References:

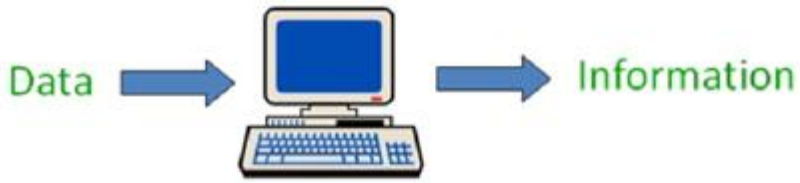
1. Gary J. Bronson, "C++ for Engineers and Scientists", 4th edition, 2013.
2. John r. Hubbard, " Schaum's Outline of Theory and Problems of Programming With C++", 1996
3. H. M. Deitel and P.J. Deitel, **"**C++ How to program**"**, 8th edition, 2012, Prentice Hall.

Syllabus

1. Introduction to Computers, Algorithms and C++ programming
2. Problem-Solving Methodology
3. Programming Basics (data types, operators and expressions, keywords and identifiers, variables and assignment, basic input/output routines)
4. Control Structures (sequence, selection [if, if\else, switch], iteration [for, while, do/while]), break and continue statements
5. Arrays (One dimension and Two dimension)
6. Functions
7. Structures
8. Pointers
9. File Input/Output
10. Introduction to Object-Oriented Programming (classes and objects)
11. Advanced Topics (Templates, Standard Template Library, Operator Overloading)

Computer system

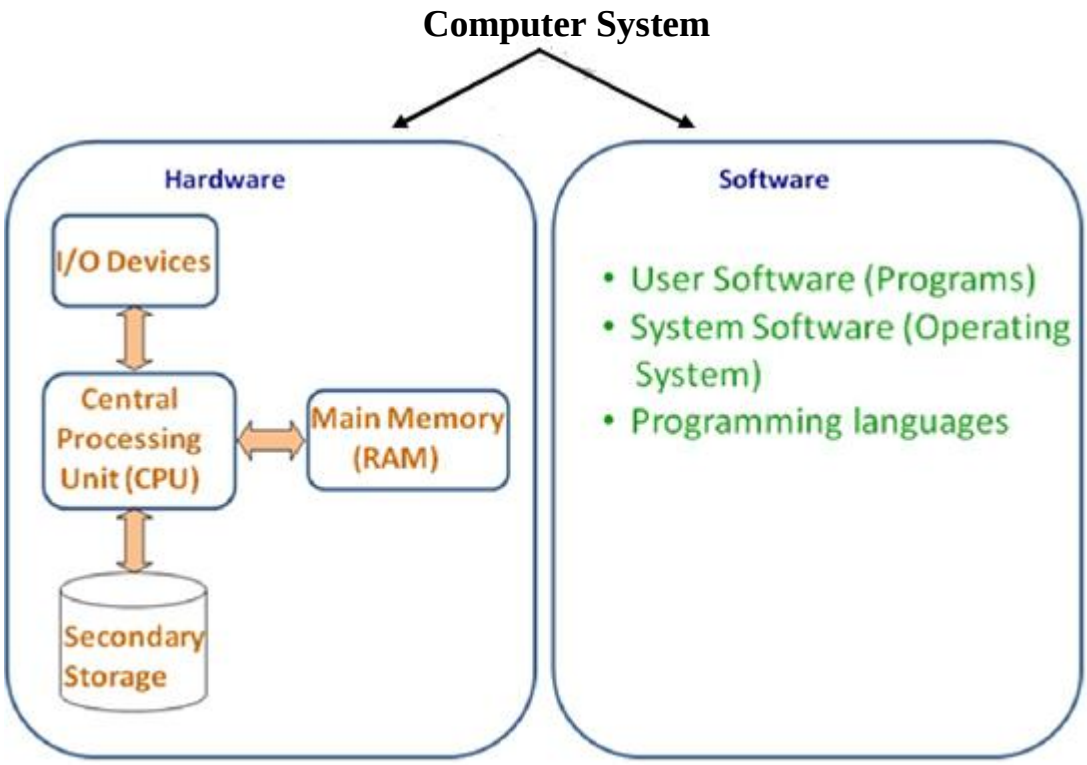
The **computer** is a machine that is designed to perform operations that are specified with a set of instructions called a program. It can perform computations and make logical decisions faster than human beings can.



