|  |  |
| --- | --- |
| **CS118 Programming Fundamentals** | **LAB 02** PRE-PROGRAMMING PHASE |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES** | |

# What Problem Can Be Solved By Computer

# When the solution can be produced by a set of step-by-step procedures or actions.

# This step-by-step action is called an algorithm.

# The algorithm will process some inputs and produced output.

# Solving problem by computer undergo two phases:

# Phase 1:

# Organizing the problem or pre-programming phase.

# Phase 2:

# Programming phase.

# PRE-PROGRAMMING PHASE

# This phase requires five steps:

# Analyzing the problem.

# Developing the Hierarchy Input Process Output (HIPO) chart or Interactivity Chart (IC).

# Developing the Input-Process-Output (IPO) Chart.

# Drawing the Program flowcharts.

# Writing the algorithms

# Analyzing The Problem

# Understand and analyze the problem to determine whether it can be solved by a computer.

# Analyze the requirements of the problem.

# Identify the following:

# Data requirement.

# Processing requirement or procedures that will be needed to solve the problem.

# The output.

# All These requirements can be presented in a Problem Analysis Chart (PAC)

|  |  |  |
| --- | --- | --- |
| Data | Processing | Output |
| Given in the problem or provided by the user. | List of processing required or procedures. | Output requirement. |

# Example # 01:

# Write a Problem Analysis Chart (PAC) to find an area of a circle where

# area = pi \* radius \* radius

|  |  |  |
| --- | --- | --- |
| Data | Processing | Output |
| radius | area = 3.14 x radius x radius | area |

# Developing the Hierarchy Input Process Output (HIPO) or Interactivity Chart

# The problem is normally big and complex.

# Thus, requires big program.

# Thus, the processing can be divided into subtasks called modules.

# Each module accomplishes one function.

# These modules are connected to each other to show the interaction of processing between the modules.

# Main/control module controls the flow all other modules.

# The IC is developed using top-down-method: top to down left to right order (also refer to order of processing).

# Modules are numbered, marked for duplication, repetition or decision.

# The interaction will form a hierarchy, called Hierarchy Input Process Output Chart (HIPO) or Interactivity Chart (IC). Programming which use this approach (problem is divided into subtasks) is called *Structured Programming*.

# 

# Example # 02:

# Write a Hierarchy Input Process Output (HIPO) to find an area of a circle where

# area = pi \* radius \* radius

# 

# Developing the Input Process Output (IPO) Chart

# Extends and organizes the information in the Problem Analysis Chart.

# It shows in more detail what data items are inputs, what is the processing or modules on that data, and what will be the result or output.

# It combines information from PAC and HIPO Chart.

# Example # 03:

# Write a Input Process Output (IPO) to find an area of a circle where

# area = pi \* radius \* radius

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Processing | Module | Output |
| - radius | - Enter radius- area = 3.14 x radius x radius- Display area- end | 1000200030000000 | - Area of a circle |

# Drawing the Program Flowcharts

# Flowchart is the graphic representations of the individual steps or actions to implement a particular module.

# The flowchart can be likened to the blueprint of a building. An architect draws a blueprint before beginning construction on a building, so the programmer draws a flowchart before writing a program.

# Flowchart is independent of any programming language.

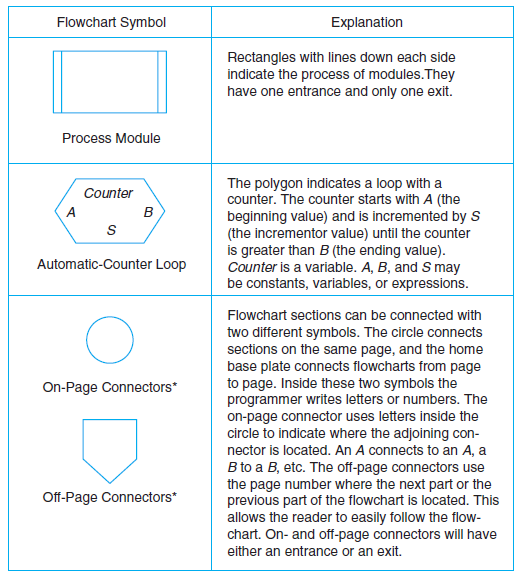
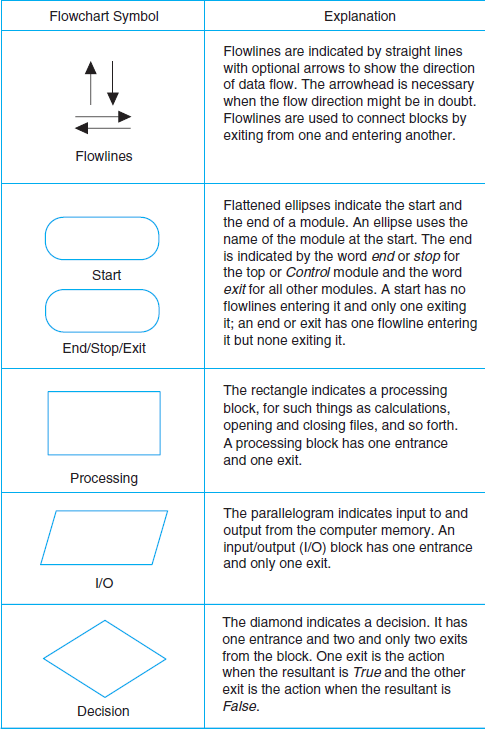
# Flowchart is the logical design of a program.

