

C Programming (CSC 110)

B.Sc. CSIT First Semester

Nature of course: Theory + Lab

Full Marks: 100

Theory: 60 (Semester Exam)

Internal Assessment: 20 (Internal Exam)

**Practical Exam: 20 (Lab Report/Lab Exam/Viva
by External Examiner)**

**- Hemanta GC
Patan Multiple Campus**

Unit 1. Problem Solving with Computer

- 1.1 Problem analysis (*requirement analysis, program design, program coding, program testing, software installation and maintenance*)
- 1.2 Algorithms and Flowchart (*symbols start/stop, read/print, processing statement, condition check, direction of flow, connectors*)
- 1.3 Coding, Compilation and Execution (*compiler, integrated development environment, compiling and linking*)
- 1.4 History of C,
- 1.5 Structure of C program (*preprocessor directive, #include and #define directives, header files and library files*)
- 1.6 Debugging, Testing and Documentation (*compiler error, linker error, and run-time error*)

Problem Solving with Computer

- A computer is a tool to solve a problem.
- Problem solving is the process of transforming the description of a problem into a solution by using our knowledge of the problem domain and by relying on our ability to select and use appropriate problem-solving strategies, techniques and tools.
- Programming is a problem solving activity. When we write a program, we are actually writing instructions for the computer to solve something for us.
- Using a computer as a problem solving tool following steps are involved

Steps involved in Problem Solving with a Computer

- **Problem Definition**
- **Problem Analysis**
- **Program Design**
- **Coding**
- **Compilation and Execution**
- **Debugging and Testing**
- **Documentation**

Problem Definition

- A clearly defined problem is already half the solution.
- Computer programming requires us to define the problem first before we even try to create a solution.
- Based on the clear definition, one can proceed to design and write the program to solve any problem.

Problem Analysis

- **Analyzing the problem require us to identify the following:**
 - Input(s) to the problem, their form and the input media to be used
 - Output(s) expected from the problem, their form and the output media to be used
 - Special constraints or conditions (if any)
 - Any formulas or equations to be used

For example, Compute the average marks obtained by students in "C- Programming".

Inputs: Marks of individual students

Outputs: The average mark of students

Formula: Average = Total Marks / No of students

i.e. $A = T/N$

Program Design

This process involve in preparing the algorithms to the problem.

- **Algorithm** :An algorithm is a clear and unambiguous specification of the steps needed to solve a problem.
- **Algorithm may be expressed as in either form,**
 1. *Human language (e.g. English)*
 2. *Pseudocode*
 3. *Graphical Representation (Flow-Chart)*

Algorithm Representation: Pseudocode

Pseudocode - is a cross between human language and a programming language.

- ***A pseudocode is:***
 - An artificial and informal language that helps programmers to develop algorithms
 - Not an actual programming language
 - Not actually executed on computers
- ***Why do we need Pseudocode?***
 - Helps to “think out” a program before writing it
 - May be converted easily into a corresponding programming language code.

Algorithm Example

Problem: *Compute the average marks of students obtained in a TEST of "Computer Programming"*




- *Human Language Specification: (In English)*
 1. Read the no of students
 2. Read the marks of each students
 3. Sum the marks of all students as total marks.
 4. Divide the total marks by number of students.
 5. Report the result of division as average marks of students.
- *Pseudocode Specification:*
 1. Input N
 2. Input N marks: $m_1, m_2, m_3, \dots, m_N$
 3. $T = (m_1 + m_2 + m_3 + \dots + m_N)$
 4. $A = T/N$
 5. Output A

Flow Chart


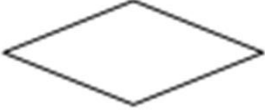


A flow-chart is:

- the graphical representation of the algorithm for any problem.
- useful for developing and representing algorithms.
- is drawn using certain special-purpose symbols connected by arrows called flowlines.
- The symbols indicate the actions to be performed, and flowlines indicate the order in which actions are to be performed

