CPE 100 COMPUTER PROGRAMMING FOR ENGINEERING

COURSE PREPARATION

Assoc. Prof. Dr. Natasha Dejdumrong

Department of Computer Engineering

King Mongkut's University of Technology Thonburi

CONTENTS

MODULE III

LESSON 11: POINTERS

I. Introduction
II. What is a Pointer?
III. How to Initialize a Pointer?
IV. How to Dereference a Pointer?
V. Pointer Arithmetic
VI. Pointer to Pointer
VII. Summary
VIII. Model Questions

LESSON 12: FUNCTIONS

I. Introduction
II. What are Functions?
III. How Functions Work?
IV. Function Declaration
V. Function Definition
VI. Summary
VII. Model Questions

LESSON 13: FUNCTION PARAMETERS

I. Introduction
II. return Statement
III. Formal Parameters
IV. Actual Parameters
V. Array as a Parameter
VI. Structure as a Parameter
VII. Union as a Parameter
VIII. Summary
IX. Model Questions

LESSON 14: PASS BY VALUE/ADDRESS

I. Introduction
II. Pass-by-value
III. Pass-by-address
IV. Some Examples

MODULE IV

LESSON 16 INTRODUCTION TO HEADER FILES

I. Introduction
II. Libraries and Header Files
III. Standard C Library & Header Files
IV. Adding Functions to Library
V. Creating User-defined Library
VI. Summary
VII. Model Questions

LESSON 17 C PRE-PROCESSOR

I. Introduction
II. What is a C Pre-Processor?
III. Syntax of a C Pre-Processor
IV. Advantages of a C Pre-Processor
V. File Inclusion Directive
VI. Conditional Compilation Directives
VII. Summary
VIII. Model Questions

LESSON 18 MACROS

I. Introduction
II. Macro & Macro Substitution
III. Simple Macro Substitution
IV. Macros with Arguments
V. Nesting of Macros
VI. Undefining a Macro
VII. Summary
VIII. Model Questions

LESSON 19 FILE PROCESSING IN C

I. Introduction

MODULE I: INTRODUCTION TO C PROGRAMMING LANGUAGE

LESSON 1 **INTRODUCTION TO C PROGRAMMING LANGUAGE**

STRUCTURE

I. INTRODUCTION

In order to communicate any idea, thought, instruction or information, humans make use of spoken language. The fact is that you have just understood the very first sentence of this chapter all due to that. However, spoken languages are not understood by machines. Not only machines but also other living creatures of this planet do not understand spoken languages. Hence, they need some different kind of a language. As an example, a ring master in circus uses the movement and sound of a whip to control the beast. Similarly, machines like cars understand mechanical motions such as the movement of steering wheel, brake pedal and so on, to perform specific actions. This means whenever some instruction is to be communicated to any living or non-living thing, we need some kind of a specific language that is understood by it. As such, computer is not an exception.

In order to communicate instructions to a computer, humans need a language which we call a programming language. Like we have a variety of spoken languages, we also have a variety of programming languages. However, computers nowadays are digital and only understand the language of 0's and 1's. Therefore, to instruct a computer to perform some specific task we have to put these 0's and 1's in a particular sequence. This kind of a programming language is called Machine Language. A set of these instructions is called a Program like a set of dialogs in a spoken language is called a Conversation. Communicating instructions to a digital computer in machine language is difficult, error prone, non-portable and so on. To overcome some of its drawbacks (specifically difficulty) an assembly language can be used which actually uses mnemonics for these strings of 1's and 0's. Indeed, assembly language is not understood directly by a computer and hence a translator (which is called assembler) is required to convert these mnemonics into binary strings. A translator converts a source program written in some programming language into executable machine code. Machine and Assembly language, both called low-level programming languages, are highly machine dependent, error-prone, difficult to understand and learn, and so on. A high-level programming language is machine independent, less error-prone, easy

to understand and learn, and so on. In this language, the statements (or instructions) are written using English words and a set of familiar mathematical symbols which makes it easier to understand and learn, and is thus less error-prone. Furthermore, the language is machine independent and hence can be ported to other machines. The high-level programming languages can be further classified into procedural, non-procedural and problem-oriented languages. This text discusses the vocabulary, grammatical rules and technical aspects of a high-level procedural language called C. It must be noted that all high-level languages also need a translator to convert it into machine language. Depending upon whether the translation is made to actual machine code or some intermediate code, the translator is called a Compiler or an Interpreter respectively. C programming language uses a compiler to convert the instructions into actual machine code. Some researchers also classify C language as a middle level language. The reason behind this is that assembly code can also be mixed with C code.

This lesson will look into the motivation behind learning C language, unveil its history of development, highlight its features, and point out advantages and disadvantages of C programming language.