# It is the basis from which the actual program code is developed.

# Flowchart serves as documentation for computer program.

# The flowchart must be drawn according to definite rules and utilizes standard symbols adopted internationally.

# The International Organization for Standardization (IOS) was the symbols shown below (You can draw the symbols using ready-made flowcharting template):



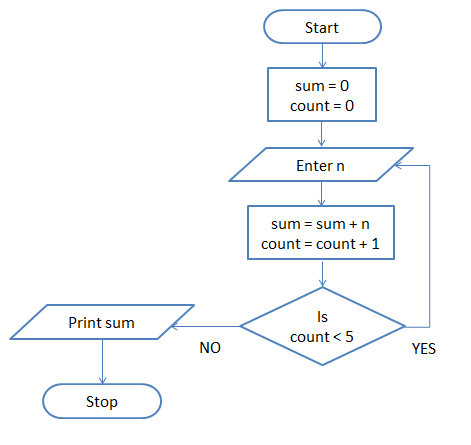
# Example # 04:

## Find the sum of 5 numbers

### Algorithm (in simple English)

1. Initialize sum = 0 and count = 0    (PROCESS)
2. Enter n    (I/O)
3. Find sum + n and assign it to sum and then increment count by 1 (PROCESS)
4. Is count < 5 (DECISION)
5. if YES go to step 2  
   else  
   Print sum (I/O)

### Flowchart



# Writing the Algorithm (Pseudo code)

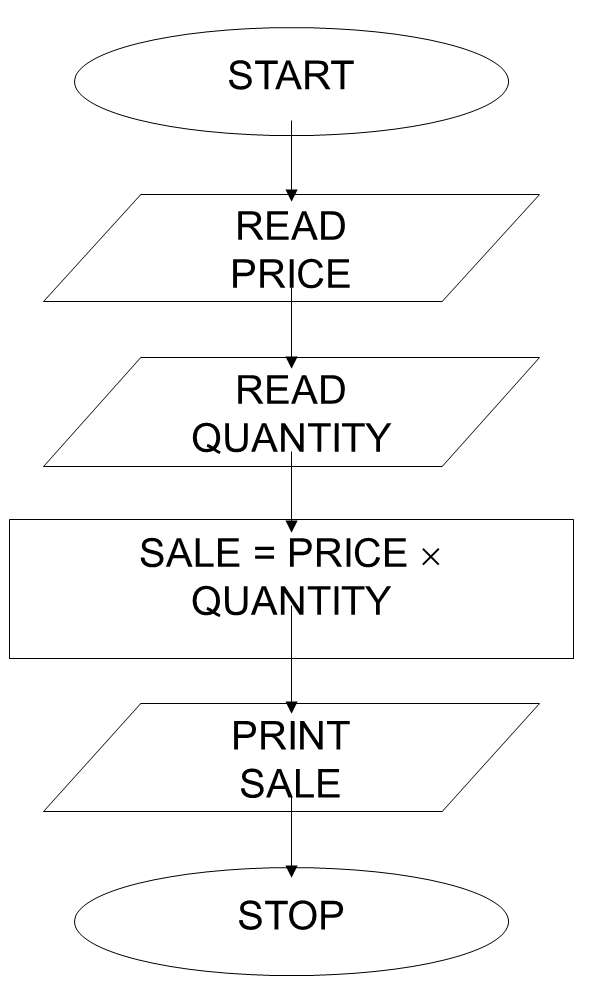
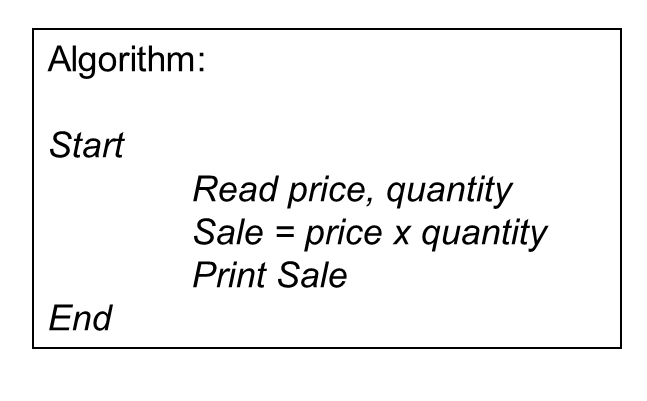
# Pseudo code means an imitation computer code.

