Chapter 1: An Overview of Computers and Programming Languages
In this chapter, you will:

• Learn about different types of computers
• Explore the hardware and software components of a computer system
• Learn about the language of a computer
• Learn about the evolution of programming languages
Objectives (cont'd.)

- Discover what a compiler is and what it does
- Examine a C++ program
- Explore how a C++ program is processed
- Become aware of Standard C++ and ANSI/ISO Standard C++
- Learn about computer program and the importance of writing good programs.
- Explore Program Development Life Cycle (PDLC)
Without software, the computer is useless.

- Software developed with programming languages
  - C++ is a programming language
- C++ suited for a wide variety of programming tasks
- Before programming, it is useful to understand terminology and computer components
A Brief Overview of the History of Computers

- Early calculation devices
  - Abacus, Pascaline
  - Leibniz device
  - Babbage machines: difference and analytic engines
  - Hollerith machine
A Brief Overview of the History of Computers (cont'd.)

- Early computer–like machines
  - Mark I
  - ENIAC
  - Von Neumann architecture
  - UNIVAC
  - Transistors and microprocessors
  - Chips – smaller and cheaper

- Modern Computers
  - Apple & IBM PC
  - Modern computers AI, Expert System, Mobile Computing
A Brief Overview of the History of Computers (cont'd.)

- Categories of computers
  - Mainframe computers
  - Midsize computers
  - Micro computers (personal computers)
Elements of a Computer System

- Hardware
- CPU
- Main memory
- Secondary storage
- Input/Output devices
- Software
Hardware

- CPU
- Main memory: RAM
- Input/output devices
- Secondary storage
Central Processing Unit and Main Memory

- **Central processing unit**
  - Brain of the computer
  - Most expensive piece of hardware
  - Carries out arithmetic and logical operations
Central Processing Unit and Main Memory (cont'd.)

![Diagram of computer hardware components and main memory]

**FIGURE 1-1** Hardware components of a computer and main memory
Central Processing Unit and Main Memory (cont'd.)

- Random access memory
- Directly connected to the CPU
- All programs must be loaded into main memory before they can be executed
- All data must be brought into main memory before it can be manipulated
- When computer power is turned off, everything in main memory is lost
Secondary Storage

- **Secondary storage**: device that stores information permanently
- **Examples of secondary storage:**
  - Hard disks
  - Flash drives
  - Floppy disks
  - Zip disks
  - CD–ROMs
  - Tapes
Input/Output Devices

- **Input devices** feed data and programs into computers
  - Keyboard
  - Mouse
  - Secondary storage
- **Output devices** display results
  - Monitor
  - Printer
  - Secondary storage
Software:

- **Software**: programs that do specific tasks
- **System programs** take control of the computer, such as an operating system
- **Application programs** perform a specific task
  - Word processors
  - Spreadsheets
  - Games
The Language of a Computer

- Digital signals: sequences of 0s and 1s
- Machine language: language of a computer
- Binary digit (bit):
  - The digit 0 or 1
- Binary code:
  - A sequence of 0s and 1s
- Byte:
  - A sequence of eight bits
# The Language of a Computer (cont’d.)

## TABLE 1-1 Binary Units

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
<th>Bits/Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td></td>
<td>8 bits</td>
</tr>
<tr>
<td>Kilobyte</td>
<td>KB</td>
<td>$2^{10}$ bytes = 1024 bytes</td>
</tr>
<tr>
<td>Megabyte</td>
<td>MB</td>
<td>$1024 \text{ KB} = 2^{10} \text{ KB} = 2^{20}$ bytes = 1,048,576 bytes</td>
</tr>
<tr>
<td>Gigabyte</td>
<td>GB</td>
<td>$1024 \text{ MB} = 2^{10} \text{ MB} = 2^{30}$ bytes = 1,073,741,824 bytes</td>
</tr>
<tr>
<td>Terabyte</td>
<td>TB</td>
<td>$1024 \text{ GB} = 2^{10} \text{ GB} = 2^{40}$ bytes = 1,099,511,627,776 bytes</td>
</tr>
<tr>
<td>Petabyte</td>
<td>PB</td>
<td>$1024 \text{ TB} = 2^{10} \text{ TB} = 2^{50}$ bytes = 1,125,899,906,842,624 bytes</td>
</tr>
<tr>
<td>Exabyte</td>
<td>EB</td>
<td>$1024 \text{ PB} = 2^{10} \text{ PB} = 2^{60}$ bytes = 1,152,921,504,606,846,976 bytes</td>
</tr>
<tr>
<td>Zettabyte</td>
<td>ZB</td>
<td>$1024 \text{ EB} = 2^{10} \text{ EB} = 2^{70}$ bytes = 1,180,591,620,717,411,303,424 bytes</td>
</tr>
</tbody>
</table>
The Language of a Computer (cont'd.)