II. WHY LEARN C?

One may argue that if there is plethora of programming languages available, then what makes C language so special? There are two answers to this question. First, C language has been used by programmers for past 30-40 years to develop every kind of utility. This means the language is well understood, the issues with the language have been completed eradicated, a lot of the principles used in C show up in a lot of other languages, and so on. Second, C combines portability across various computer architectures as provided by any other high-level language while retaining most of the control of the hardware as provided by assembly language. It is a stable and mature language whose features are unlikely to disappear for a long time

III. HISTORY OF C LANGUAGE

In 1965, Bell Telephone Laboratories, General Electric Company and Massachusetts Institute of Technology where working together to develop a new operating system called MULTICS. In 1969, Bell Laboratories ended its participation in the project. However, the participating members of Bell Labs, mainly Ken Thompson and Dennis Ritchie, still had an intention to create such kind of an OS which finally matured into UNIX operating system.

After it early success in 1970, Ken Thompson set out to implement a FORTRAN compiler for the new system, but instead came up with the language B. B Language was influenced by Richard Martins BCPL language which was type-less and interpretive. Hence, both BCPL and B language had some performance drawbacks.

In 1972, in an effort to add “types” and definition of data structures to B language and use compiler instead of interpreter, Dennis Ritchie came up with a new B language called C language.

Just like natural languages, programming languages also change. The original specification of the C language (as devised by Dennis Richie) together with close variations is sometimes known as Old-style C or Traditional C. However, in 1988 ANSI (American National Standards Institute) published a new specification of the language

which has become an international standard which is accepted by all compilers. It is known as ANSI C. ANSI C is mostly a superset (i.e. provides additional functionality) over Old-style C.

IV. FEATURES OF C LANGUAGE

C language has a rich set of features. These include:

1) There are a small, fixed number of keywords, including a full set of control flow primitives. This means there is not much vocabulary to learn.
2) C is a powerful, flexible language that provides fast program execution and imposes few constraints on the programmer.
3) It allows low level access to information and commands while still retaining the portability and syntax of a high-level language. These qualities make it a useful language for both systems programming and general-purpose programs.
4) Another strong point of C is its use of modularity. Sections of code can be stored in libraries for re-use in future programs.
5) There are a large number of arithmetical, relational, bitwise and logical operators.
6) All data has a type, but implicit conversions can be performed. User-defined and compound data types are possible.
7) Complex functionality such as I/O, string manipulation, and mathematical functions are consistently delegated to library routines.
8) C is a Structured Compiled language widely used in the development of system software.

V. ADVANTAGES OF C LANGUAGE

C language has in-numerous advantages. The important advantages of C language are described

as follows:

1) Easy to Understand: C language is a structured programming language. The program written in C language is easy to understand and modify.
2) Middle Level Language: C language has combined features of low-level language (such as assembly language) and some of high-level language. It is, therefore, sometimes C language is also referred to as middle level language. Most of the problems that can be solved using assembly language can also be solved in C language.
3) Machine Independent: Program written in C language is machine independent. It means that a program written on

one type of Computer system can be executed on another type of Computer.

4) Built in Functions: C language has a large number of built in functions. The programmer uses most of these built in functions for writing source program, instead of writing its own code.
5) Hardware Control: Like assembly language, the programmer can write programs in C to directly access or control the hardware components of the computer system.
6) Easy to learn and use: C language is easy to learn and to write program as compared to low level languages such as assembly language.
7) Basis for C++: Today, the most popular programming is C+. C is the basis for C++. The program structure of C is similar to C++. The statements, commands (or functions) and methodologies used in C are also available in C++. Thus, learning C is a first step towards learning C++.
8) Modularity: C is a structured programming language. The programmer can divide the logic of program into smaller units or modules. These modules can be written and translated independently. Including

VI. DISADVANTAGES OF C LANGUAGE

Though C language has rich set of features and have enormous advantages, however it suffers from some disadvantages. These include:

1) C does not have OOPS feature.
2) There is no runtime checking in C language.
3) There is no strict type checking.
4) C doesn't have the concept of namespace.
5) C imposes few constraints on the programmer. The main area this shows up is in C's lack of type checking. This can be a powerful advantage to an experienced programmer but a dangerous disadvantage to a novice.

VII. FLOWCHART

A diagram represents the workflow or process which helps to solve a task, step by step. A flowchart is a pictorial representation depicting the flow of steps in a program, people in an organization, or pages in a presentation.

Flowchart symbols are universally recognized across multiple disciplines, and will allow an instructional designer to communicate a great deal of information accurately, efficiently, and succinctly.