# It is used in place of symbols or a flowchart to describe the logic of a program. Thus, it is a set of instructions (descriptive form) to describe the logic of a program.

# Pseudo code is close to the actual programming language.

# Using the Pseudo code, the programmer can start to write the actual code.

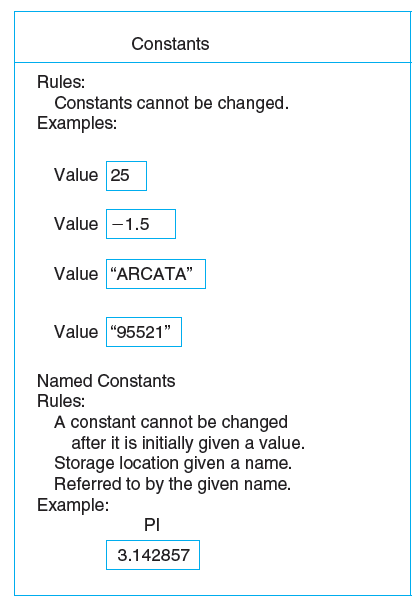
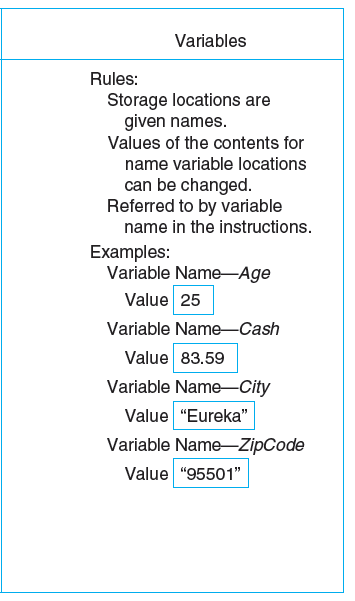
## Example # 12:

**** 

# Important Concepts to Learn

# 

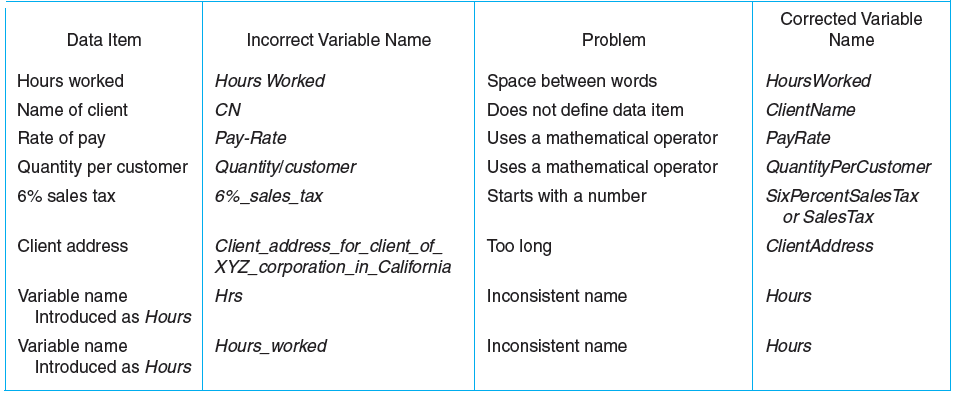
# Constants and Variables on the Computer