- **ASCII (American Standard Code for Information Interchange)**
  - 128 characters
  - A is encoded as 1000001 (66th character)
  - 3 is encoded as 0110011
The Language of a Computer (cont'd.)

- EBCDIC
  - Used by IBM
  - 256 characters

- Unicode
  - 65536 characters
  - Two bytes are needed to store a character
The Evolution of Programming Languages

- Early computers were programmed in machine language

To calculate \( \text{wages} = \text{rates} \times \text{hours} \) in machine language:

\[
\begin{align*}
100100 & \ 010001 \quad \text{//Load} \\
100110 & \ 010010 \quad \text{//Multiply} \\
100010 & \ 010011 \quad \text{//Store}
\end{align*}
\]
The Evolution of Programming Languages (cont'd.)

- Assembly language instructions are mnemonic
- Assembler: translates a program written in assembly language into machine language

**TABLE 1-2 Examples of Instructions in Assembly Language and Machine Language**

<table>
<thead>
<tr>
<th>Assembly Language</th>
<th>Machine Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD</td>
<td>100100</td>
</tr>
<tr>
<td>STOR</td>
<td>100010</td>
</tr>
<tr>
<td>MULT</td>
<td>100110</td>
</tr>
<tr>
<td>ADD</td>
<td>100101</td>
</tr>
<tr>
<td>SUB</td>
<td>100011</td>
</tr>
</tbody>
</table>
The Evolution of Programming Languages (cont'd.)

- Using assembly language instructions, \( \text{wages} = \text{rates} \times \text{hours} \) can be written as:

  \[
  \begin{align*}
  \text{LOAD} & \quad \text{rate} \\
  \text{MULT} & \quad \text{hour} \\
  \text{STOR} & \quad \text{wages}
  \end{align*}
  \]
The Evolution of Programming Languages (cont'd.)

- High-level languages include Basic, FORTRAN, COBOL, Pascal, C, C++, C#, and Java
- **Compiler**: translates a program written in a high-level language into machine language
- The equation \( \text{wages} = \text{rate} \times \text{hours} \) can be written in C++ as:
  \[
  \text{wages} = \text{rate} \times \text{hours};
  \]
#include <iostream>
using namespace std;
int main()
{
    cout << "My first C++ program." << endl;
    return 0;
}

Sample Run:
My first C++ program.
Processing a C++ Program (cont'd.)

- To execute a C++ program:
  - Use an editor to create a source program in C++
  - Preprocessor directives begin with # and are processed by a preprocessor
  - Use the compiler to:
    - Check that the program obeys the rules
    - Translate into machine language (object program)
To execute a C++ program (cont'd.):

- **Linker:**
  - Combines object program with other programs provided by the SDK to create executable code
- **Loader:**
  - Loads executable program into main memory
- The last step is to execute the program
Processing a C++ Program (cont'd.)

**FIGURE 1-3** Processing a C++ program
Two popular approaches to programming design
- Structured
- Object-oriented
Structured Programming

• **Structured design:**
  – Dividing a problem into smaller subproblems

• **Structured programming:**
  – Implementing a structured design

• The structured design approach is also called:
  – Top-down (or bottom-up) design
  – Stepwise refinement
  – Modular programming
Object-Oriented Programming

- Identify components called objects
- Specify relevant data and possible operations to be performed on that data
- Each object consists of data and operations on that data
- An object combines data and operations on the data into a single unit
A programming language that implements OOD is called an object-oriented programming (OOP) language.

Learn how to represent data in computer memory, how to manipulate data, and how to implement operations.

Write algorithms and implement them in a programming language.
Object-Oriented Programming (cont'd.)

- Learn how to combine data and operations on the data into a single unit called an object
- C++ was designed to implement OOD
- OOD is used with structured design
ANSI/ISO Standard C++

- C++ evolved from C
- C++ designed by Bjarne Stroustrup at Bell Laboratories in early 1980s
- C++ programs were not always portable from one compiler to another
- In mid-1998, ANSI/ISO C++ language standards were approved
Quick Quiz

- True or False: Main memory stores numbers and letters.
- True or False: Programs do not have to be loaded into main memory before they are executed.
- A _________ is 1024 bytes.
- The basic language of a machine is called -----------.
- What is the role of a compiler?
- True or False: A loader is a program that combines the object program with other programs in the library.
A list of instructions for computer that can be executed by a computer
The instructions are made up of statements used in a programming language
The Importance of Programming

