Lecture 2 – System / Application Software – Algorithms and Anaylsis

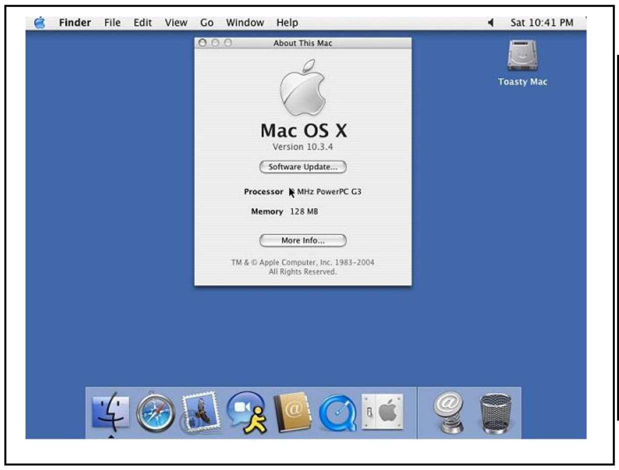
Software

Software is another name for programs. The programs that provide the instructions for a computer to execute (boot program, assembler, compiler, operating system, applications, function library, browser, etc.)

Two major kinds of software

System Software : A collection of programs – not a single program .

- Includes Operating System software, Utilities, and Device Drivers
- Enables the application software to interact with the hardware, and helps the computer manage its resources
- Two best-known operating systems for microcomputers are Windows 10 and Mac OS X

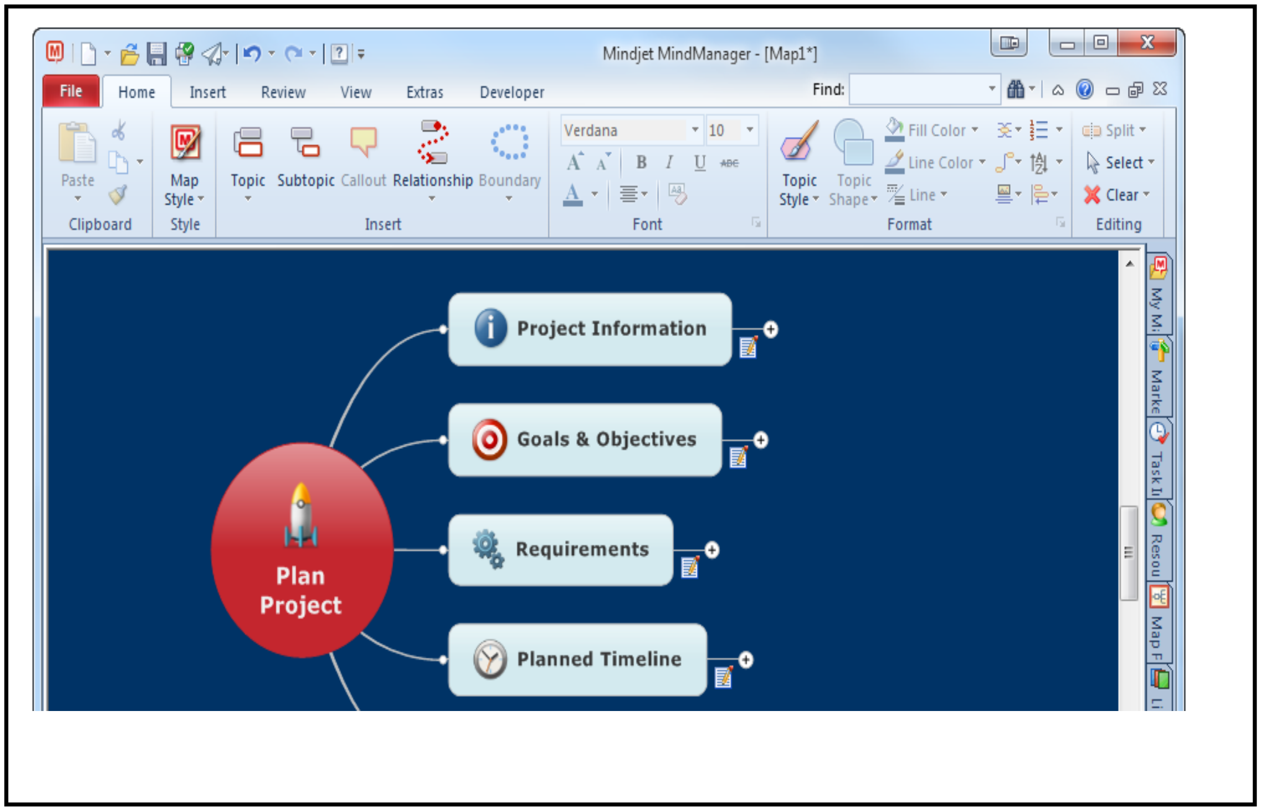


Application Software

End-user software : Software written to address specific needs—to solve problems in the real world such as : Word processing programs, games, inventory control systems, automobile diagnostic programs, and missile guidance programs are all application software

Two major categories

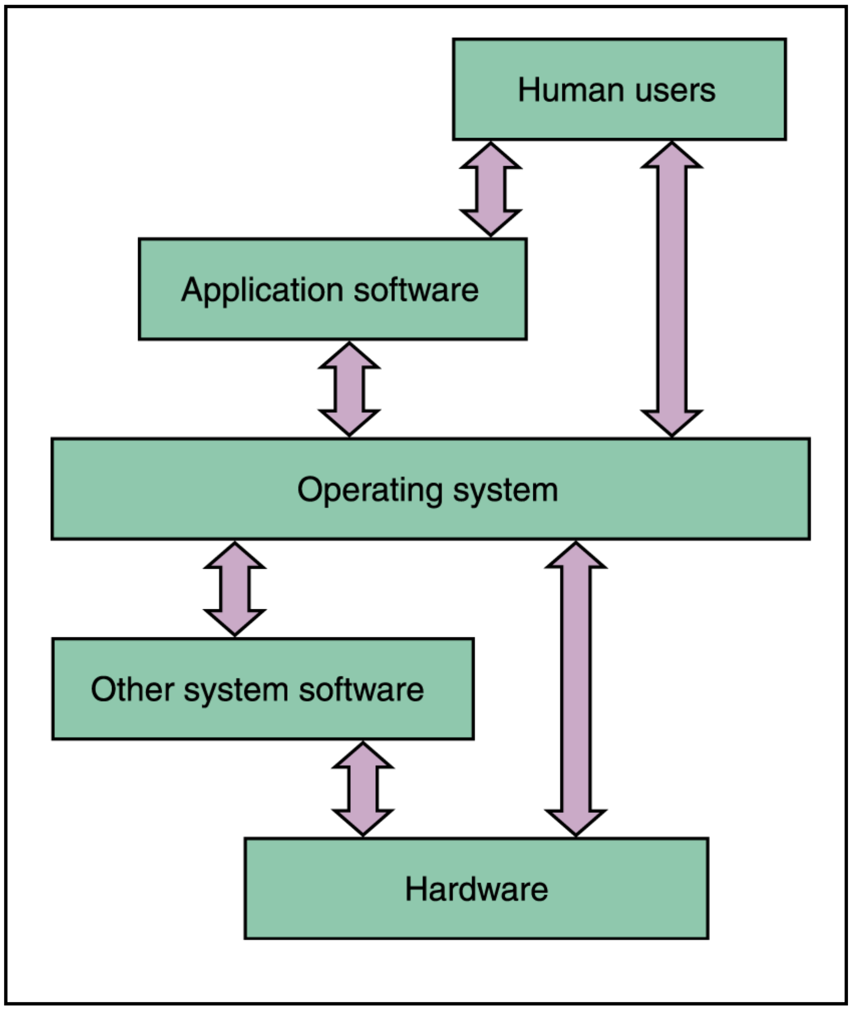
- **Basic applications**
- **Specialized applications**



An operating system

- Manages computer resources, such as memory and input/output devices
- Provides an interface through which a human can interact with the computer
- Allows an application program to interact with these other system resources

An operating system interacts with many aspects of a computer system.