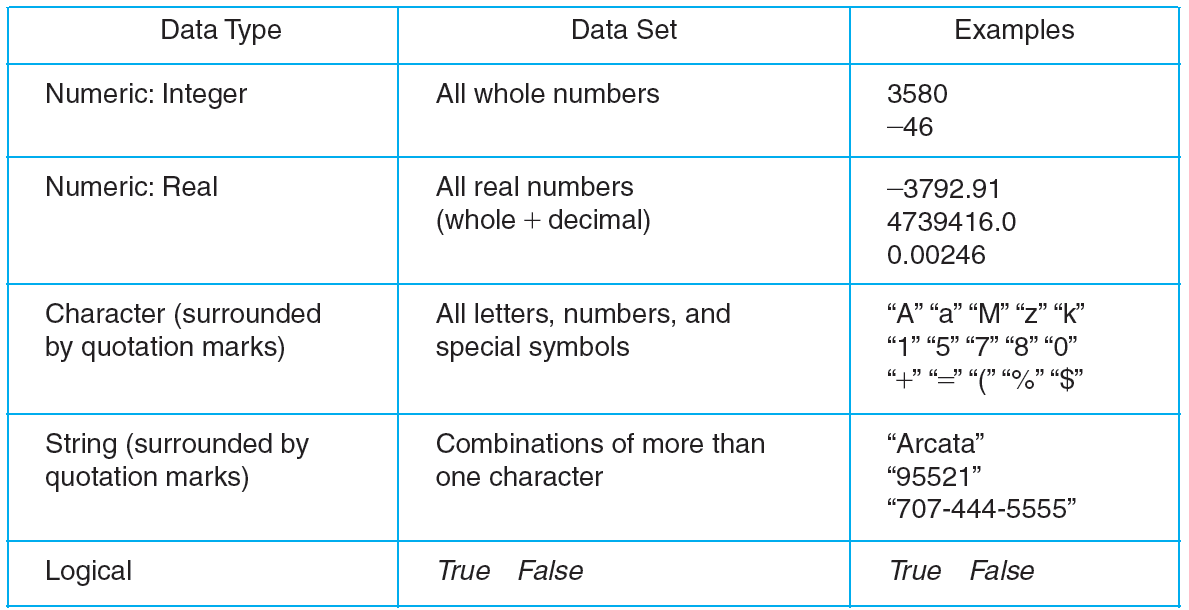
**Rules for Naming and Using Variables**

1. Name a variable according to what it represents.
2. Do not use spaces.
3. Start a variable name with a letter.
4. Do not use a dash or any other symbol that is used as a mathematical operator.
5. Consistent usage of variable name.
6. Consistent use of upper, lowercase characters in variable names
7. Use naming convention specified by your company