- A computer without a program is useless.
  - Must have input, process and output data
- Programmer must have a good understanding of computers, problem solving approaches and compositions of the chosen programming languages.
A good program must have the following:

- Reliability of output
  - Produce correct output.
  - A different set of input data is used to ensure the reliability of the output.
- Program’s Efficiency
  - Produce no error during execution.
  - Achieve its goal so that the final result can be trusted.
  - Program needs to be outlined first using pseudocode or flowchart.
- Interactivity
  - Interaction between the user and the program must be well defined so that the user knows the processing status.
  - User friendly program allows user to respond to instruction correctly.
  - E.g. Give clear instruction to the user on how to complete the data entry task. Without the instruction, user might give invalid input.
The Importance of Good Programs (cont’d)

- Program readability
  - How other person (programmer) views one’s program.
  - The use of indentation and comment improve readability

- Indentation
  - Helps in making the structure of the program clearer and easier to read.
  - A statement within statement should be indented to show the user which statements subordinate of the other.
  - E.g. if-else, while and do-while statements should be indented. Embedded braces are indented to make it easier to find the matching pairs.

- Comments
  - Internal documentation
  - Convenience for anyone reading the program
  - Can be placed anywhere within program and will not be processed by the compiler.
    - Line comment
    - Block comment
Program 1.1: Compute the sum of two integer numbers

Authors: Rohaiza Bokhari

```c++
#include <iostream.h>

void main () {
    int x, y, sum;
    cout<<"Enter first number : ";
    cin>>x;
    cout<<"Enter second number : ";
    cin>>y;
    sum = x + y;
    cout<<"Sum : "<<sum;
}
```
In developing a program, there is no complete set of rules or specific algorithm to follow. Developers try to use a reasonably consistent problem solving for constructing computer programs.

- **PDLC**
  - **Framework** for each step taken in developing the computer program
  - Acts as a *guideline for the programmer* throughout the process
  - **5 steps**
    - Analysis
    - Design
    - Implementation
    - Testing/debugging
    - Maintenance
Program Development Life Cycle (cont’d)
PDLC

- Program development is a continuous process
- Diagram shows that after maintenance phase, the arrow is pointing back to the first phase
- This situation occurs when there are errors in the program and need to be rectified
- Programmer will redo back to the crucial phase that is to understand/analyze the problem.
Determine objectives, inputs, processing, outputs

Analyzing the problem to obtain a clear understanding of the problem requirement.

Involves identifying the problem

- Input
- Processing
- Output
Analyzing the problem of how to find the average mark in a class

<table>
<thead>
<tr>
<th>Input</th>
<th>Process</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marks for each student</td>
<td>Get marks for every student</td>
<td>Average mark</td>
</tr>
<tr>
<td></td>
<td>Calculate total marks for all student</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Formula: Total marks = Total Marks + Marks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Condition</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Each marks must be in the range between 0 to 100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- User must determine the last data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Calculate Average Mark</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Formula: Average = Total Marks / No of Students</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Print the Average Mark</td>
<td></td>
</tr>
</tbody>
</table>
Programmer needs to plan the program implementation by creating a structure on how to solve the problem that is known as ‘Algorithms’

- Create solutions using algorithms, pseudocode, flowcharts, or other logical structures
- Algorithm: A step-by-step problem solving process in which a solution is arrived at in a finite amount of time
- Algorithm refer to steps and procedure to solve problem in terms of
  - The actions to be executed
  - The order in which these actions are to be executed
Example 1
1. Get out of bed
2. Take off pajamas
3. Take a shower
4. Get dressed
5. Eat breakfast
6. Carpool to work

Example 2
1. Get out of bed
2. Take off pajamas
3. Get dressed
4. Take a shower
5. Eat breakfast
6. Carpool to work

Program control – specifying the order in which statements are to be executed in a computer program
Example 2
The algorithm of determining the average of five test scores.
1. Get the five test scores.
2. Add the five test scores. Suppose sum stands for the sum of the test scores.
3. Suppose average stands for the average test score. Then:
   average = sum/5
Design (cont’d)

- Two ways of presenting the algorithms
  - Pseudocode
  - Program flowchart

- Pseudocode
  - A program design tool that uses English-like statements to outline the logic of a program.
  - Artificial and informal language that helps programmers develop algorithms
  - ‘Text-based’ detail (algorithmic) design tool
  - Use a simple word and easy to understand
Pseudocode examples

Start
  Get score1, score2, score3, score4, score5
  sum = score1 + score2 + score3 + score4 + score5
  average = sum / 5
  Display average
End
Pseudocode examples

If student's grade is greater than or equal to 60
   Print "passed"
else
   Print "failed"

