CSE 1061 Fundamentals of Programming

Lecture #1

Spring 2015

Computer Science & Engineering Program
The School of EE & Computing
Adama Science & Technology University

Problem Solving and Computer Programming

- Problem solving life cycle
- Basics of programming language

Case study: Heat Transfer
Guess a Number Game

Reading assignment

- Chapter 1 of the textbook
- Read about
  - ❖ Computer Storage
  - ❖ How a computer runs programs?
  - ❖ Pseudo Code and Flowchart

-Heat Transfer Notes
http://www.freestudy.co.uk/heat%20transfer/

Outline

- Problem Solving Life Cycle
  - Phase 1: Development
  - Phase 2: Documentation
  - Phase 3: Maintenance
- Basics of Programming Language
  - Programming Languages
    - Low level and High level languages
    - Compiled vs. Interpreted
    - Procedural and Object Oriented
  - Compilation in C++
  - General Notes on C++

Software Development

- Computer Program
  - Self-contained set of instructions used to instruct a computer to produce specific result

- Problem Solving Life Cycle
  - Also called Software Development Procedure
  - Helps developers understand the problem to be solved and create an effective, appropriate software solution

Problem Solving Life Cycle

- Requirement
- Phase 1: Development
  - Analysis
  - Design
  - Coding
  - Testing
- Phase 2: Documentation
- Phase 3: Maintenance

Requirement

- Request for a program or statement of a problem
- After a program requirement is received, phase 1 begins.
- In the real world, it is a complex task of extracting and documenting user requirements

Phase 1: Development

- Step 1: Analyze the Problem
  - Determine and understand the output items the program must produce
  - Determine the input items

Phase 1: Development

- Step 2: Develop a Solution
  - Select the exact set of steps, called an "algorithm", to solve the problem
  - Refine the algorithm
    - Start with initial solution in the analysis step until you have an acceptable and complete solution
  - Check solution
- ***Read about Pseudo Code & Flowchart

Phase 1: Development

- Step 3: Code the Solution
  - Consists of actually writing a C++ program that corresponds to the solution developed in Step 2
  - Program should contain well-defined patterns or structures of the following types:
    - Sequence
    - Selection
    - Iteration
    - Invocation

Phase 1: Development

- Step 3: Code the Solution
  - **Sequence**: Defines the order in which instructions are executed
  - **Selection**: Allows a choice between different operations, based on some condition
  - **Iteration**: Allows the same operation to be repeated based on some condition
    - Also called looping or repetition
  - **Invocation**: Involves invoking a set of statements when needed

Phase 1: Development

- Step 4: Test and Correct the Program
  - **Testing**: Method to verify correctness and that requirements are met
  - **Bug** : A program error
  - **Debugging** : The process of locating an error, and correcting and verifying the correction
  - Testing may reveal errors, but does not guarantee the absence of errors