Information could be:

- Results of scientific calculations
- Results of statistical analysis
- Conclusions of logic deduction
- Results of information retrieval or data summary

The computer system consists of **Hardware** and **Software.**



A **program** is a sequence of instructions that instructs the computer what to do.

What is programming?

- Programming is the ability to get computer to perform useful tasks for us
- Programming is about problem solving
- Programming is about solution development
  - From the problem to an algorithm
  - From algorithm to a program
- Programming is about logic
- Programming is about how computer works

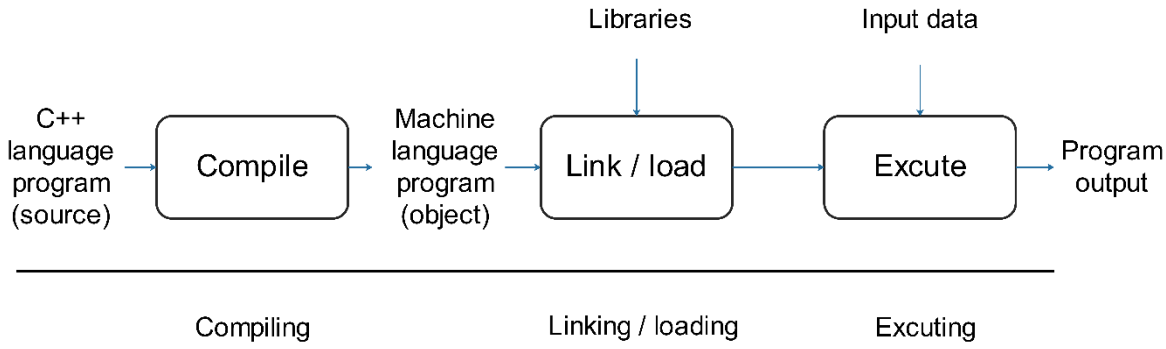
Programming Languages

- Types of Programming Languages
  - Machine language
  - Assembly language
  - High level languages
- High Level Programming Languages
  - Imperative (Procedural) Languages e.g. Bascal and C
  - Object-Oriented Languages e.g. C++ and Java
  - Other Types of Languages e.g. Prolog
- Programming Style
  - Structured (procedural) programming
  - Object-oriented programming
  - Event-driven programming

C++ Program Execution

- *Editing*: writing program source code
- *Compiling*: finding syntax errors, if errors are found then debug, else generate object program (program in machine language)
- *Debugging*: correcting compile-time errors , then recompile
- *Linking*: connecting to existing utilities (e.g. Libraries)
- *Loading*: loading the program into memory
- *Running*: execute (run) the program, if run-time errors are found then correct them and recompile.

The processes of compilation, linking/loading, and execution are outlined in the figure below.



Problem-Solving Methodology

The process for problem-solving has five steps:

1. State the problem clearly.
2. Describe the input and output information.
3. Work the problem by hand (or with a calculator) for a simple set of data.
4. Develop an algorithm and convert it to a computer program. The algorithm can be listed as operations that are performed one after another.
5. Test the program with a variety of data.

Development of Algorithms

To describe the process of solving a problem with sequential logic, we use the **top-down design**. There are several ways to develop an algorithm:

- *Decomposition outline:* is written sequentially and represents an ordered set of steps.

  Example: Compute the straight-line distance between two points in a plane.

  Decomposition outline:

  1. Give values to the two points.
  2. Compute the lengths of the two sides of the right triangle generated by the two points.
  3. Compute the distance between the two points, which is equal to the length of the hypotenuse of the triangle.
  4. Print the distance between the two points.

- *Pseudocode:* uses English-like statements to describe the steps in an algorithm.
- *Flowcharts:* uses a diagram to do the same.

The next example shows an algorithm described by a pseudocode and flowchart.

