CPNM Lecture 1 - Introduction

Mridul Sankar Barik

Jadavpur University

2022

CPNM - Syllabus I

1. **C Programming Introduction**: History of Computing, Evolution of Programming Languages, Compilers, Interpreter, Algorithms and Flowcharts, Structure of a C Program **[2L]**
2. **Expressions**: Basic Data Types, Variables, Type Qualifiers, Variable Scopes, Constants, Operators, Operator Precedence, Expression Evaluation, Type Conversion in Expressions, Type Casting **[2L]**
3. **Console I/O**: Reading and Writing different data types **[1L]**
4. **Statements**: Selection Statements (if, switch-case), Iteration Statements (for loop, while loop, do-while loop), Jump Statements (return, goto, break, exit, continue) **[2L]**
5. **Arrays and Strings**: Single Dimension Arrays, Double Dimension Arrays, Strings, Arrays of Strings, String Library Functions **[3L]**
6. **Functions**: General Form, Function Prototypes, Parameter Passing Mechanisms, Command Line Arguments, Recursion **[3L]**

CPNM - Syllabus II

7. **Pointers**: Pointer Variables, Pointer Operators, Pointer Expressions, Pointers and Arrays, Functions and Pointers, Pointers to Functions, Dynamic Memory Allocation **[4L]**
8. **Structures, Unions**: Structures, Arrays of Structures, Structure Pointers, Unions **[2L]**
9. **File I/O**: Data Organization, File Operations, Text Files and Binary Files, Random Access **[3L]**

Numerical Methods

1. **Approximations and Errors Associated with Numerical Methods [1L]**
2. **Solution of Non-Linear Equations**: Bisection method, Method of False Position, Newton-Raphson Method **[3L]**
3. **Solution of Linear Simultaneous Equations**: Direct methods: Gauss-Jordan Elimination, Matrix Inversion using Gauss-Jordan Elimination; Iterative methods: Jacobi's Method **[4L]**

CPNM - Syllabus III

4. **Methods for Interpolation**: Newton's Forward Difference Formula, Newton's Backward Difference Formula, Lagrange's formula **[3L]**
5. **Curve fitting**: Method of Least Squared Error **[2L]**
6. **Methods for Differentiation and Integration**: Computation of Derivatives using Newton's Forward/Backward Difference Formulae. Trapezoidal Method, Simpson's Method. **[2L]**
7. **Solution of Differential Equations**: Euler's Method, Modified Euler's Method, Runge-Kutta 2^{nd} and 4^{th} Order Formulae **[3L]**

Text Books I

1. The C Programming Language, by Brian W. Kernighan and Dennis M. Ritchie, Second Edition, Pearson
2. C: The Complete Reference by Herbert Schildt, Fourth Edition, 2017, McGraw Hill Education.
3. Programming With C by Byron Gottfried, Fourth Edition, 2018, McGraw Hill Education.
4. Programming in ANSI C by E Balagurusamy, Seventh Edition, McGraw Hill Education
5. C Primer Plus by Stephen Prata, 5th Edition, SAMS Publishing, 2005
6. C: A Reference Manual by Samuel P. Harbison and Guy L. Steele, 5th Edition, Prentice Hall, 2003
7. C Traps and Pitfalls by Andrew Koenig, Addison Wesley Professional, 1989

Text Books II

8. C Programming: A Modern Approach by K. N. King, 2nd Edition, W. W. Norton and Company, 2008
9. Computer Oriented Numerical Methods by V. Rajaraman, Fourth Edition, 2018, PHI Learning.
10. Introductory Methods of Numerical Analysis by S.S.Sastry, Fifth Edition, 2012, PHI.
11. Numerical Methods for Engineers by S. C. Chapra and R. P. Canale, 7th Edition, 2016, McGraw Hill Education.
12. Numerical Methods by J.H.Mathews, PHI
13. Numerical Analysis and Algorithms by P. Niyogi, TMH
14. Numerical Methods for scientific and engineering computations by Jain, Iyengar and Jain, New Age International publisher
15. Computer Systems and Data Analysis by D.K.Basu, M.Nasipuri and M.Kundu, Narosa Publishers.

Text Books III

16. Numerical Methods by Sukhendu Dey and Shishir Gupta, McGraw Hill Education, First Edition, 2013.
17. Numerical Mathematical Analysis, by James B. Scarborough, 2nd Edition, Baltimore: Johns Hopkins Press, 1950.
18. Many free books and lots of material on C coding practices available on Web.