Phase II: Documentation

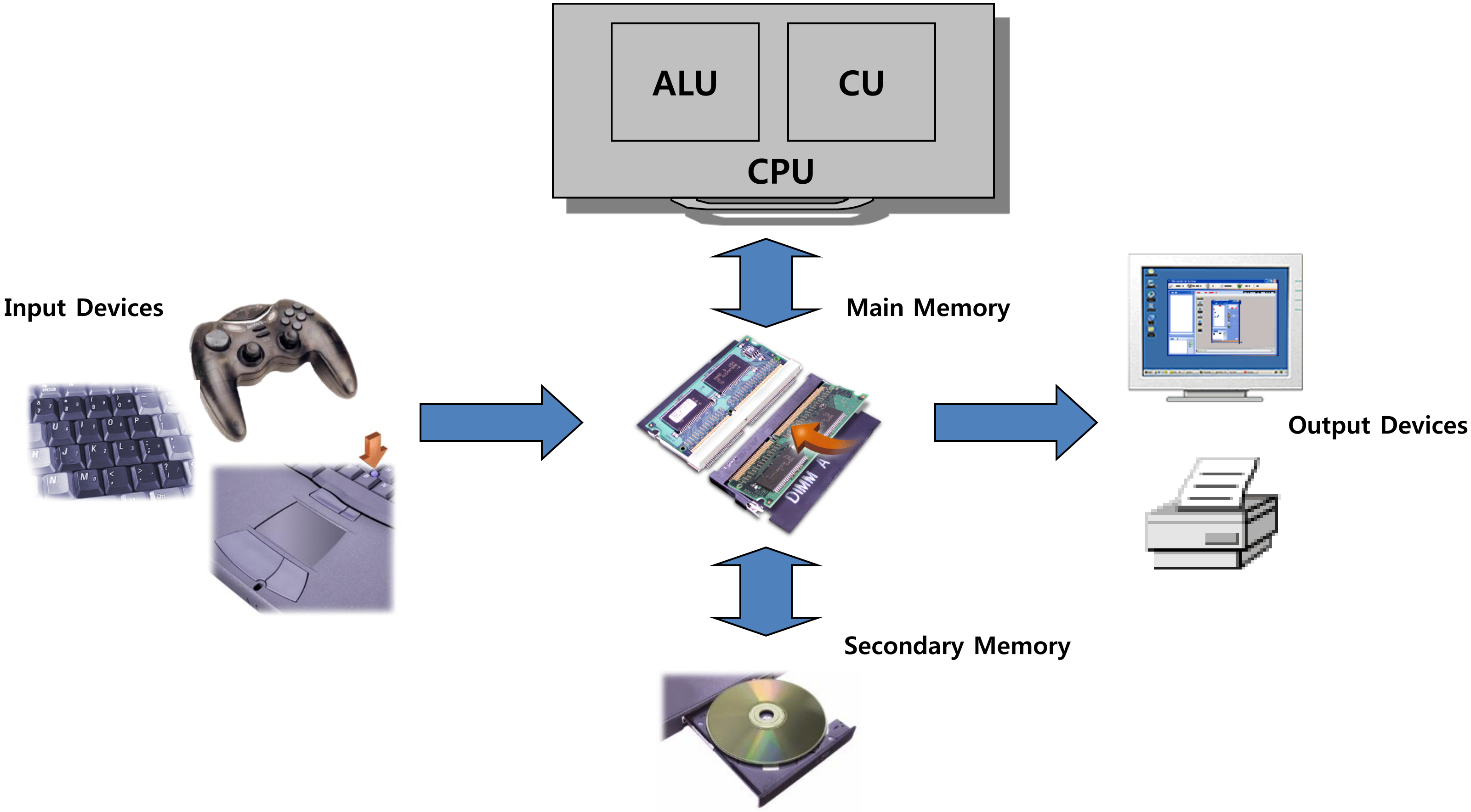
- Five main documents are needed:
  - Program description
  - Algorithm development and changes
  - Well-commented program listing
  - Sample test runs
  - Users' manual

Phase III: Maintenance

- Ongoing correction of newly discovered bugs
- Revisions to meet changing user needs
- Addition of new features
- Usually the longest phase
- Good documentation vital for effective maintenance