- The various roles of an operating system generally revolve around the idea of "sharing nicely"
- An operating system manages resources, and these resources are often shared in one way or another among programs that want to use them

- **Multiprogramming** The technique of keeping multiple programs in main memory at the same time that compete for access to the CPU so that they can execute

- **Memory management** The process of keeping track of what programs are in memory and where in memory they reside

Resource Management

- **Process** : A program in execution
- The operating system performs **process management** to carefully track the progress of a process and all of its intermediate states
- **CPU scheduling :** determines which process in memory is executed by the CPU at any given point

Timesharing

- **Timesharing system** : A system that allows **multiple users** to interact with a computer at the same time
- **Multiprogramming :** A technique that allows **multiple processes** to be active at once, allowing programmers to interact with the computer system directly, while still sharing its resources
- In a timesharing system, each user has his or her own **virtual machine**, in which all system resources are (in effect) available for use

Other Factors

- **Real-time System :** A system in which response time is crucial given the nature of the application
- **Response time :** The time delay between receiving a stimulus and producing a response

Memory Management

- Operating systems must employ techniques to
  - Track where and how a program resides in memory
  - Convert **logical addresses** into actual **addresses**

CPU Scheduling

- **CPU Scheduling** The act of determining which process in the *ready* state should be moved to the *running* state
  - Many processes may be in the ready state
  - Only **one** process can be in the running state, making progress at any one time
- *Which one gets to move from ready to running?*

- **Nonpreemptive scheduling** The currently executing process gives up the CPU voluntarily

- **Preemptive scheduling** The operating system decides to favor another process, preempting the currently executing process

- **Turnaround time** The amount of time between when a process arrives in the ready state the first time and when it exits the running state for the last time

CPU Scheduling Algorithms

First-Come, First-Served

- Processes are moved to the CPU in the order in which they arrive in the running state

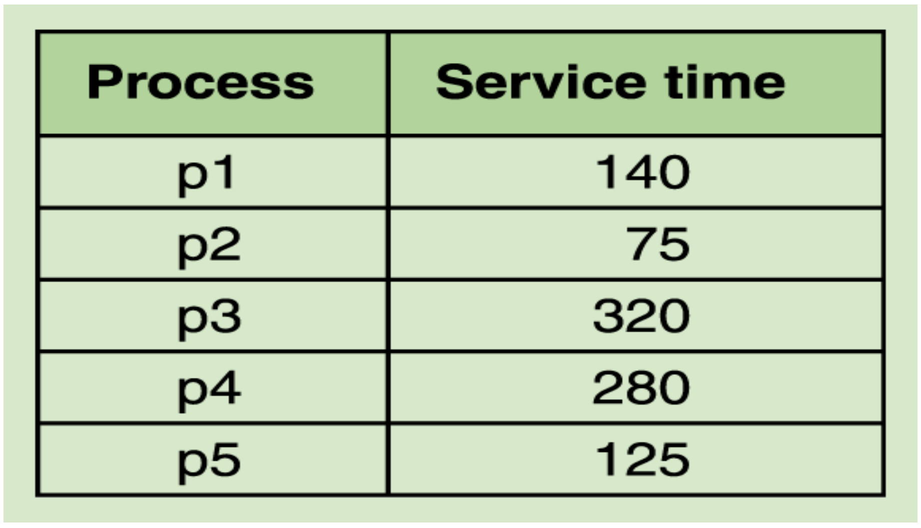
Shortest Job Next

- Process with shortest estimated running time in the ready state is moved into the running state first

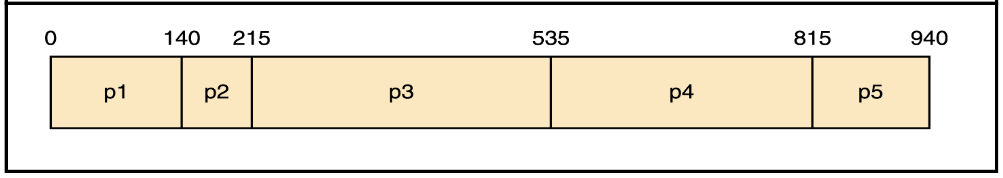
Round Robin

- Each process runs for a specified time slice and moves from the running state to the ready state to await its next turn if not finished

First-Come, First-Served

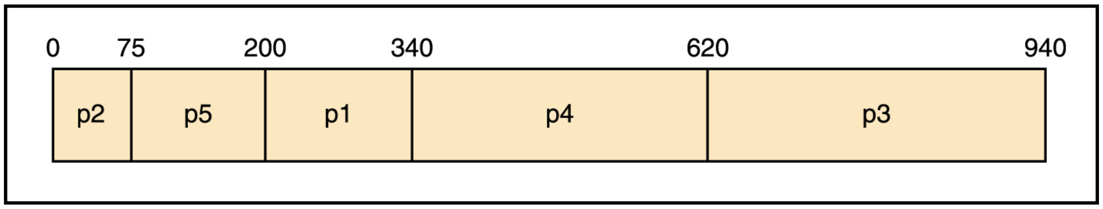


Process	Service time
p1	140
p2	75
p3	320
p4	280
p5	125



Shortest Job Next

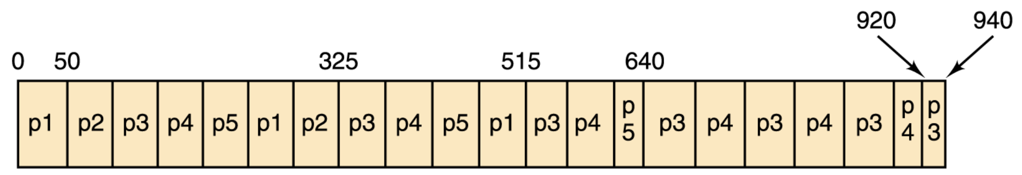
- Looks at all processes in the ready state and dispatches the one with the smallest service time



Round Robin

- Distributes the processing time equitably among all ready processes
- The algorithm establishes a particular **time slice** (or time quantum), which is the amount of time each process receives before being preempted and returned to the ready state to allow another process its turn