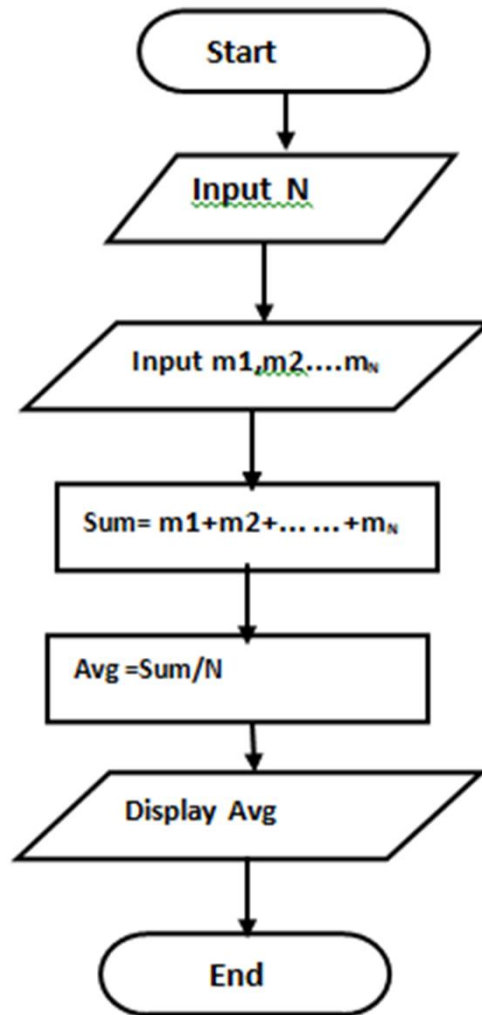
Basic Flow-Chart Symbols

Symbol	Description
	Oval symbol (or <u>termination symbol</u>) indicates the beginning or end of a program, or a section of code
	Rectangle symbol (or <u>action symbol</u>) indicates any type of processing, calculation
	Subprocess symbol used for pre-defined process(sub-routines)

Flow Chart Symbols

	Parallelogram symbol For input/output operation
	Diamond symbol (or <u>decision symbol</u>) indicates that a decision is to be made
	Flowline connects one symbol to another
	Entry or exit points are used to attach the one flowchart to another