Computer Hardware

- Computer hardware: Components that support the capabilities of the computer



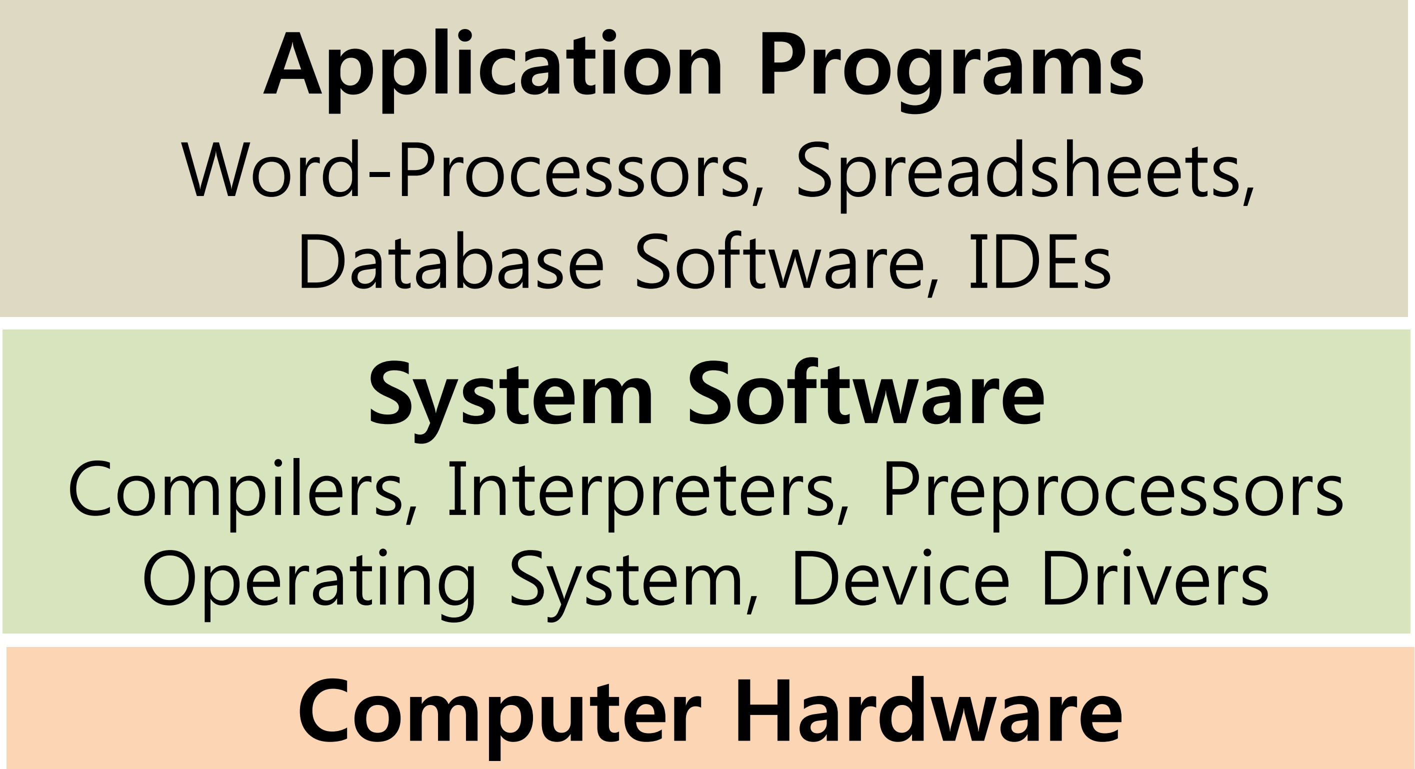
Computer Software

- **Application software:** Programs written to perform particular tasks for users
- **System software:** Collection of programs to operate the computer system

Computer Software

- **Operating system:** The set of system programs used to operate and control a computer
- Tasks performed by the OS include:
  - Memory management
  - Allocation of CPU time
  - Control of input and output
  - Management of secondary storage devices