Suppose the time slice was 50



Are they preemptive or non-preemptive? Explain

- *First-Come, First-Served?*
- *Shortest Job Next?*
- *Round Robin?*

File Systems

- **File** A named collection of related data
- **File system** The logical view that an operating system provides so that users can manage information as a collection of files
- **Directory** A named group of files

Text and Binary Files

- **Text file** A file in which the bytes of data are organized as characters from the ASCII or Unicode character sets
- **Binary file** A file that contains data in a specific format, requiring interpretation

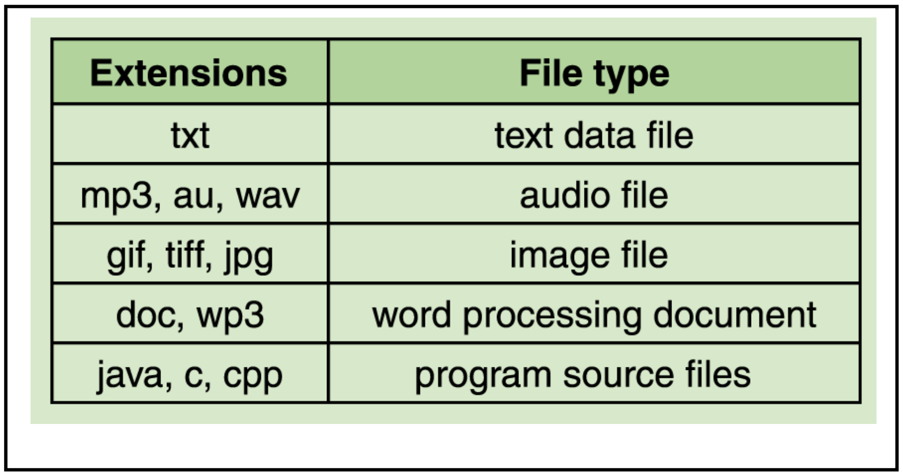
File Types

- Most files, whether they are in text or binary format, contain a specific type of information
  - For example, a file may contain a Java program, a JPEG image, or an MP3 audio clip

- The kind of information contained in a document is called the **file type**. Most operating systems recognize a list of specific file types

- File names are often separated, usually by a period, into two parts
  – Main name
  – File extension

- The **file extension** indicates the type of the file



Extensions	File type
txt	text data file
mp3, au, wav	audio file
gif, tiff, jpg	image file
doc, wp3	word processing document
java, c, cpp	program source files

File Operations

- Create a file
- Delete a file
- Open a file
- Close a file
- Read data from a file
- Write data to a file
- Reposition the current file pointer in a file
- Append data to the end of a file
- Truncate a file (delete its contents)
- Rename a file
- Copy a file

File Access

- **Sequential access** Information in the file is processed in order, and read and write operations move the current file pointer as far as needed to read or write the data. The most common file access technique, and the simplest to implement
- **Direct access** Files are conceptually divided into numbered logical records and each logical record can be accessed directly by number

File Protection

- A file's protection settings in the Unix operating system is divided into three categories
    - Owner
    - Group
    - World

	Read	Read / Write	Execute
Owner	Yes	Yes	No
Group	Yes	No	No
World	No	No	No

Ethical Issues

1. If a flow of Operating system’s security allows a malicious programmer to gain unauthorized access to sensitive data , to what extend the developer of the operating system be held responsible.

2. Is it your responsibility to lock your house in such a way that intruders cannot get in , or is it the public’s responsibility to stay out your house unless invited. Is it the responsibility of an operating system to guard access to a computer and its contents   , or is it the responsibility of   hackers to leave the machine alone.

Application Software

Application software are programs that direct the performance of a particular use, or application, of computers to meet the information processing needs of end users. They include A off-the-shelf software such as word processing and spreadsheet packages, as well as internally or externally developed software that is designed to meet the specific needs of an organization.

Software trends have been away from custom-designed one-of-kind programs developed by the professional programmers or end users of an organization toward the use of A off-the-shelf software packages acquired by end users from software vendors.

Spreadsheets

Electronic spreadsheet packages are programs that are used for analysis, planning, and modelling. They provide electronic replacement for more traditional tools such as paper worksheets, pencils, and calculators. In a worksheet of rows and columns are stored in the computer's memory and displayed on the video screen. Data and formulas are entered into the worksheet and the computer performs the calculations based on the formulas entered. A spreadsheet package can also be used as a decision support tool to perform what-if analysis.

Database Management

Database management packages facilitate the storage, maintenance, and utilization of data in a database that is shared by many users. Microcomputer DBMs enables the users to:

1. Create and maintain a database

2. Query a database with a query language

3. Prepare formatted reports

In addition, packages offer security features, network connectivity, and the ability to present graphical output, as well as to perform spreadsheet-type computations.

Word Processing

Word processing packages are programs that computerize the creation, edition, and printing of documents by electronically processing text data. Word processing is an important application of office automation. Word processing is the most popular authoring and presentation activity. In fact, it is the most common personal computing application.

Desktop Publishing

Organizations use desktop publishing software to produce their own printed materials like newsletters, brochures, manuals, and books with several type styles, graphics, and colors on each page. The components required to set up a modest desktop publishing system include: a high-resolution display, a laser printer, desktop publishing software, and perhaps a scanner.

Presentation Software

The goal of presentation graphics is to provide information in a graphical form that helps end users and mangers understand business proposals and performance and make better decisions about them. It is important to note that presentation graphics does not replace reports and displays of numbers and text material.

Personal Information Management

Personal Information management (PIM) packages are tools that help knowledge workers track tasks, people, projects, commitments, and ideas. These packages help end users store, organize, and retrieve text and numerical data in the form of notes, lists, clippings, tables, memos, letters, reports, and so on.

Communications Software and Web Browser

Communications software enables the user to connect to a telecommunications network in order to exchange information with other users or systems. The software provides the following capabilities:

1. Sending and receiving electronic mail

2. File transfer. You can download a program or a data file from a remote computer to your own workstation or upload a file to the remote computer.

3. Terminal emulation - enabling the personal computer to act as a terminal when required in a particular application.

4. Sending and receiving a fax

More and more frequently, the reason for connecting to a telecommunications network is to gain access to the resources of the Internet. Web browsers are rapidly becoming one of the most popular categories of software packages. A ***browser*** is a program that enables its user to access electronic documents in included in the Internet's World Wide Web, a collection of interlinked hypermedia databases distributed among remote sites.

Programming Languages and their Translators

Much of the applications software used in an organization needs to be programmed or customized. Programming languages are the languages which computer programs are written int. A programming language allows a programmer or end user to develop the sets of instructions that constitute a computer program. These languages have evolved over four generations and can be grouped into five major categories:

1. Machine languages

2. Assembler languages

3. High-level languages

4. Fourth generation languages

5. Object-oriented languages

Machine Languages:

Machine languages are the most basic level of programming languages. They were the first generation of machine languages.

- Programs had to be written using binary codes unique to each computer.
- Programming was difficult and error-prone.
- Programs are not portable to other computers.

