Problem Solving

Problem Solving

- Computer programs, known as *software*, are instructions to the computer. You tell a computer what to do through programs.
- Without programs, a computer is an empty machine.
- Computers do not understand human languages, so you need to use computer languages to communicate with them. Programs are written using programming languages.
- Programming is a process of problem-solving (Problem Solution by computer)

Problem Solving Process

Phase 1 – Analysis and Design

- Analyze the problem by outlining the problem and its requirements
- Design (algorithm) to solve the problem (Flow chart, pseudo code)
- Algorithm tracing

Algorithm?

Step-by-step problem-solving process

Phase 2 - Implementing the algorithm

- Implement the algorithm in code (in Programming Language → Program)
- Verify that the algorithm works

Phase 3 - Maintenance

- Use and modify the program if the requirements change

Analyze the Problem

Understand the problem and the requirements

- Does the program require user interaction?
- Does the program manipulate data?
- What is the output?
- Are all possible circumstances handled?

If the problem is complex, divide it into subproblems

- Analyze each subproblem as above

Example:

Convert a student mark from decimal mode to ABC mode.

Understand problem requirements

- Does program require user interaction? →*Input the mark*
- Does program manipulate data? → *covert mark* → Control Requirements (**IF statement**)
- What is the output? *The mark converted to A or B or C or error*
- Is there subproblem? *No*

Design

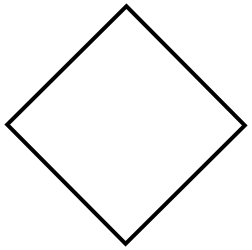
Algorithm Design

1. *Flowcharts*
2. *pseudo-code*

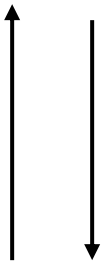
Design

Flow Chart Symbols

Start and End



Selection



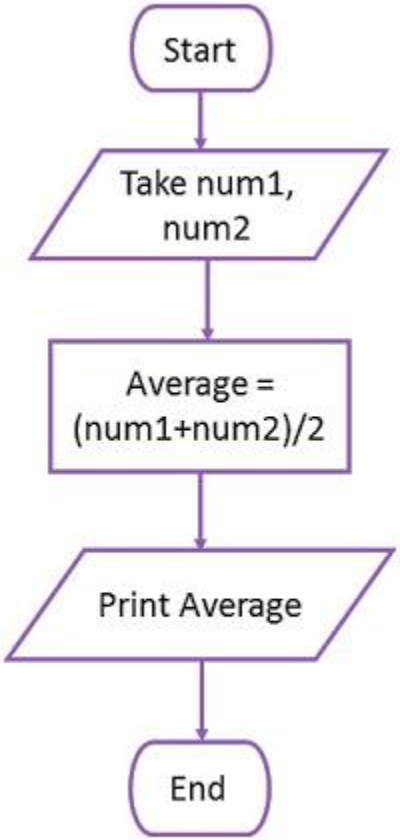
Data Flow

Input / output

Calculation

Design

Flowchart example



Design

Pseudo-code

<Algorithm name>

// input ?

// function?

// Output?

Begin

<data definition>

<actions>

End

The comment lines **“//”**

Design

Algorithm Tracing

- Draw flowchart
- Find all possible paths
- Check each path with appropriate input data
- Observed Outputs do not conform to the expected ones → error in the algorithm.

Efficiency:

- *an algorithm may work correctly but be inefficient – by taking more time and using more resources than required to solve the problem.*

*- becomes more important for larger programs***.**

Implementing the algorithm

- Implementation is also called **Coding**.
- After testing your algorithm, you can code it in any programming language.
- In our lab, we are going to use C++ language.

Implementing the algorithm

Verify that the algorithm works.

An error could occur during:

- analyzing the problem,
- developing an algorithm, and/or
- coding the algorithm

Errors are of three types:

- syntax errors
- run-time errors
- logic errors

Implementing the algorithm

- Syntax errors: are detected by the C compiler
  - Source code does not conform to one or more of C's grammar rules
  - Example of syntax errors: undeclared variable, …
  - Often one mistake leads to multiple error messages – which can be confusing
- Run-time errors: detected and displayed by the computer during execution.
  - Occurs when a program directs a computer to perform an illegal operation. Example: int x=y/0;
  - will stop program execution and display a message
- Logic errors: caused by faulty algorithm
  - sign of error: incorrect program output
  - cure: thorough testing and comparison with expected results

A logic error is referred to as a bug, so finding logic errors is called debugging.