A Layered View of the Computer



Application Programs

Word-Processors, Spreadsheets, Database Software, IDEs

System Software

Compilers, Interpreters, Preprocessors Operating System, Device Drivers

Computer Hardware

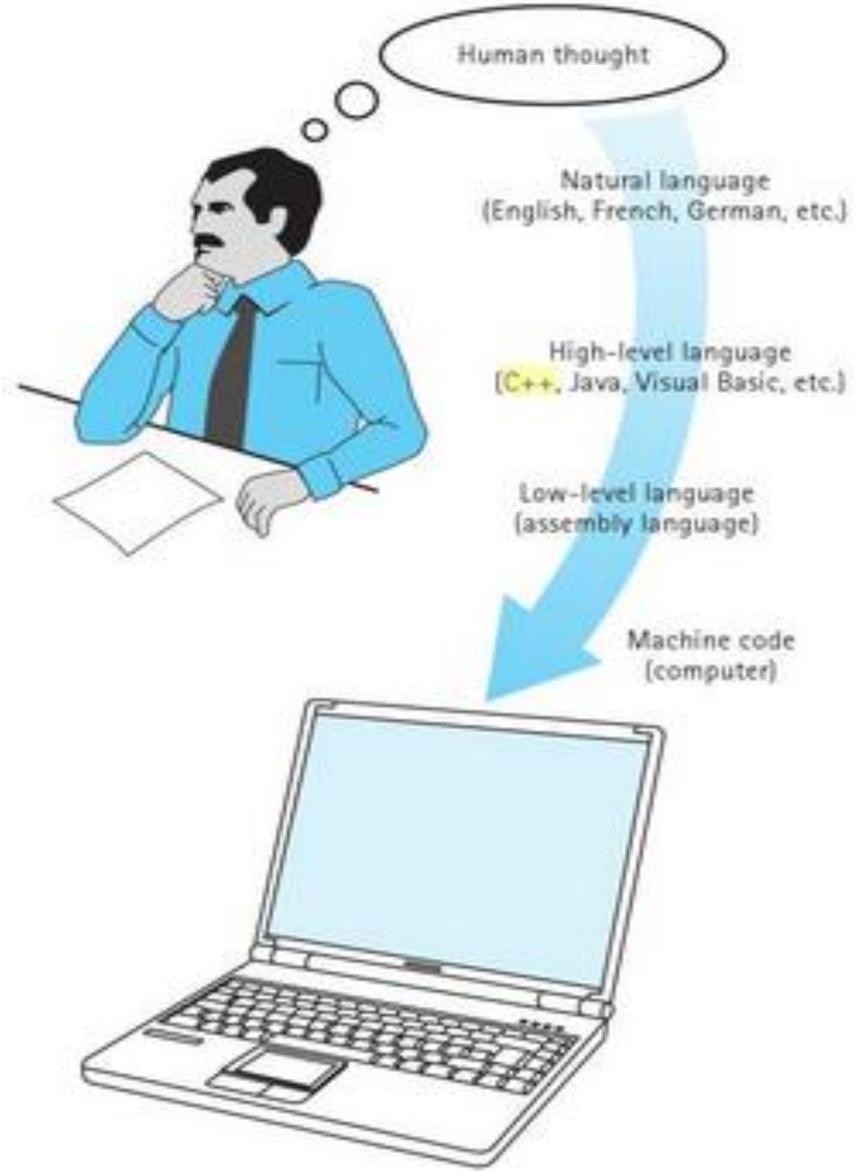
Programming Languages

- **Programming**: Process of writing a program, or software
- **Programming language**
  - Set of instructions used to construct a program
  - Comes in a variety of forms and types

Low and High Level Languages

- **Low-level languages:** Languages that use instructions tied directly to one type of computer
  - machine language, assembly language
- **High-level languages:** Instructions resemble written languages, such as English
  - Can be run on a variety of computer types
  - Visual Basic, C, C++, Java

Low and High Level Languages



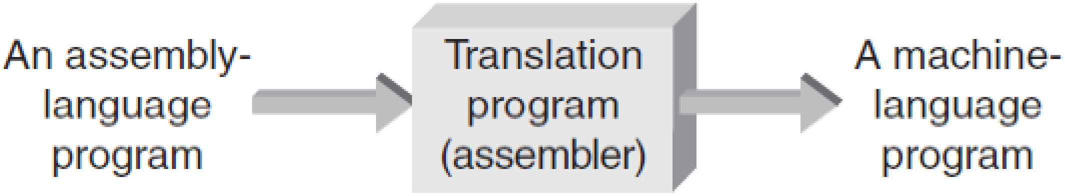
Machine Language

- **Machine language programs** : only programs that can actually be used to operate a computer
  - Also referred to as object codes
  - Consists of a sequence of instructions composed of binary numbers
- Contains two parts: an instruction and an address

Operation	Address
0010	0000 0000 0100
0100	0000 0000 0101
0011	0000 0000 0110

Assembly Language

- **Assembly language programs:** Substitute word-like symbols, such as ADD, SUB, and MUL, for binary opcodes
  - Use decimal numbers and labels for memory addresses
  - One-to-one correspondence
    - Usually, each line of assembly code produces one machine instruction
  - Example: ADD 1, 2
- **Assemblers**: Translate programs into machine language



An assembly-language program

Translation program (assembler)