Assembler Languages:

They were developed to reduce the difficulties in writing machine language programs. Assembly language is also specific to a computer model or a series of models and the programs are not portable to other computers

Assembly languages are used today only when tight control over computer hardware resources is required, such as in certain systems programs, particularly those for real-time computing.

High-Level Languages

High-level languages are the third generation programming languages. These languages provide statements, each of which is translated into several machine-language instructions. High-level languages include **COBOL** (business application programs), **BASIC** (microcomputer end users), **FORTRAN** (scientific and engineering applications), and more popular today are **C, C++, Rust , and Visual Basic**.

Advantages:

1. Easier to learn and understand than an assembler language as instructions (statements) that resemble human language or the standard notation of mathematics.

2. Are machine-independent programs therefore programs written in a high-level language do not have to be reprogrammed when a new computer is installed.

3. Programmers do not have to learn a new language for each computer they program.

Disadvantages:

1. Less efficient than assembler language programs and require a greater amount of computer time for translation into machine instructions.

Beyond High-Level Programming Languages

The fourth-generation languages (4GLs) specify what needs to be done rather than detailing steps to doing it. 4GLs include a variety of programming languages that are more nonprocedural and conversational than prior languages.

Object-oriented programming (OOP) languages tie data elements and the procedures or actions that will be performed on them, together into objects. Examples include Smalltalk, C++, Visual Basic, Java, Turbo C++, C++, Object C+, and Python.

Languages that facilitate **parallel processing** in systems with a large number of processors.

Functional languages (such as LISP), based on the mathematical concept of computation as an application of functions.

Limited subsets of **natural languages** which can be processed thanks to the progress in artificial intelligence.

Advantages:

1. OOP languages are easier to use and more efficient for programming the graphics-oriented user interface required by many applications.

2. Programmed objects are reusable.

Translators: Compilers and Interpreters

A variety of software packages are available to help programmers develop computer programs. For example, programming language translators are programs that translate other programs into machine language instruction codes that computers can execute. Other software packages called programming tools help programmers write programs by providing program creation and editing facilities. Language translator programs (language processors) are programs that translate other programs into machine language instruction codes the computer can execute. These programs allow you to write your own programs by providing program creation and editing facilities.

Programming language translator programs are known by a variety of names.

Compiler: translates (compiles) high-level language statements (source programs) to machine language programs.

Interpreter: translates and executes each program statement one at a time, instead of first producing a complete machine language program, like compilers and assemblers do.

ALGORITHM

- A ***procedure*** for solving a problem in terms of
  - the ***actions*** to be executed, and
  - the ***order*** in which these actions are to be executed
- A **sequence of instructions** that one must perform in order to solve a well formulated problem in a finite amount of time using a finite amount of data. The instructions must be unambiguous.
- An **algorithm** is a procedure for accomplishing some tasks which :
  - Given an *initial state*
  - Terminate in a *defined end-state*

Problem Definition

- What is the task to be accomplished? For example , Calculate the average of the grades for a given student
- What are the time / space / speed / performance requirements ?

Problem Solving

- The act of **finding a solution** to a perplexing, distressing, vexing, or unsettled question is Problem solving
- G. Polya wrote *How to Solve It: A New Aspect of Mathematical Method*
- His How to Solve It list is quite general
    - Written in the context of solving mathematical problems
    - The list becomes applicable to all types of problems

Ask Questions...

In order to understand the problem ask :-

- *What do I know about the problem?*
- *What is the information that I have to process in order the find the solution?*
- *What does the solution look like?*
- *What sort of special cases exist?*
- *How will I recognize that I have found the solution?*

Look for Familiar Things

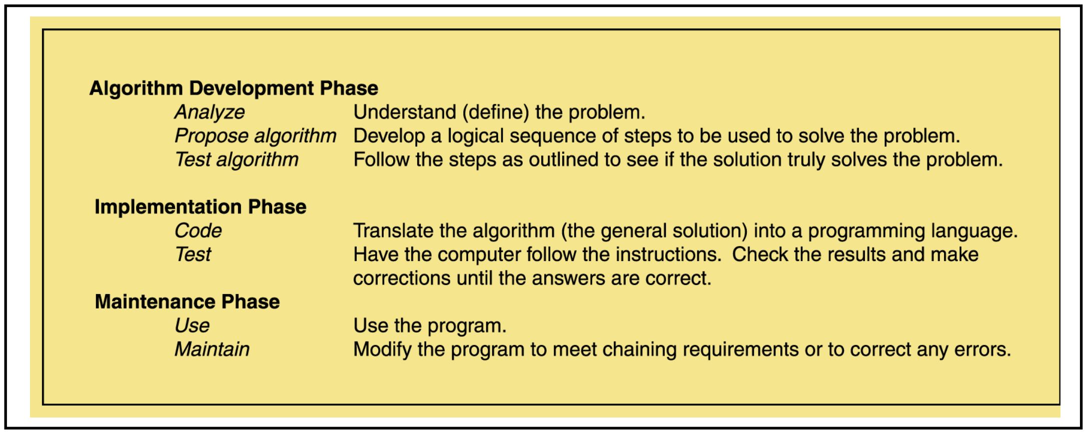
- You should never reinvent the wheel
- In computing, you see certain problems again and again in different guises
- A good programmer sees a task, or perhaps part of a task (a subtask),
  that has been solved before and plugs in the solution

Divide and Conquer

Break up a large problem into smaller units that we can handle

- Applies the concept of abstraction
- The divide-and-conquer approach can be applied over and over again until each subtask is manageable

Computer Problem-Solving Process



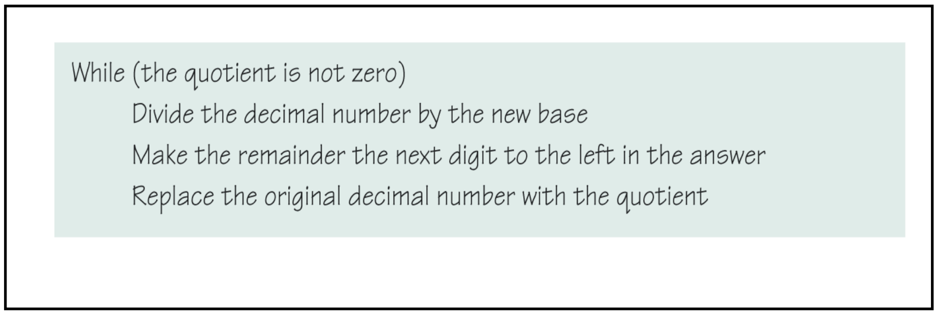
Algorithm Development Phase	
Analyze	Understand (define) the problem.
Propose algorithm	Develop a logical sequence of steps to be used to solve the problem.
Test algorithm	Follow the steps as outlined to see if the solution truly solves the problem.
Implementation Phase	
Code	Translate the algorithm (the general solution) into a programming language.
Test	Have the computer follow the instructions. Check the results and make corrections until the answers are correct.
Maintenance Phase	
Use	Use the program.
Maintain	Modify the program to meet chaining requirements or to correct any errors.