Set total to zero
Set grade counter to one
While grade counter is less than or equal to ten
   Input the next grade
   Add the grade into the total
Set the class average to the total divided by ten
Print the class average.
FlowChart

- A program design tool that shows graphically step by step how a computer program will process data and execute.
- Using graphic or specific symbols to represent the algorithm
  - Standardized symbols to show the steps the computer needs to take to accomplish the program’s objective
Design (cont’d)

- **Flowchart**
  - **Process** – execution of statement processes input to output
  - **Data** – Data input or output display
  - **Arrows** – continuation of the flow/to show where the flowchart goes
  - **Decision** – Condition that decides which way to go
  - **Connectors** – connectors to other pages
  - **Start/End** – symbol to start or to terminate flowchart
Design (cont’d)

- Flowchart example

```
start

Get score1, score2, score3, score4, score5

sum = score1 + score2 + score3 + score4 + score5

average = sum / 5

Display average

end
```
Implementation/Coding

- **Coding**
  - write a program using a programming language

- **Implementation**
  - Process of changing the algorithms into coding using the programming language
Testing/Debugging (cont’d)

- The process of ensuring a program is free of errors
- Test the program to correct syntax and logic error
- Involves testing the completed program to verify that it works as desired – produce expected output
- The testing process will involve the using of real data and fictitious data
Errors :

- Syntax Error
  - A programming error that occurs when the programmer has not followed the rules of the programming language
  - It happens during compilation
  - Problems with grammar, spelling or punctuation
  - E.g. left off semi colon, misspelled a reserve word
  - Easiest to find because program itself helps to find
Errors:
- Logic error
  - A programming error that occurs when running a program produces incorrect result
  - More difficult to find
  - No compiler will stop and tell that program have a logic error
  - E.g. \((2 + 3) \times 5 = 25\) but, \(2 + 3 \times 5 = 17\)
Errors

- Run time error
  - Occurs during the execution of a program and are detected by the compiler.
  - E.g.
    - Use some code in the library but was not in your computer for some reason – would get a run time message stating that program could not run
    - Program that tires to divide some number by zero – illegal operation
  - Debugging process
    - Debugger is available in C++ for detecting errors while a program is being executed.
Testing/Debugging (cont’d)

- Testing
  - Alpha testing
    - an internal onsite test
  - Beta testing
    - One or more rounds of outside test
    - Allow the program to be tested by a wide variety of individuals or companies on a wide variety of hardware
Maintenance

- Document
  - To put together all materials that have been generated throughout PDLC
  - Flowchart, messages, algorithms, lines of code, user manual
  - Internal documentation
    - Used by other programmers to help them know a certain way or how the program is written
  - External documentation
    - User manuals, FAQs, Help
Update program/software to correct errors, improve usability, standardize and adjust to organizational changes.

Involves modifying the program to remove previously undetected errors and to keep up to date as government regulations or company policies change.

Maintaining until program become redundant or too old – maintenance stop and PDLC start all over again.
Quick Quiz

- List the 5 steps in PDLC.
- List and define the error types in developing a program.
- What are the symbols used in flowchart? Define each of the symbol.
- What is the purpose of having flowchart or pseudocode in developing a program? What are the differences between flowchart and pseudocode?
Exercises

- Design a program that receives two integer inputs from the user, adds the two integer number and displays the result.
  - Draw the IPO table
Design a program that converts pounds into kilograms. The program prompts the user to enter a number in pounds, converts it to kilograms and displays the result. One pound is 0.454 kilograms.
- Draw the IPO table
Exercises

- Design a program that allows the user to enter a value that represents the radius of a circle. The program calculates the diameter and then calculates the circumference. The program prints both the diameter and the circumference.
  - List the input, process and output in the IPO table.
Exercises

- Design a flowchart and a pseudocode to determine whether the applicant is qualified to apply for a credit card or not based on his salary. If the salary is greater than RM1500, then display a message that applicant entitles to apply. Otherwise, display a message that the applicant cannot apply for the credit card.
ABC Portrait Studio calculates charge for its customers based on the number of subjects posing for the portrait. Design a flowchart and pseudocode that accepts the number of subjects in portrait. Then, display the number of subjects and calculated fee based on the fee schedule as follows:

<table>
<thead>
<tr>
<th>Subjects in Portrait</th>
<th>Price per Subjects (RM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 3</td>
<td>20</td>
</tr>
<tr>
<td>4 – 6</td>
<td>30</td>
</tr>
<tr>
<td>&gt; 7</td>
<td>40</td>
</tr>
</tbody>
</table>
References

- Norizan Mohamad & Mazidah Putih, Problem Solving with C++, University Publication Centre (UPENA), UiTM, 2006