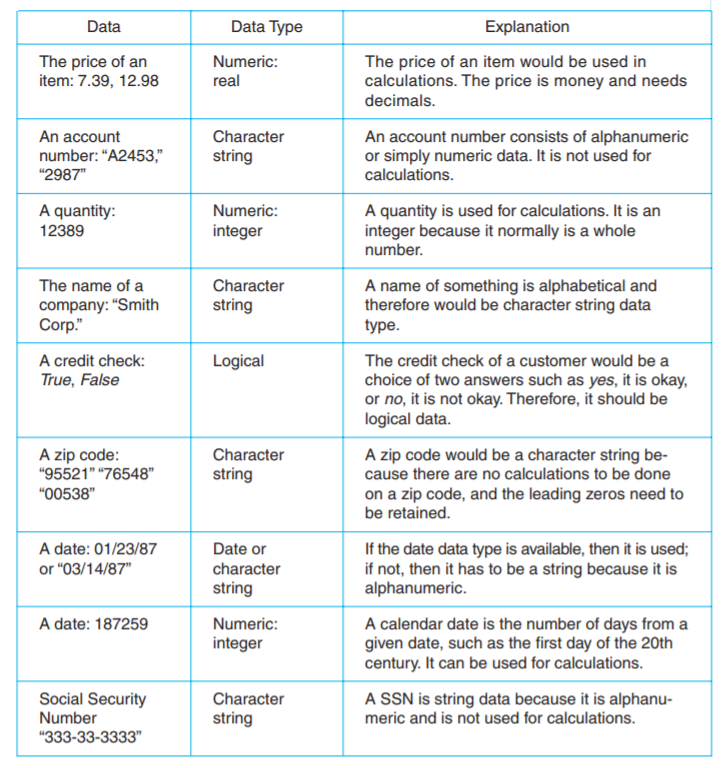
**Incorrect Variable Names**



**Data Types and Their Data Sets**



**Examples of Data Types**

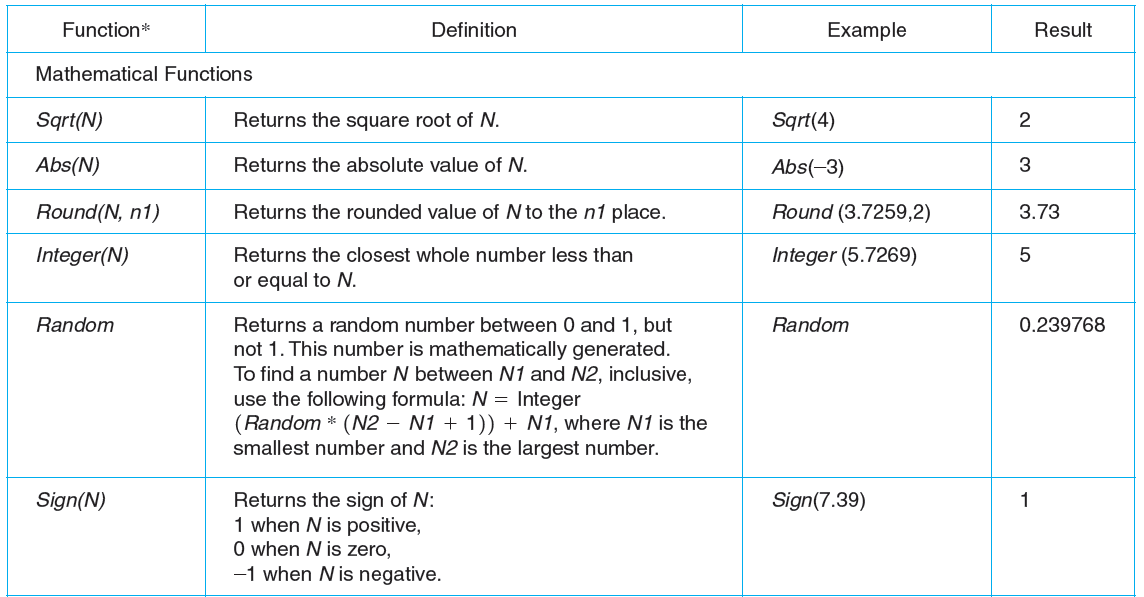


**Functions**

Functions have been divided into classes.

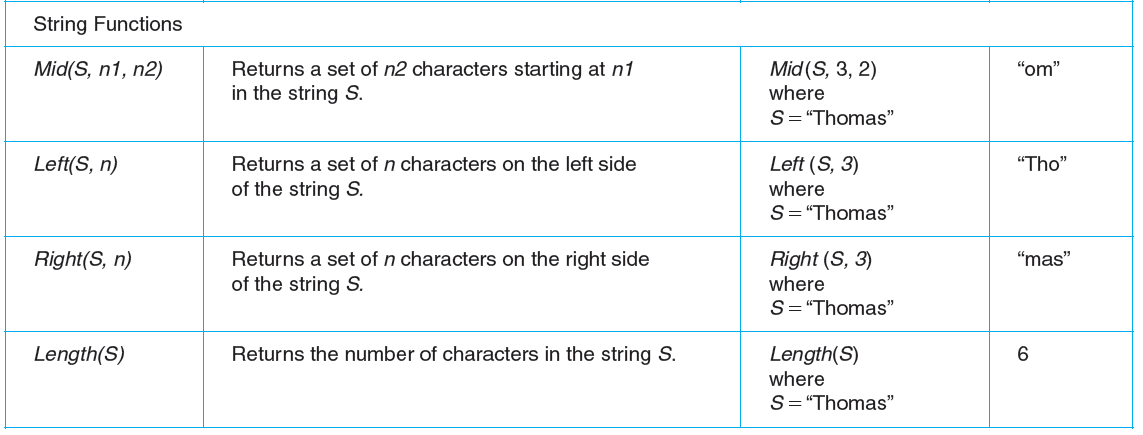
1. **Mathematical functions.**

Often used in science and business, mathematical functions calculate such things as square root, absolute value, or a random number. Other mathematical functions used primarily for scientific purposes have not been included in the table.



1. **String functions.**

These are used to manipulate string variables. For example, they copy part of the string into another variable, find the length or the number of characters in the string, and so forth.

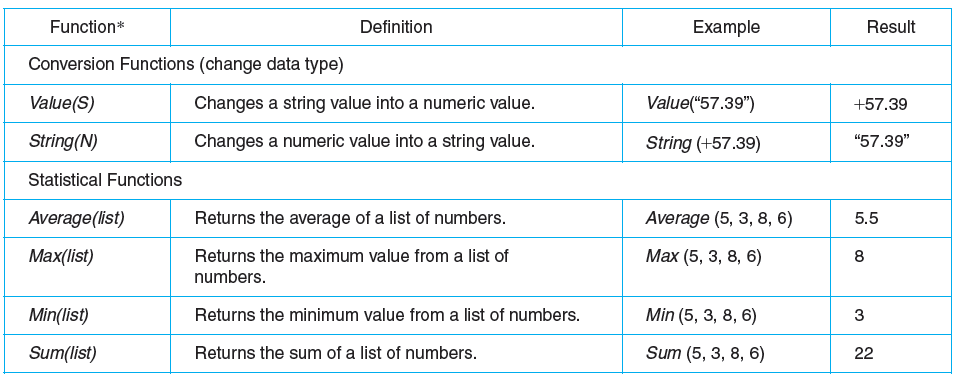


1. **Conversion Function.**

These functions are used to convert data from one data type to another. For example, since character strings cannot be used in calculations, one of these functions would convert a string value to a numeric value.