Pseudocode

- Uses a mixture of English and formatting to make the steps in the solution explicit
- Describe: in natural language / pseudo-code / diagrams / etc.

  Criteria to follow:

    - Input: Zero or more quantities (externally produced)
    - Output: One or more quantities
    - Equations … etc



```
While (the quotient is not zero)
    Divide the decimal number by the new base
    Make the remainder the next digit to the left in the answer
    Replace the original decimal number with the quotient
```

Developing an Algorithm

- The plan must be suitable in a suitable form
- Two methodologies that currently used
    - Top-down design
    - Object-oriented design

Implementation, Testing, Maintainance

- Implementation

  Decide on the programming language to use such C, C++, Lisp, Java, Perl, Prolog, assembly, etc. , etc.

  Write clean, well documented code

- Test, test, test

- Integrate feedback from users, fix bugs, ensure compatibility across different versions → Maintenance

Testing the Algorithm : How

- **Desk checking** Working through a design at a desk with a pencil and paper
- **Walk-through** Manual simulation of the design by the team members, taking sample data values and simulating the design using the sample data
- **Inspection** One person (not the designer) reads the design (handed out in advance) line by line while the others point out errors

Object-Oriented Design

- A problem-solving methodology that produces a solution to a problem in terms of self-contained entities called *objects*
- **Object** A thing or entity that makes sense within the context of the problem
    - For example, a student
- A group of similar objects is described by an **object class**, or **class**

Object-Oriented Design Methodology

- Four stages to the decomposition process
    - Brainstorming
    - Filtering
    - Scenarios

Functionality of most programming Languages

- **Sequence** Executing statements in sequence until an instruction is encountered that changes this sequencing
- **Selection** Deciding which action to take
- **Iteration** (looping) Repeating an action

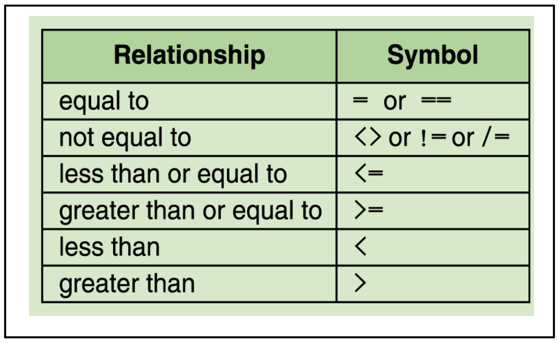
Both selection and iteration require the use of a **Boolean expression**

Boolean Expressions

- **Boolean expression** A sequence of identifiers, separated by compatible operators, that evaluates to true or false
- Boolean expression can be
  - A Boolean variable
  - An arithmetic expression followed by a relational operator followed by an arithmetic expression
  - A Boolean expression followed by a Boolean operator followed by a Boolean expression

- **Variable** A location in memory that is referenced by an identifier that contains a data value
  Thus, a Boolean variable is a location in memory that can contain either ***true*** or ***false***
- A relational operator between
  two arithmetic expressions is asking if the
  relationship exists between the two expressions

- For example,
  xValue < yValue



Relationship	Symbol
equal to	= or ==
not equal to	<> or != or /=
less than or equal to	<=
greater than or equal to	>=
less than	<
greater than	>

Strong Typing

- **Strong typing** The requirement that only a value of the proper type can be stored into a variable
- **Data type** A description of the set of values and the basic set of operations that can be applied to values of the type

Data Types

- Integer numbers
- Real numbers
- Characters
- Boolean values
- Strings

Integers

- The range varies depending upon how many bytes are assigned to represent an integer value
- Some high-level languages provide several integer types of different sizes
- Operations that can be applied to integers are the standard arithmetic and relational operations

Reals

- Like the integer data type, the range varies depending on the number of bytes assigned to represent a real number
- Many high-level languages have two sizes of real numbers
- The operations that can be applied to real numbers are the same as those that can be applied to integer numbers

Characters

- It takes one byte to represent characters in the ASCII character set
- Two bytes to represent characters in the Unicode character set
- Our English alphabet is represented in ASCII, which is a subset of Unicode
- Applying arithmetic operations to characters doesn't make much sense
- Comparing characters does make sense, so the relational operators can be applied to characters
- The meaning of "less than" and "greater than" when applied to characters is "comes before" and "comes after" in the character set

Boolean

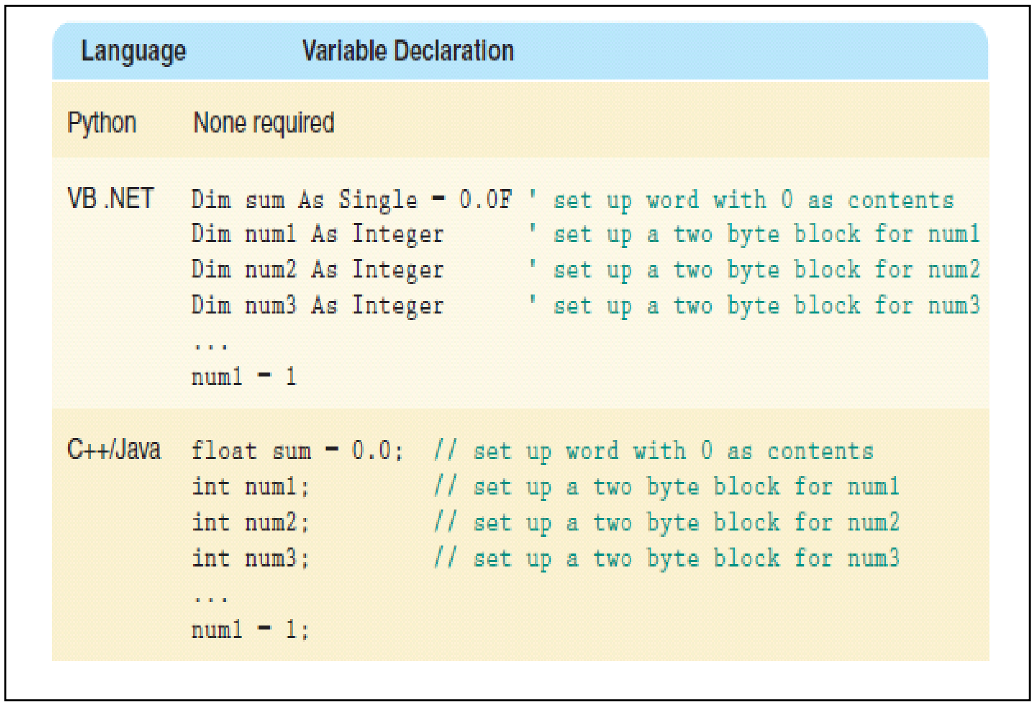
- The Boolean data type consists of two values: true and false
- Not all high-level languages support the Boolean data type
- If a language does not, then you can simulate Boolean values by saying that the Boolean value true is represented by 1 and false is represented by 0

Strings

- A string is a sequence of characters considered as one data value
- For example: “This is a string.”
  - Containing 17 characters: one uppercase letter, 12 lowercase letters, three blanks, and a period
- The operations defined on strings vary from language to language
  - They include concatenation of strings and comparison of strings in terms of lexicographic order