A machine-language program

High Level Language

- One-to-many correspondence
  - Each statement corresponds to several machine language instructions
- Much easier to program than in assembly language.
- Data are referenced using descriptive names
- Operations can be described using familiar symbols
- Example:

cost = price + tax

Compiled vs. Interpreted

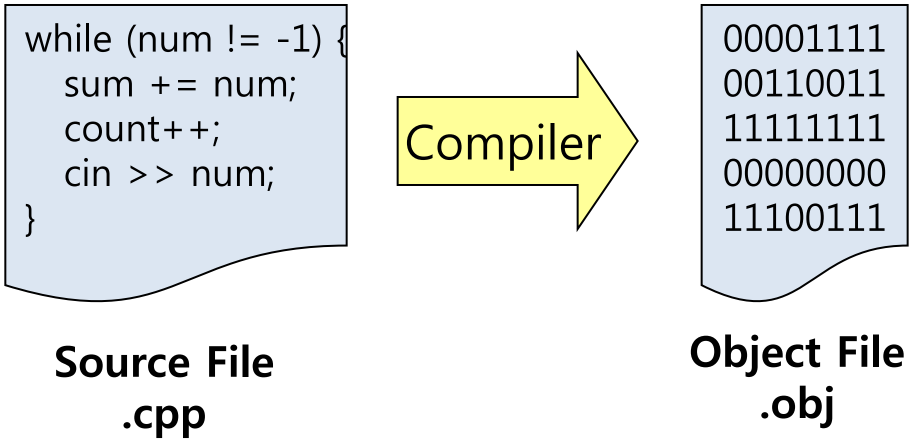
- **Interpreted**: Each statement is translated individually and executed immediately after translation
- **Compiled**: All statements are translated and stored as an executable program, or object program; execution occurs later
    - C++ is predominantly a compiled language

Source Code

- The programs written in a high or low-level language
- It is written by the programmer.
- Can be compiled automatically into object code or machine code or executed by an interpreter.
- C++ source programs have extension '.cpp' or '.h'

Compiler

- Translates high-level language to machine language.
- Input to a compiler is the **source code**
- Output from the compiler is **the object code** (.obj).



```
while (num != -1) {
    sum += num;
    count++;
    cin >> num;
}
```

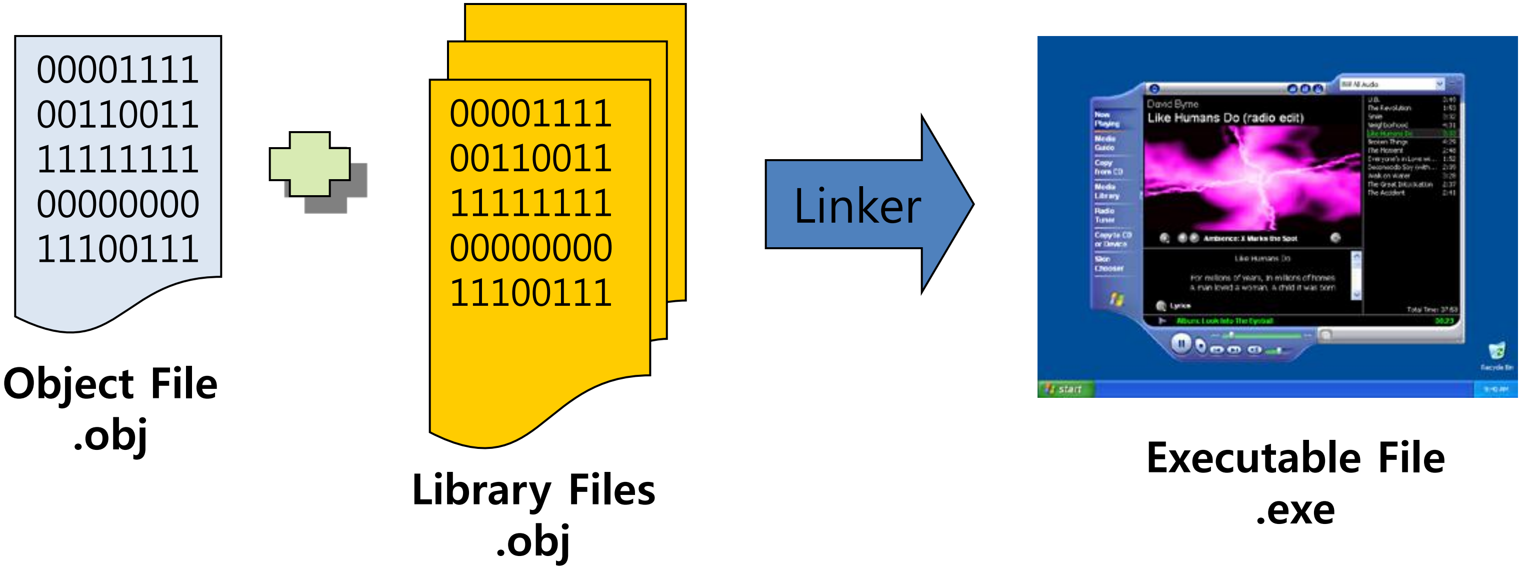
```
00001111
00110011
11111111
00000000
11100111
```