Flow Chart: Example

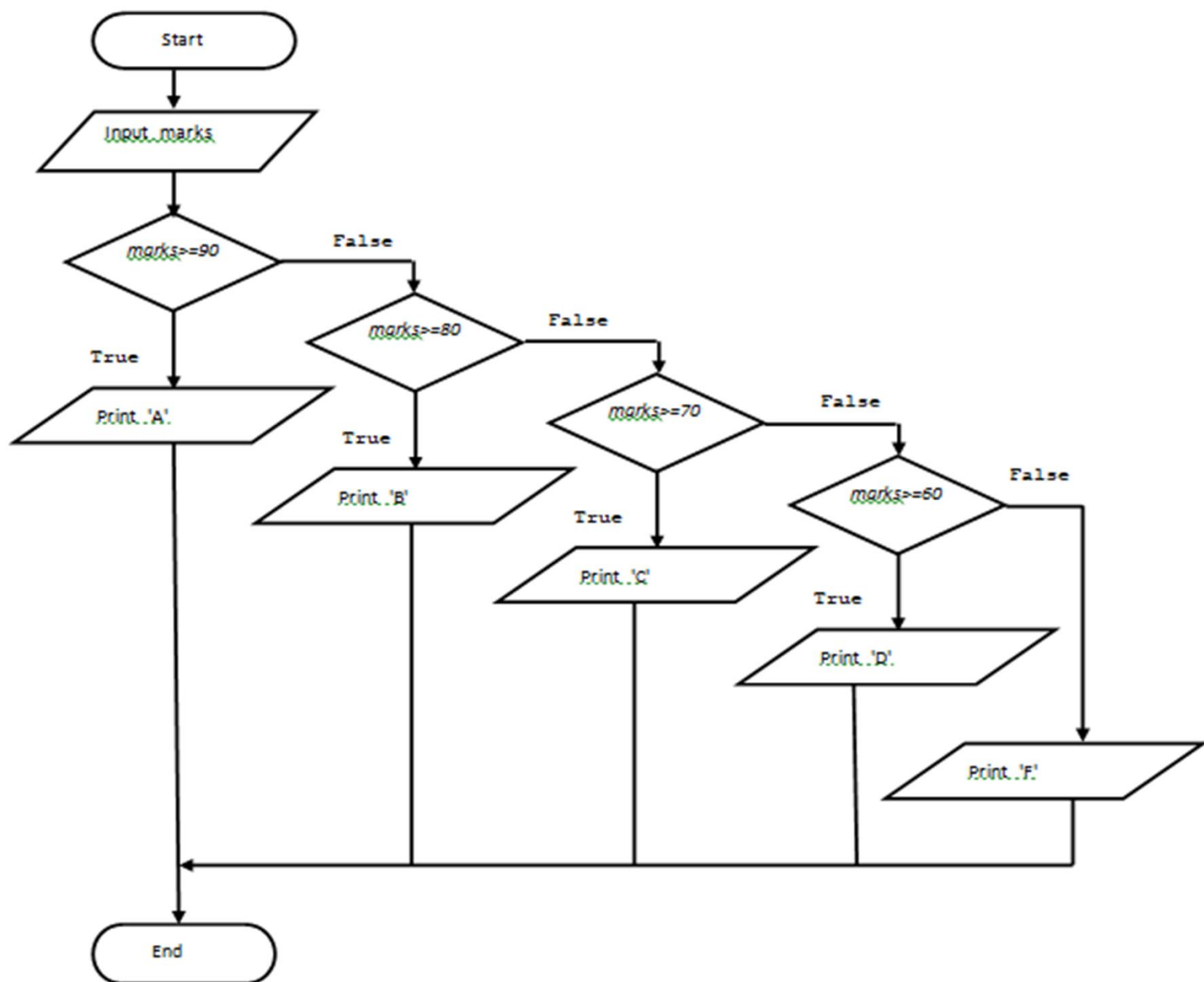


**Flow chart to solve the problem
to find the average marks of students
in a Test**

Example: Pseudocode and Flow chart

Problem : Print the grade of student based on the input score

- Input the marks of student
- if mark is greater than or equal to 90
 print "A"
- else
 if student's mark is greater than or equal to 80
 print "B"
- else
 if student's mark is greater than or equal to 70
 print "C"
- else
 if student's mark is greater than or equal to 60
 print "D"
- else
 print "F"



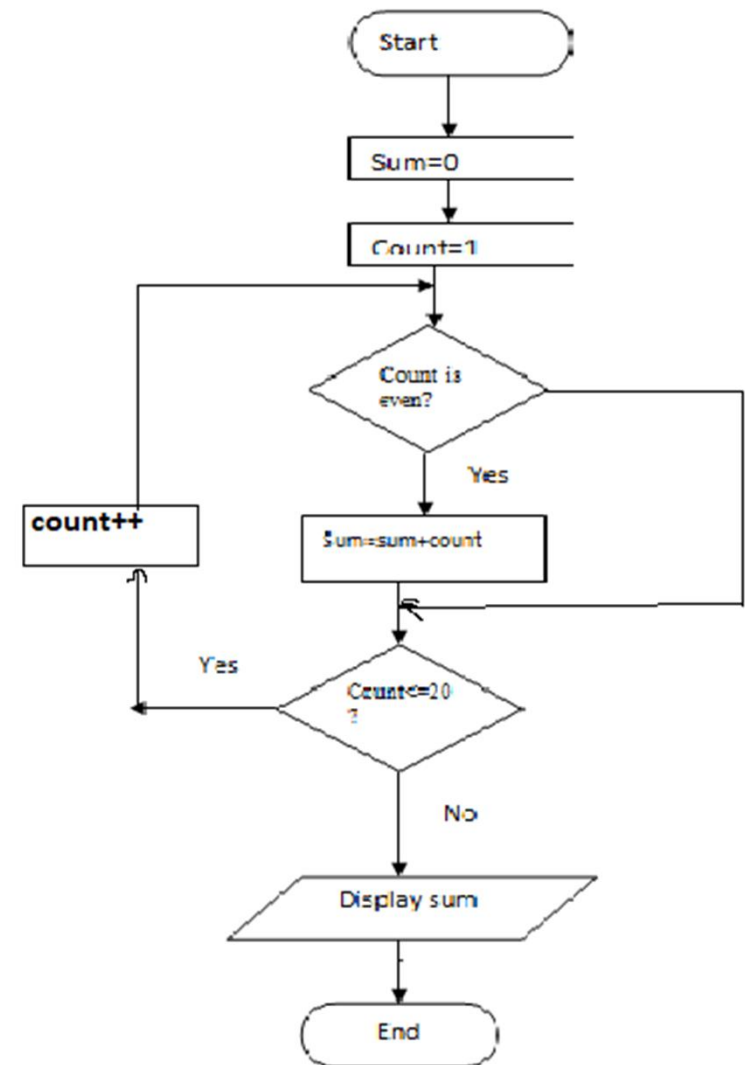
Example: Continued.....

- The algorithm sums all the even numbers between 1 and 20 inclusive and then displays the sum. It uses a repeat loop and contains a null else within the repeat loop.