Declarations

- **Declaration** A statement that associates an identifier with a variable, an action, or some other entity within the language that can be given a name so that the programmer can refer to that item by name



Language	Variable Declaration
Python	None required

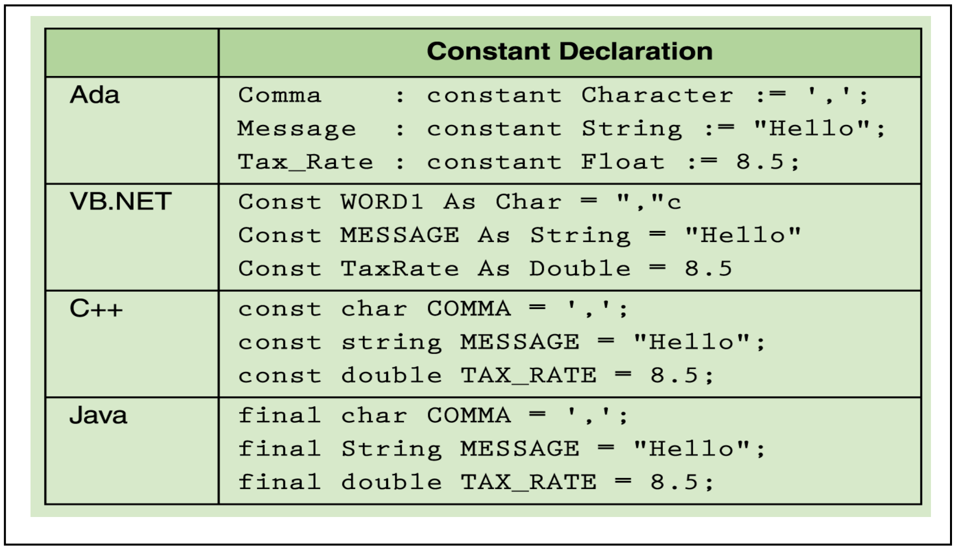
VB .NET	`Dim sum As Single = 0.0F ' set up word with 0 as contents`
	`Dim num1 As Integer      ' set up a two byte block for num1`
	`Dim num2 As Integer      ' set up a two byte block for num2`
	`Dim num3 As Integer      ' set up a two byte block for num3`
	`...`
	`num1 = 1`

C++/Java	`float sum = 0.0;  // set up word with 0 as contents`
	`int num1;         // set up a two byte block for num1`
	`int num2;         // set up a two byte block for num2`
	`int num3;         // set up a two byte block for num3`
	`...`
	`num1 = 1;`

- **Reserved word** A word in a language that has special meaning
- **Case-sensitive** Uppercase and lowercase letters are considered the same

Assignment statement

- **Assignment statement** An action statement (not a declaration) that says to evaluate the expression on the right-hand side of the symbol and store that value into the place named on the left-hand side
- **Named constant** A location in memory, referenced by an identifier, that contains a data value that cannot be changed



	Constant Declaration

Ada	`Comma    : constant Character := ',';`
	`Message  : constant String := "Hello";`
	`Tax_Rate : constant Float := 8.5;`

VB.NET	`Const WORD1 As Char = ","c`
	`Const MESSAGE As String = "Hello"`
	`Const TaxRate As Double = 8.5`

C++	`const char COMMA = ',';`
	`const string MESSAGE = "Hello";`
	`const double TAX_RATE = 8.5;`

Java	`final char COMMA = ',';`
	`final String MESSAGE = "Hello";`
	`final double TAX_RATE = 8.5;`

Input/Output Structures

Pseudocode algorithms used the expressions *Read or Get* and *Write or Print*

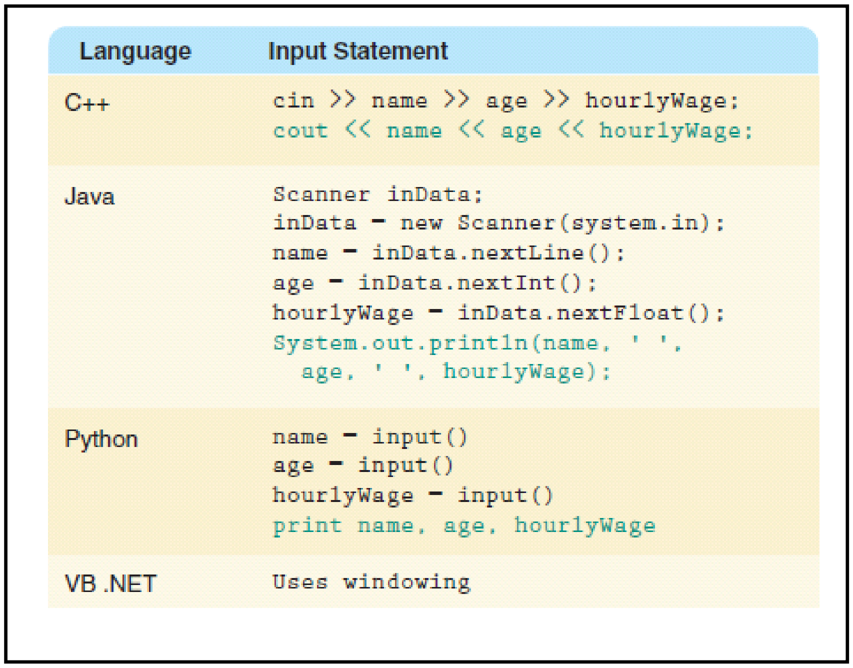
```
Read name, age, hourlyWage
```

name is a string;

age is an integer;

hourlyWage is a real

The data must be a string, an integer, and a real in that order.



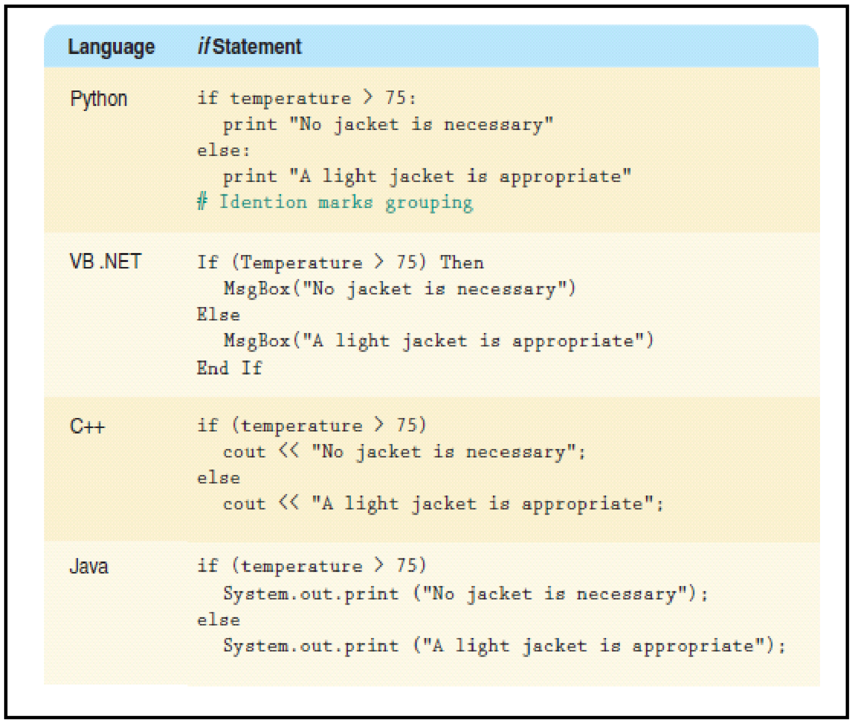
Language	Input Statement
C++	`cin >> name >> age >> hourlyWage;`
	`cout << name << age << hourlyWage;`
Java	`Scanner inData;`
	`inData = new Scanner(system.in);`
	`name = inData.nextLine();`
	`age = inData.nextInt();`
	`hourlyWage = inData.nextFloat();`
	`System.out.println(name, ' ',`
	`age, ' ', hourlyWage);`
Python	`name = input()`
	`age = input()`
	`hourlyWage = input()`
	`print name, age, hourlyWage`
VB .NET	`Uses windowing`