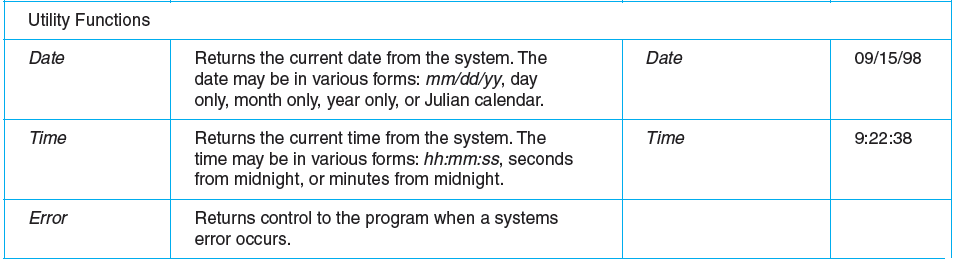
1. **Statistical functions.**

These functions are used to calculate things such as maximum values, minimum values, and so forth.



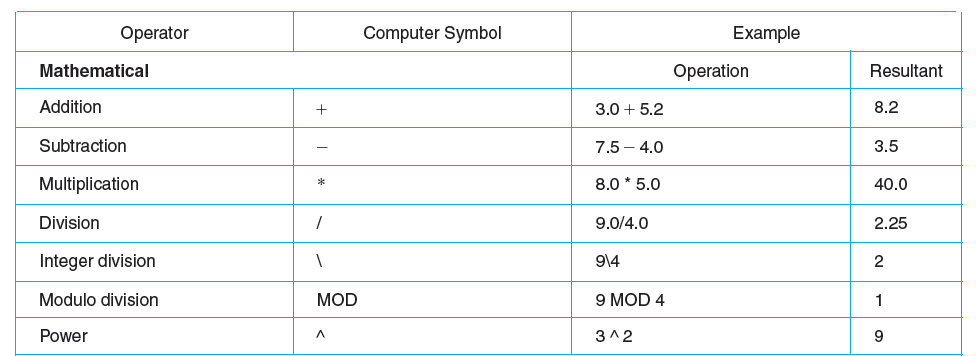
1. **Utility Function.**

This class is very important in business programming because most reports require some use of utility functions. They access information outside the program and the language in the computer system. Examples of these include date and time functions.