The pseudocode:

```
1. sum = 0
2. count = 1
3. REPEAT
  IF count is even THEN
    sum = sum + count
  count = count + 1
  UNTIL count > 20
4. DISPLAY sum
```

Flow chart



Coding

- Coding is the real job of programmer.
- The algorithm to solve a problem which is described by pseudo-code or flow chart is converted into actual programming language code.
- The code written by programmer by using any programming language like C is called the source code or source program.

Compiling/Linking

- ***Compiler:*** A program that translates a program written in source language (SourceCode) into the equivalent target code.
- ***Linker:*** A program that links the compiler generated intermediate code with required necessary library files.
- ***Loader:*** A loader is the program that is used to load the executable program instructions into memory for execution
- ***Integrated Development Environment(IDE):*** The IDE is the program that allows to write source code , compile the source code as well as link and execute the program from the single interface.

Compilation and execution

- The source code written in any programming language is not directly executed by the computer.
- It should be translated into to the machine readable format i.e. actual machine language. The process of translation of source code into the target code is called the compilation.
- Each programming language has its own compiler program that translates the source code into its target code.
- The converted program in actual machine language is then executed by the computer which is known as program execution.

Debugging and Testing

- A written program may have errors, some errors can be detected by the language compilers and some errors can not be identified by the compiler and occurred during the program execution.
- **Common types of errors are:**
 - ***Syntax Errors:*** Identified by compiler at the program compilation time, easily detected and corrected.
 - ***Logical Errors:*** Not identified by the compiler at compile time and identified at the execution time. e.g. misuse of operators. Detected during program execution
 - ***Linker Errors:*** Errors detected during linking the compiled code with necessary libraries. Detected by Linker.
 - ***Runtime Errors:*** Detected during program executions. These are basically logical errors and also the errors related to memory allocations.

Debugging and Testing

- Testing is the process of checking the program for its correct functionality by executing the program with some input data set and observing the output of the program.
- **Testing may be:**
 - ***Black Box Testing***: Just give input test data and view the output which is correct or not as expected(user level testing)
 - ***White Box Testing***: Detail debugging of code for testing the correctness of the program(Programmar level testing)

Documentation

Documentation is the task of keeping all the references of the related program for future references.

- From the start of the problem solving to the end of the implementation of the program, all the tasks should be documented i.e. kept for future references.
- Documentation is also the important part of the problem solving or program development.
- **Documentation may be of two types**
 - ***Technical Documentation***: known as ***programmer's documentations*** which includes the problem analysis to implementation details for that program. It is needed for future reference for any modification, update of the program.(Internal comments and External reference document)
 - ***User manual***: is the documentation prepared for the end-user of the program that guides the user how to operate the program.

Programming Languages

- ***High Level Language:*** Easy understandable by human and closer to human language, and mostly used for writing the programs e.g. C, C++, JAVA, FORTRAN etc.
- ***Assembly Language:*** An **assembly language** is a low-level programming **language** designed for a specific type of processor. It may be produced by compiling source **code** from a high-level programming **language** (such as C/C++) but can also be written from scratch.
- ***Machine Language:*** Machine language, or **machine code**, is a low-level **language** comprised of binary digits (ones and zeros). It is understandable by machine processor.

History of C

- C is a general-purpose computer programming language developed between 1969 and 1973 by [Dennis Ritchie](#) at the [Bell Telephone Laboratories](#) for use with the [Unix operating system](#).
- According to Ritchie, the most creative period occurred in 1972.
- It was named "C" because its features were derived from an earlier language called “B” at Bell Lab.
- Although C was designed for implementing **system software**, it is also widely used for developing portable application software.
- In 1978, Brian Kernighan and Dennis Ritchie published the first edition of ***The C Programming Language***.
- The specification of C language referred in this book was called by C programmer as “K&R”, and followed for long times.
- The second edition of the book covers the later [ANSI C](#) standard.

Structure of C program

The general form of a C program is as follows :

- Preprocessor directives - Starting with symbol #
- Global declarations of user functions
- Global variables declarations
- main()
- {
 - local variables to function main ;
 - statements associated with function main ;
- }
- definitions of functions like
 - function1()
 - {
 - local variables to function 1 ;
 - statements associated with function 1 ;
 - }
 - function2()
 - {
 - local variables to function f2 ;
 - statements associated with function 2 ;
 - }
- ...etc

Thank You