Control Structures

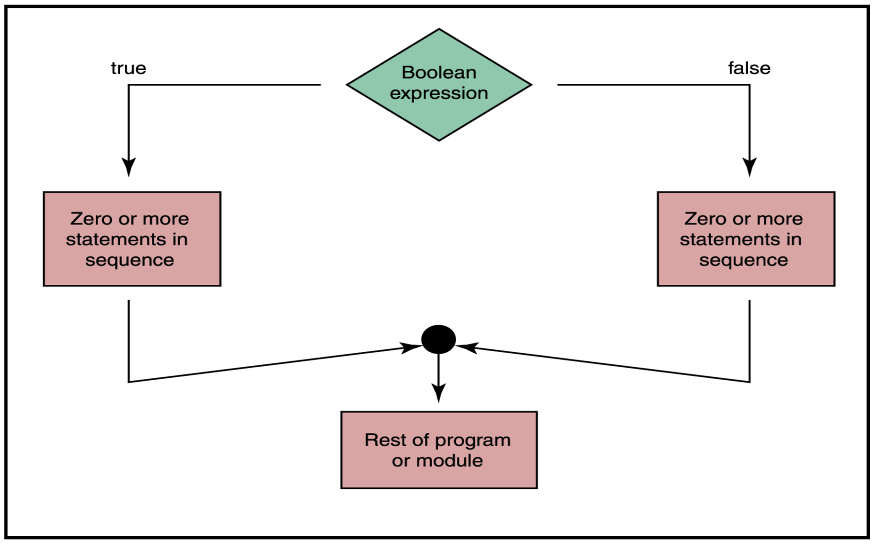
- Control structure An instruction that determines the order in which other instructions in a program are executed
- Structured programming A programming methodology in which each logical unit of a program should have just one entry and one exit
- Sequence, selection statements, looping statements, and subprogram statements are control structures

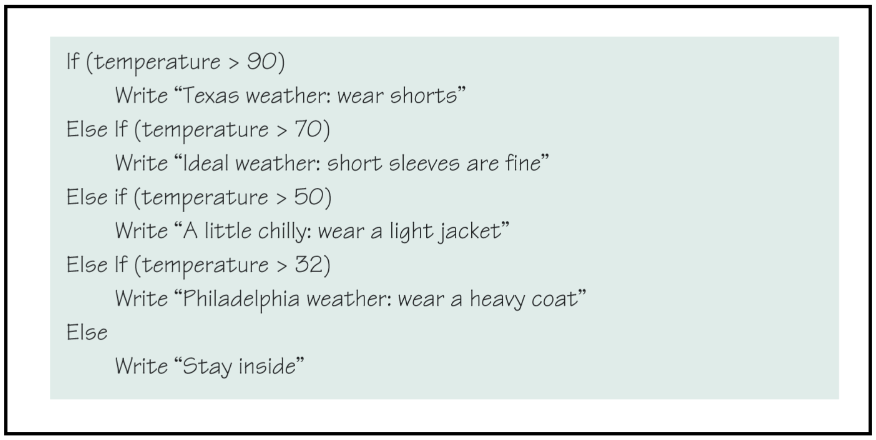
Selection Statements

- The ***if*** statement allows the program to test the state of the program variables using a Boolean expression



Language	*if* Statement
Python	`if temperature > 75:`
	`   print "No jacket is necessary"`
	`else:`
	`   print "A light jacket is appropriate"`
	`# Idention marks grouping`
VB .NET	`If (Temperature > 75) Then`
	`   MsgBox("No jacket is necessary")`
	`Else`
	`   MsgBox("A light jacket is appropriate")`
	`End If`
C++	`if (temperature > 75)`
	`   cout << "No jacket is necessary";`
	`else`
	`   cout << "A light jacket is appropriate";`
Java	`if (temperature > 75)`
	`   System.out.print ("No jacket is necessary");`
	`else`
	`   System.out.print ("A light jacket is appropriate");`

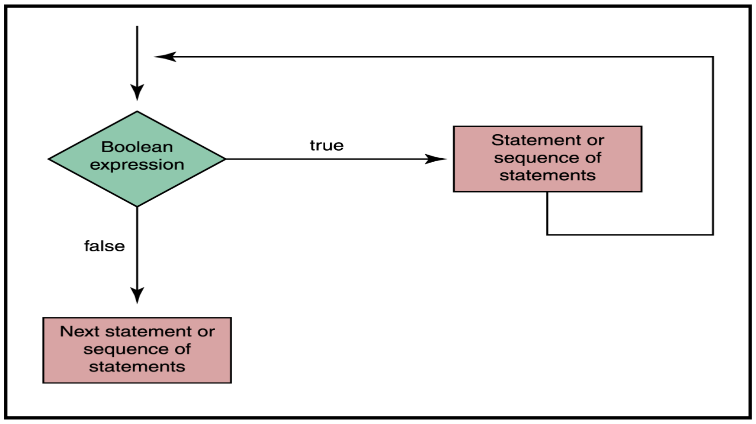

true
Boolean expression
false
Zero or more statements in sequence
Zero or more statements in sequence
Rest of program or module