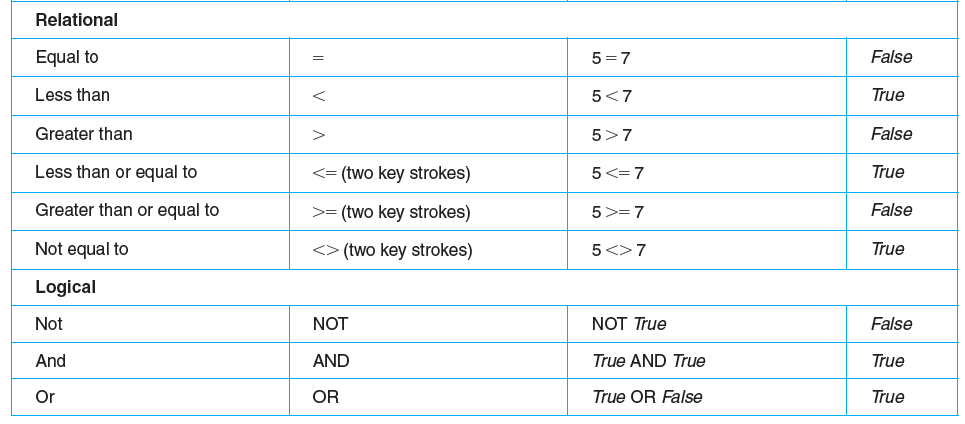


**Operators and Their Computer Symbols**

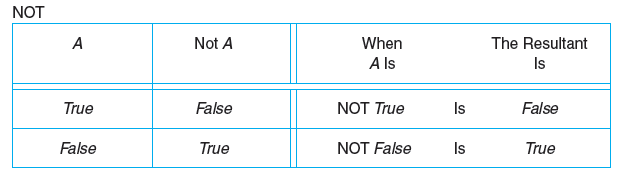
**Mathematical Operators:**

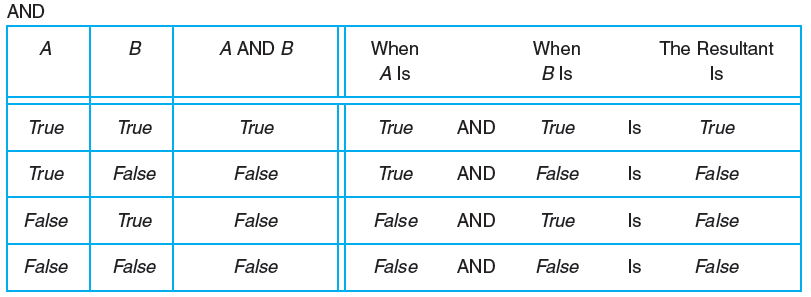


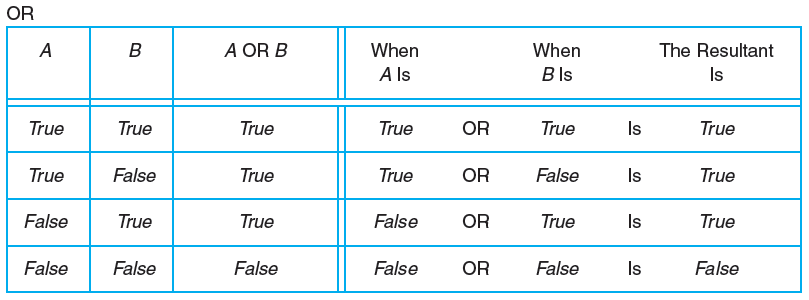
**Relational Operators:**



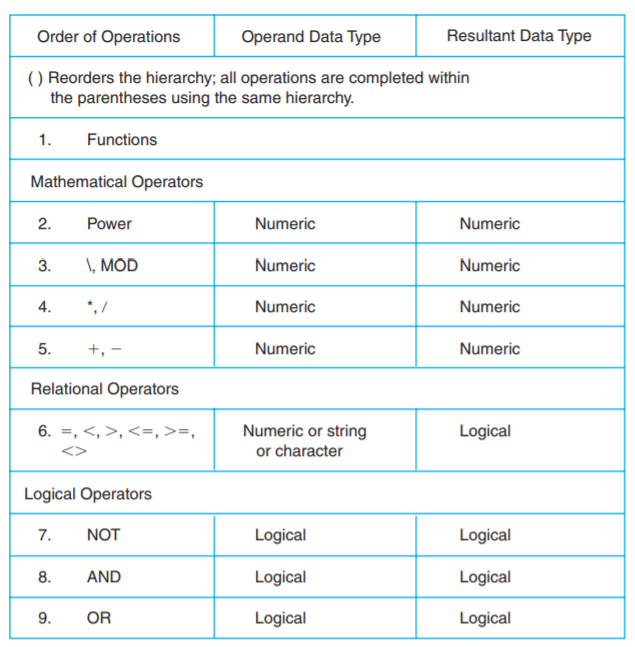
**Logical Operators:**



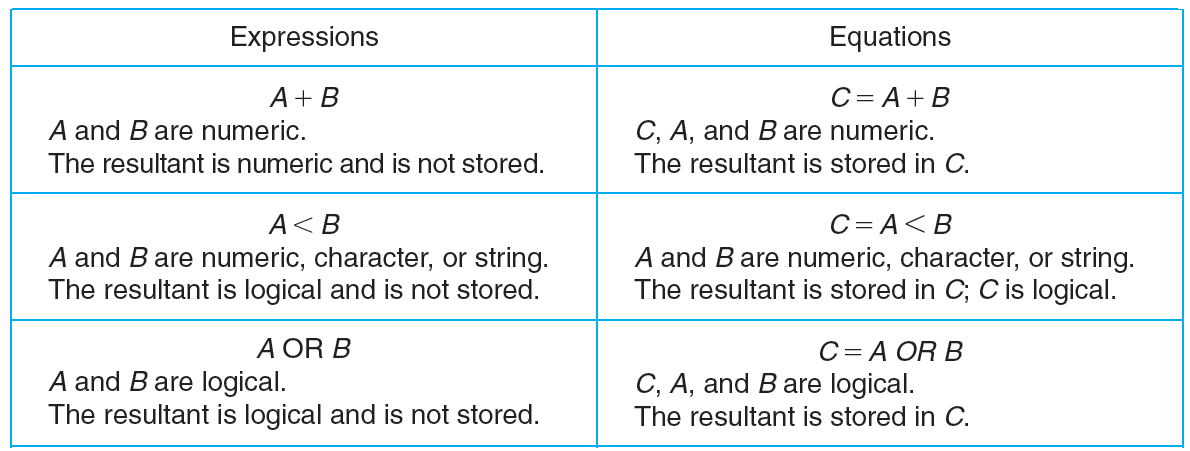




**Hierarchy of Operations**



**Expressions and Equations**



**Evaluating a Mathematical Expression**

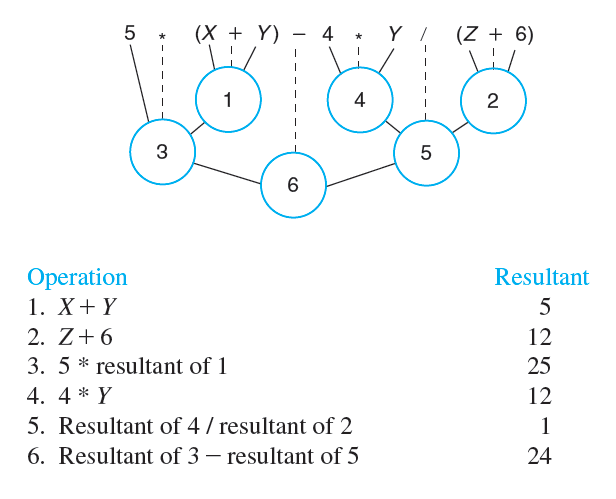
**Example:**

To find out if proposed solutions are correct, it is important for the programmer to evaluate, or test, all expressions and equations. Assume the programmer has written the expression

**5 \* (X + Y) - 4 \* Y/(Z + 6)**

The programmer uses the following values to evaluate the expression:

**X = 2, Y = 3, Z = 6**



**Example:**

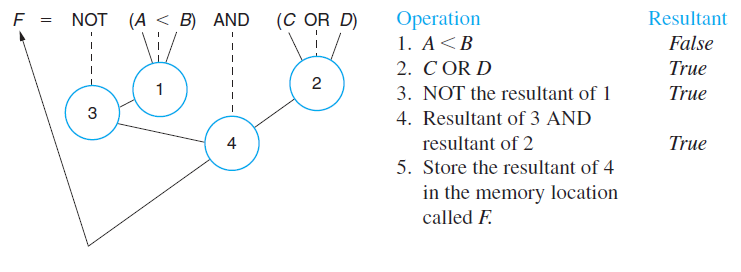
Assume the programmer has written the following equation:

**F = NOT(A < B) AND (C OR D)**

The programmer uses the following values to evaluate the equation:

**A = 4, B = 2, C = True, D = False**

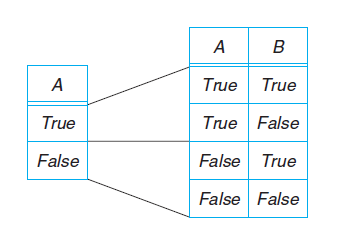
The operations are completed in hierarchical order from left to right, as illustrated in the



**Developing a Table of All Possible Resultants of a Logical Expression**

**Example:**

* Two unknowns—*A* and *B.*
* Four combinations: *B* can be either *True* or *False* for each value of *A..*