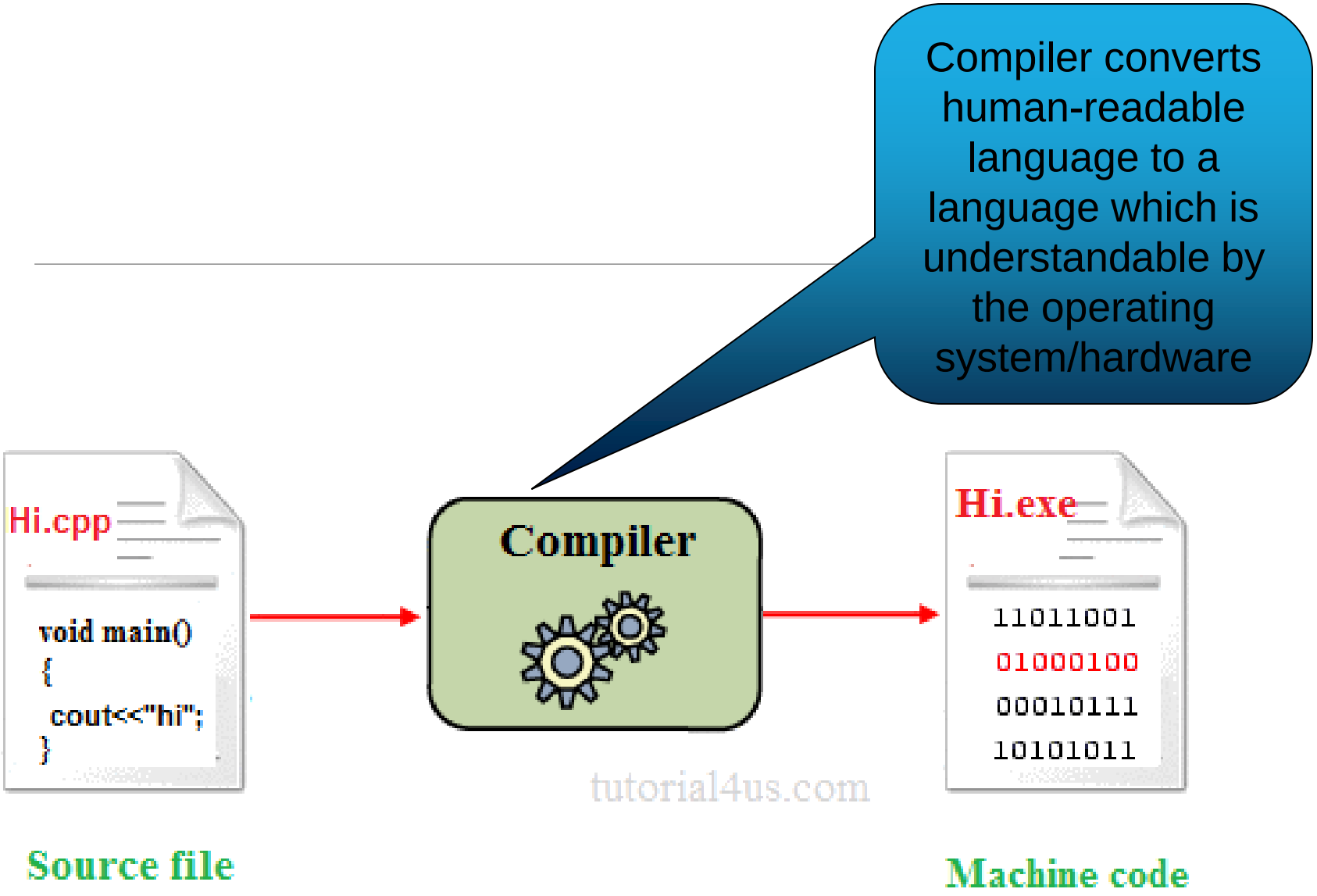
Implementing the algorithm

After writing the code, the computer must do the following:

- Compilation:
- Execution:

Compilation

- Performed by a program called the **compiler**
- Translates the preprocessor-modified source code into **object code (machine code)**
- Checks for **syntax errors** and **warnings**
- Saves the object code to a disk file
- If any compiler errors are received, no object code file will be generated.
- An object code file will be generated if only warnings, not errors, are received.


Compiler converts human-readable language to a language which is understandable by the operating system/hardware
Hi.cpp
void main()
{
cout<<"hi";
}
Compiler
Hi.exe
11011001
01000100
00010111
10101011
tutorial4us.com
Source file
Machine code

Execution

when the program does not have errors, the computer executes it to produce the output.