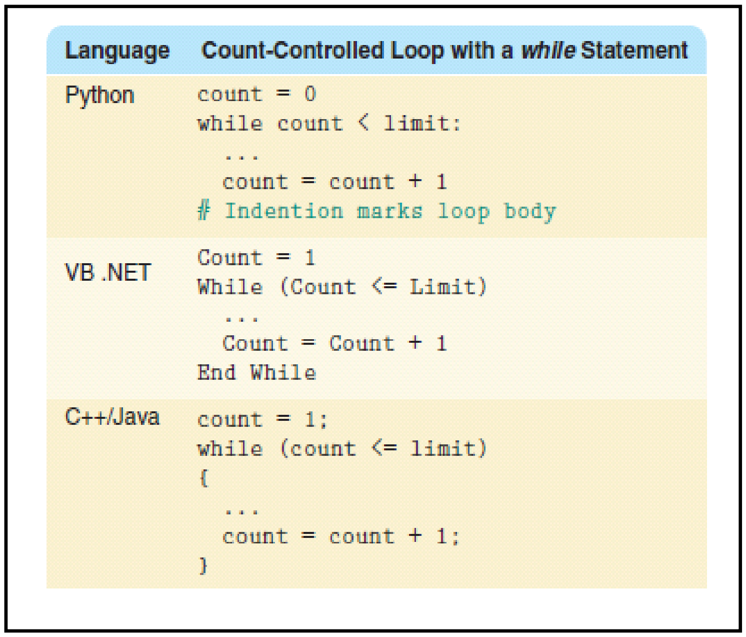
```
If (temperature > 90)
      Write "Texas weather: wear shorts"
Else If (temperature > 70)
      Write "Ideal weather: short sleeves are fine"
Else if (temperature > 50)
      Write "A little chilly: wear a light jacket"
Else If (temperature > 32)
      Write "Philadelphia weather: wear a heavy coat"
Else
      Write "Stay inside"
```

Looping Statements

- The ***while*** statement is used to repeat a course of action
- Let's look at two distinct types of repetitions
- Count-controlled loops
  - Repeat a specified number of times
  - Use of a special variable called a loop control variable



Boolean expression
true
Statement or sequence of statements
false
Next statement or sequence of statements



Language	Count-Controlled Loop with a *while* Statement

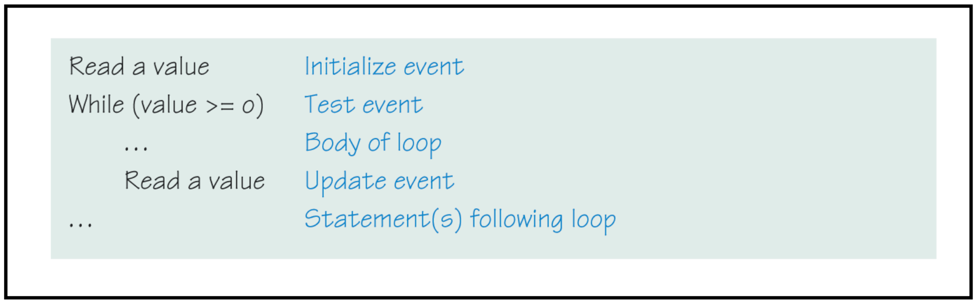
Python	`count = 0`
	`while count < limit:`
	`  ...`
	`  count = count + 1`
	`# Indention marks loop body`

VB .NET	`Count = 1`
	`While (Count <= Limit)`
	`  ...`
	`  Count = Count + 1`
	`End While`

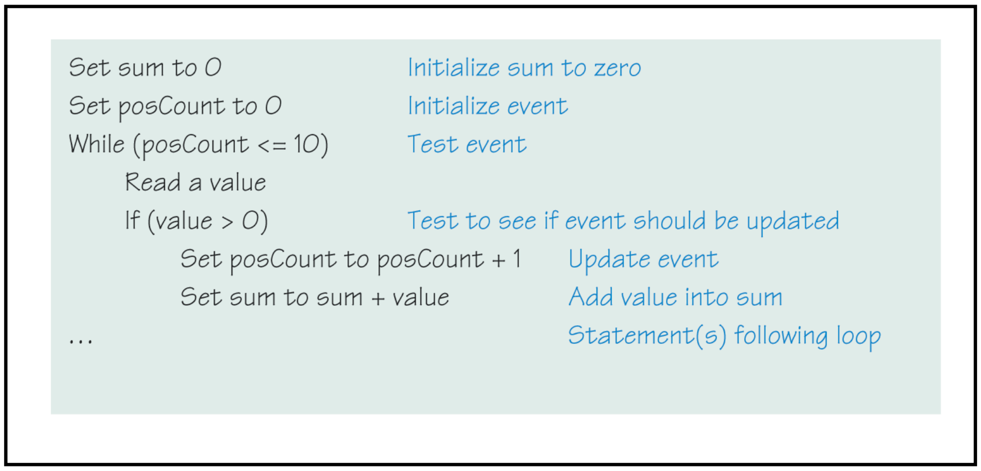
C++/Java	`count = 1;`
	`while (count <= limit)`
	`{`
	`  ...`
	`  count = count + 1;`
	`}`

Event-controlled loops

- The number of repetitions is controlled by an event that occurs within the body of the loop itself



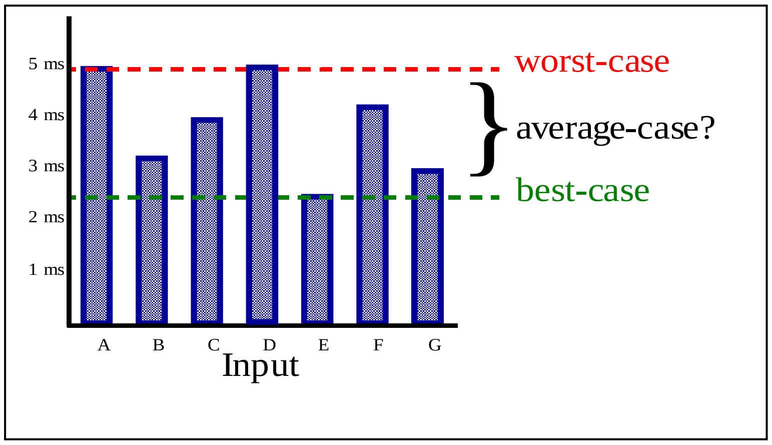
```
Read a value              Initialize event
While (value >= 0)        Test event
    . . .                 Body of loop
    Read a value          Update event
. . .                     Statement(s) following loop
```



```
Set sum to 0                      Initialize sum to zero
Set posCount to 0                 Initialize event
While (posCount <= 10)            Test event
    Read a value
    If (value > 0)                Test to see if event should be updated
        Set posCount to posCount + 1      Update event
        Set sum to sum + value            Add value into sum
. . .                                     Statement(s) following loop
```

Algorithm Analysis

- **Space complexity** : How much space is required . The amount of memory required by an algorithm to run to completion. Some algorithms may be more efficient if data completely loaded into memory . Need to look also at system limitations
- **Time complexity** : How much time does it take to run the algorithm. Often, we deal with estimates! The running time of an algorithm or a data structure method typically grows with the input size, although it may also vary for different inputs of the same size. Also, the running time is affected by a lot of factors, such as the hardware environment and the software environment.
- Algorithms **running time** **is** an important issue
- Typically algorithms are measured by their **worst case**



Running Time

- **The running time of an algorithm varies with the inputs, and typically grows with the size of the inputs**

- **To evaluate an algorithm or to compare two algorithms, we focus on their relative rates of growth with respect to the increase of the input size.**

Ethical Issues:

- **When a large software systems are developed by many people, how should liabilities be assigned. Is there hierarchy of responiosblity. Are there degree of liability.**

- **We have seen that large, complex software systems are often developed by many individuals, few of which may have a complete picture of a an entire project. Is it ethically proper for an employee to contribute to a project without full knowledge of its function ??**