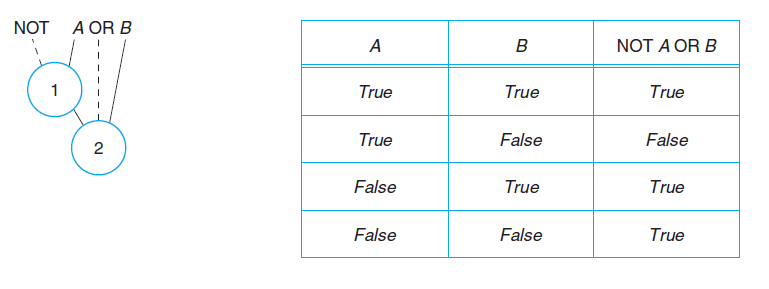


**Developing a Table of All Possible Resultants of a Logical Expression**

**Example:**

**Given the expression**

**NOT A OR B**



**Decision:**

Following are the main types of decision statements.

|  |  |  |
| --- | --- | --- |
| **Statement** | **Description** | **C - Syntax** |
| [if statement](http://www.tutorialspoint.com/cplusplus/cpp_if_statement.htm) | An if statement consists of a boolean expression followed by one or more statements. | if (expression)  {  Statement;  } |
| [if...elsestatement](http://www.tutorialspoint.com/cplusplus/cpp_if_else_statement.htm) | An if statement can be followed by an optional else statement, which executes when the boolean expression is false. | if(expression)  {  statement1;  }  else  {  statement2;  } |
| [if](http://www.tutorialspoint.com/cplusplus/cpp_nested_switch.htm)-else-if | This construct is useful where two or more alternatives are available for selection. | If (condition)  {  statement 1;  }  else if (condition)  {  statement 2;  }  else  {  statements n ;  } |
| [nested ifstatements](http://www.tutorialspoint.com/cplusplus/cpp_nested_if.htm) | An entire *if...else* construct is written within either the body of the if statement or the body of an else statement. | if(condition\_1)  {  If (condition\_2)  {  block statement\_1;  }  else  {  block statement\_2;  }  }  else  {  block statement\_3;  }  block statement\_4; |