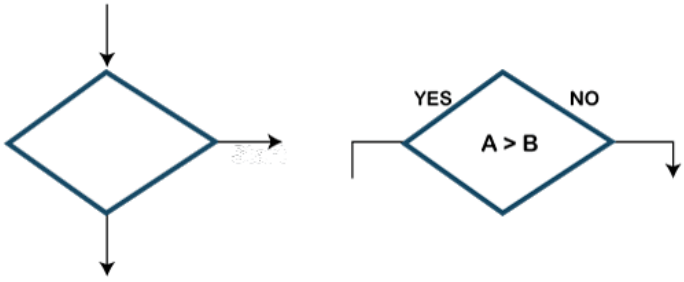
Flowchart symbols:

The various symbols used in Flowchart Designs are given below.

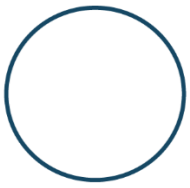
Terminal Symbol: In the flowchart, it is represented with the help of a circle for denoting the start and stop symbol. The symbol given below is used to represent the terminal symbol.

Input/output Symbol: The input symbol is used to represent the input data, and the output symbol is used to display the output operation. The symbol given below is used for representing the Input/output symbol.

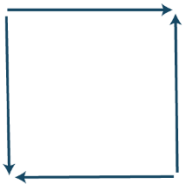
Processing Symbol: It is represented in a flowchart with the help of a rectangle box used to represent the arithmetic and data movement instructions. The symbol given below is used to represent the processing symbol.



Decision Symbol: Diamond symbol is used for represents decision-making statements. The symbol given below is used to represent the decision symbol.

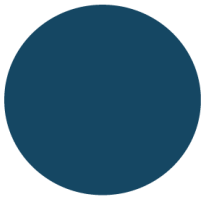


Connector Symbol: The connector symbol is used if flows discontinued at some point and continued again at another place. The following symbol is the representation of the connector symbol.



Flow lines: It represents the exact sequence in which instructions are executed. Arrows are used to represent the flow lines in a flowchart. The symbol given below is used for representing the flow lines.

Hexagon symbol (Flat): It is used to create a preparation box containing the loop setting statement. The symbol given below is used for representing the Hexagon symbol.



On-Page Reference Symbol: This symbol contains a letter inside that indicates the flow continues on a matching symbol containing the same letters somewhere else on the same page. The symbol given below is used for representing the on-page reference symbol.

Off-Page Reference: This symbol contains a letter inside indicating that the flow continues on a matching symbol containing the same letter somewhere else on a different page. The symbol given below is used to represent the off-page reference symbol.

Delay or Bottleneck: This symbol is used for identifying a delay in a flowchart. The alternative name used for the delay is the bottleneck.

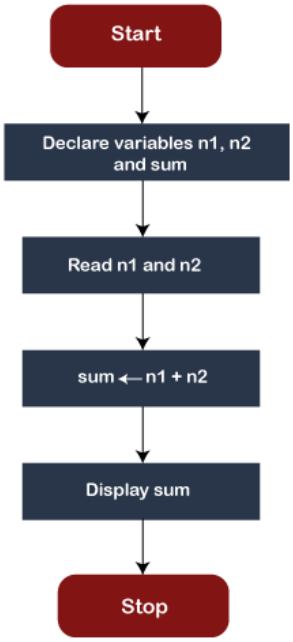
The symbol given below is used to represent the delay or bottleneck symbol.

Document Symbol: This symbol is used in a flowchart to indicate a document or report. The symbol given below is used to represent the document symbol.

Internal storage symbol: The symbol given below is used to represent the internal storage symbol.

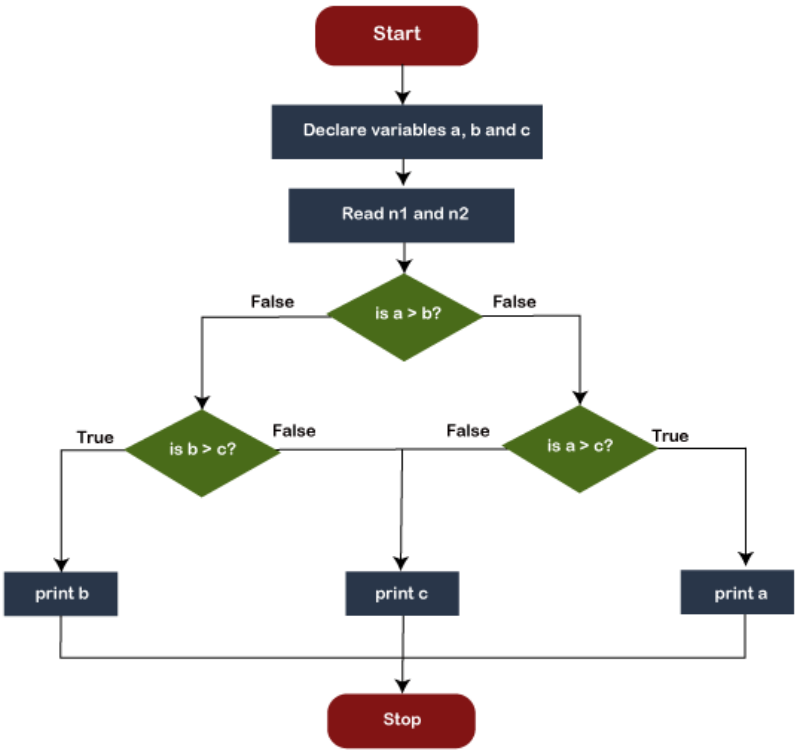
Example 1:

Design a flowchart for adding two numbers entered by the user.



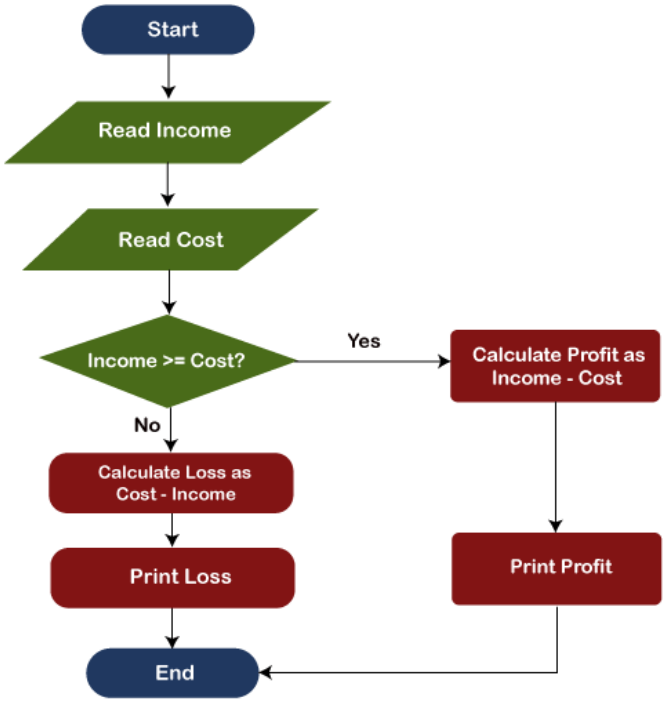
Example 2:

Design a flowchart for finding the largest among three numbers entered by the user.



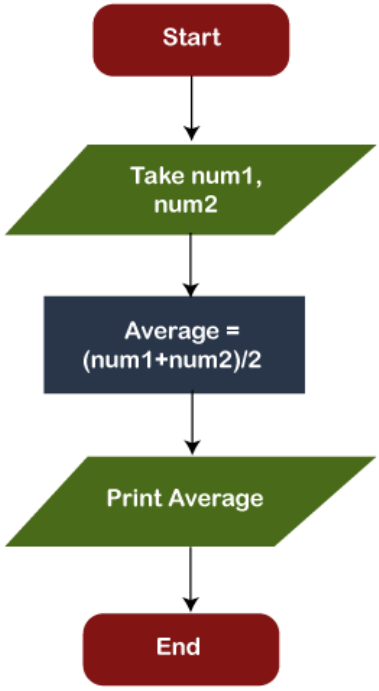
Example 3:

Design a flowchart for calculating the profit and loss according to the value entered by the user.



Example 4:

Draw a flowchart to calculate the average of two numbers.



VIII. PSEUDOCODE

Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.

The rules of Pseudocode are reasonably straightforward. All statements showing "dependency" are to be indented. These include while, do, for, if, switch. Examples below will illustrate this notion.

Examples 1:

If student's grade is greater than or equal to 60

Print "passed"

else

Print "failed"

Examples 2:

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.

IX. SUMMARY

- Programming languages are means to instruct a computer to perform some desired task.
- A program is a set of instructions which are converted into equivalent machine instructions understood by machine.
- All programming languages require a translator to convert the program into machine code.
- C language is a structured compiled programming language which is widely used in the development of application and system software.
- C language was developed by Dennis Ritchie in order to port UNIX across various platforms.
- C language has rich set of features, innumerous advantages and few disadvantages.
- A flowchart is a pictorial representation depicting the flow of steps in a program, people in an organization, or pages in a presentation.

X. MODEL QUESTIONS

A. Write a short note on classification of programming languages.

B. Discuss history of C programming C language.

C. Explain features of C programming language.

D. Discuss advantages and dis-advantages of C programming language.

E. Write flowchart

F. Write pseudocode