Source File
.cpp

Object File
.obj

Linker

- Combines your object code with pre-compiled object libraries for input/output etc.
- Input are the object files (.obj).
- Output is the executable file (.exe) in Windows.



00001111
00110011
11111111
00000000
11100111

Object File
.obj

00001111
00110011
11111111
00000000
11100111

Library Files
.obj

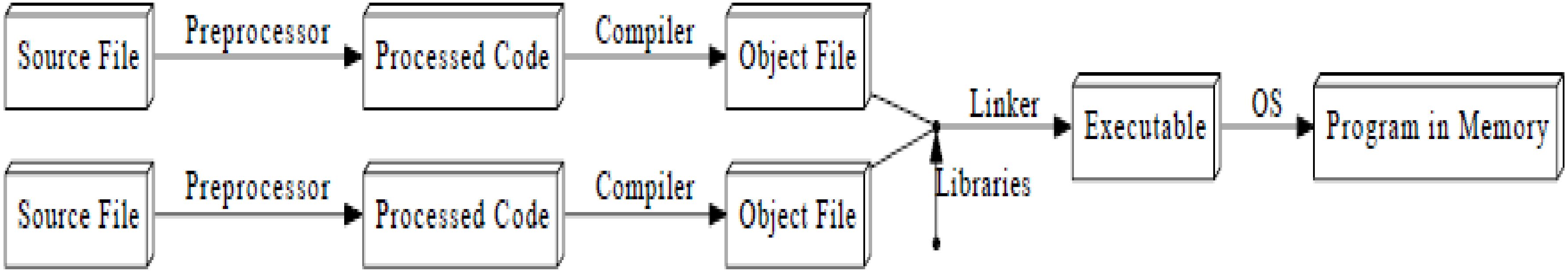
Linker

Executable File
.exe

Compilation Process in C++

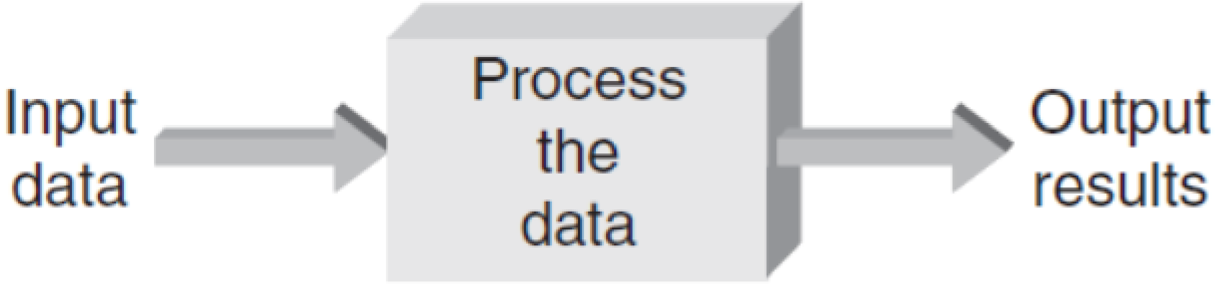
- C++ actually adds an extra step to the compilation process
  - the code is run through a preprocessor, which applies some modifications to the source code, before being fed to the compiler.

