Else
                  Display "F"
```

If…Then Examples

```
'Bonus, 12% commission rate, and a day off
'awarded if sales greater than 50000
If sales > 50000
   {getsBonus = True
   commissionRate = 0.12
   daysOff = daysOff + 1
}
else{getsBonus = false
   commissionRate = 0.02
   daysOff = daysOff + .5
}
e
```

Use of a Trailing Else

- If average is greater than 100, lblGrade is assigned the text “Invalid”

```
If sngAverage < 60 Then
     Text = "F"
ElseIf sngAverage < 70 Then
   Text = "D"
ElseIf sngAverage < 80 Then
   Text = "C"
ElseIf sngAverage < 90 Then
   Text = "B"
ElseIf sngAverage <= 100 Then
   Text = "A"
Else
   Text = "Invalid"
End If
```

Selection Structures

- The **If...Then** selection structure performs (selects) an action (or sequence of actions) based on a condition.
  - If the condition evaluates to True, the actions specified by the If...Then structure are executed. If evaluates to False, the actions structure are skipped.
- The **If...Then...Else** selection structure
  - Performs an action (or sequence of actions) if a condition is true and performs a different action (or sequence of actions) if the condition is false.