Example:

Pseudocode Notation

main:

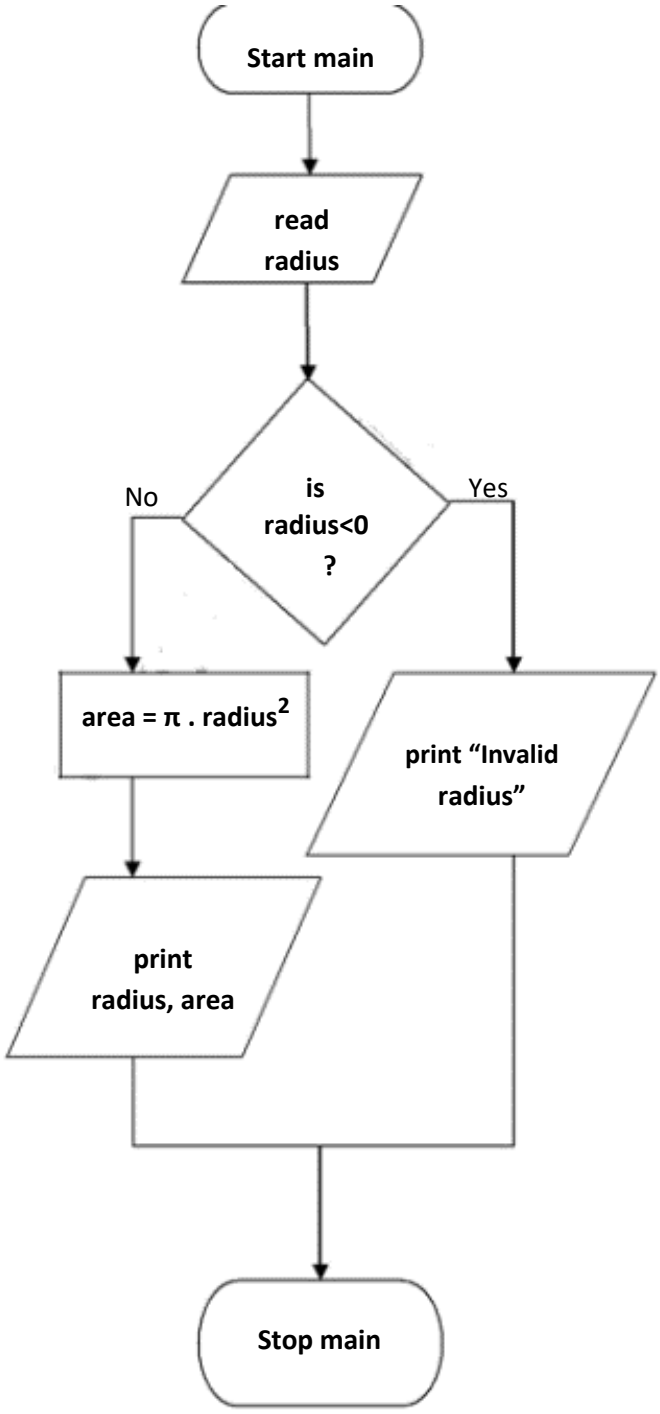
read radius

if radius < 0 then
print **“**Invalid radius**”**
else
set area to π . radius^2

print radius, area

end

Flowchart Symbol



C++ Program Structure

- Comments
- Libraries
- main function
  - Statements that define memory locations (constants and variables)
  - Statements that specify actions to be taken (i.e. flow of controls)
  - Operational expressions involving data representations
- Representations of data of different types (e.g. structure)
- Other Functions and procedures

```
// Comments
.....
.....
// Comments

Links to Libraries
...... Lib 1
......
...... Lib M
Links to Libraries

Main Program
{
   input & initialisation
   Expressions, Statements
   function calls
   output & termination
}
```

A Simple C++ Program

```
// welcome.cpp
//
// This program prints
// a welcome message
```

```
#include <iostream.h>

void main()
{
  // print welcome message
  cout<<"Welcome to C++ programming";
}
```

Line-By-Line Explanation

1. // indicates that everything following it until the end of the line is a comment: it is ignored by the compiler. Another way to write a comment is to put it between /* and */ (e.g. x = 1 + /*sneaky comment here*/ 1;). A comment of this form may span multiple lines. Comments exist to explain non-obvious things going on in the code. Use them: document your code well!
2. Lines beginning with # are preprocessor commands, which usually change what code is actually being compiled. #include tells the preprocessor to dump in the contents of another file, here the iostream file, which defines the procedures for input/output.
3. void or int main() {...}defines the code that should execute when the program starts up. The curly braces represent grouping of multiple commands into a block. More about this syntax in the next few lectures.
4. cout << : This is the syntax for outputting some piece of text to the screen.