    Eg. Removing comment



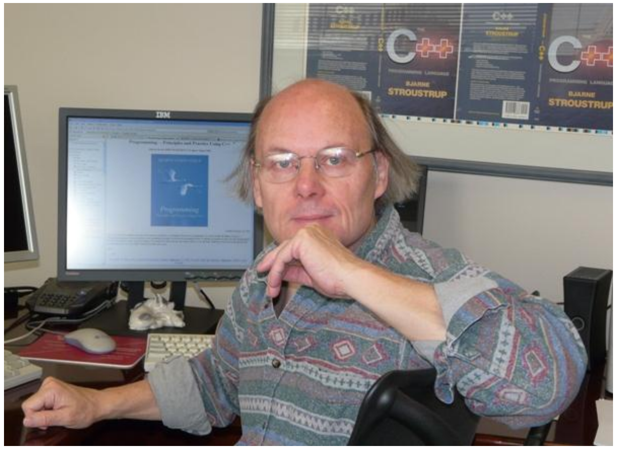
Program in Memory

- The purpose of most application programs is to process data to produce specific results



General Notes on C++

- C++ is immensely popular
  - particularly for applications that require speed and/or access to some low-level features.
- It was created in 1979 by Bjarne Stroustrup
  - at first as a set of extensions
  - to the C programming language.



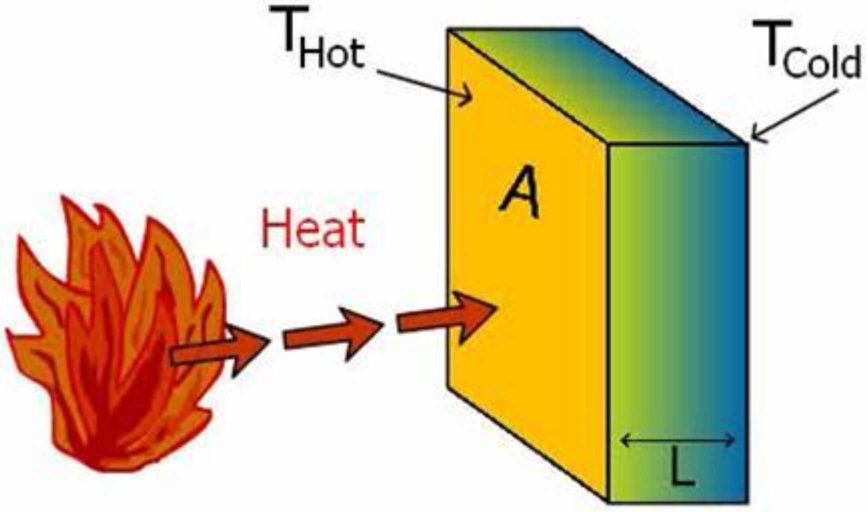
General Notes on C++

- You can write graphical programs in C++
    - it is much harder and less portable than text based(console)programs.
    - We will be sticking to console programs in this course.
- Everything in C++ is case sensitive:
    - someName is not the same as SomeName.

Case Study: Heat Transfer

- The following formula is used to determine the heat transferred through a flat substance, such as a wall, with its two sides maintained at different temperatures:

$$\mathrm{q} = -k\left(\frac{T_2 - T_1}{d}\right)$$



Case Study: Heat Transfer

- q is the heat transfer.
- k is the thermal conductivity of the substance in watts/m°C.
- T_2 is the higher temperature on one side of the substance in °C.
- T_1 is the lower temperature on the other side of the wall in °C.
- d is the thickness of the substance in meters.

Case Study: Heat Transfer

- Determine the units of q
  - by calculating the units resulting from the right side of the formula.
- A glass window
  - thickness of 0.5 centimeters,
  - thermal conductivity of 0.77 watts/m°C, and
  - outside and inside temperatures of 36.6°C & 22.2°,
  - Determine the value of q.

Case Study: Heat Transfer

- If you were asked to write a computer program to determine the heat transfer through a substance, what **inputs**, **outputs**, and **algorithm** would your program require?

Step 1: Analyze the Problem

- The objective is to randomly generate an integer greater than or equal to 0 and less than 100. Then prompt the player (user) to guess the number.
- **Input**: Guessed Number
- **Output**: Guess is Correct or Guess is Higher, or Guess is Lower

Case Study: Guess a Number Game

Step 2: Develop a Solution

1. Accept guessed number from user
2. If the player guesses the number correctly, output an appropriate message.
3. If the guessed number is less than the random number generated

   display "guess is lower than the number. Guess again!";

4. Otherwise

   display "guess is higher than the number. Guess again!";

Case Study: Guess a Number Game

Step 3: Then prompt the player to enter another number.

Step 4: The player is prompted to guess the random number until the player enters the correct number and stop after fixed no of trials .