Evolution of Programming Languages

- 1940's: Machine language
- 1950's: Assembly language
- 1960's: High level language, ex: FORTRAN, BASIC, COBOL
- 1970's: System programming language, ex: C
- 1980's: Object oriented language, ex: C++
- 1990's: Scripting, Web, Component Based, ex: Java, Perl, Python, Visual Basic, Java Script
- 2000's: C#, Clojure, Go, and many domain specific languages

Compilers

- A program that converts a program written in one language (source language) into an equivalent program in another language (target language).
- Example: C, C++
- Advantages
  - Compile once, run target many times
- Disadvantage
  - Debugging requires much more software support

Interpreters

- Reads and executes source program one line at a time
- An interpreted program runs slower than a compiled program
- Example - Java source is first compiled to byte code and then it is interpreted by the Java Runtime Environment (JRE)

Brief History of C

- Many ideas of C originated from the language BCPL, developed by Martin Richards
- Language B was written by Ken Thompson in 1970 at AT&T Bell Labs
- In 1972 Dennis Ritchie at Bell Labs writes C language
- In 1978 the book "The C Programming Language" by Kernighan and Ritchie was published
- In 1983, the American National Standards Institute (ANSI) established a committee to provide a modern, comprehensive definition of C
- The ANSI standard, or "ANSI C", was completed in late 1988
- A new ANSI C standard with significant changes came out in 1999

Structure of a C Program

```
#include<stdio.h>
int main(void){
  printf("Hello World\n");
    return(0);
}
```

- `#include<stdio.h>`
  $\Rightarrow$ includes information about C's standard I/O library
- Program executable code goes inside `main`
- `printf` is a function from C's standard I/O library, for producing formatted output
- `\n` tells `printf` to advance to the next line after printing the message
- `return 0` $\Rightarrow$ program returns the value 0 to the OS when it terminates

Stages in C Program Compilation

- **Preprocessing**: Preprocessor processes commands that start with # known as preprocessor directives. It adds things or modifies the code.
- **Compiling**: Translates C source program into assembly code
- **Assembling**: Translates assembly code to object code (machine instructions)
- **Linking**: Linker combines generated object code with additional code (i.e. library functions that are used in the program) to yield an executable code

Running C Programs

- `gcc` is one of the most popular C compilers
- It is similar to Unix `cc` compiler
- `gcc` does everything: preprocessing, compilation, assembly and linking
- `-c` option ⇒ Compile or assemble the source files, but do not link
- `-S` option ⇒ Stop after the stage of compilation proper; do not assemble.
- `-E` option ⇒ Stop after the preprocessing stage; do not run the compiler proper.
- `-o file` ⇒ Place output in file file. If `-o` is not specified, the default is to put an executable file in `a.out`.
- `gcc test.c` ⇒ produces `a.out` as the target executable
- `gcc test.c -o test` ⇒ produces `test` as the target executable

Functions in C

- Functions are building blocks of C programs, which is a little more than a collection of functions
- A function is a series of statements that have been grouped together and given a name
- Some functions compute a value and uses `return` statement to specify what it computes
- Two types of functions: user defined and library
- The `main` function gets called automatically when the program is executed

C Statements

- Statements are command executed when the program runs
- Statements are terminated by a semicolon
- Preprocessor directives do not end with a semicolon
- Comments are included within /* and */

Variables I

- Variables store data temporarily during program execution
- Every variable must have a `type` that tells the kind of data it stores
- A variable of type `int` stores a whole number
- A variable of type `float` stores a number with fraction. Value of a `float` variable is an approximation of the number that was stored in it.
- Variables must be declared before they are used

```
int height;
float profit;
```

- Declaration of several variables of the same type can be combined

```
int height, length, width, volume;
float profit, loss;
```

Variables II

- C99 standard doesn't require declarations to come before statements
- In C++, Java it is a common practice not to declare a variable until they are first needed
- A variable can be given a value via assignments

```
height=8;
length=12;
width=10;
profit=100.20;
```

- The numbers 8, 12, 10, 100.20 are constants
- Mixing types (i.e. assigning a `float` constant to an `int` variable etc.) in assignments is possible but is not safe
- Most variables are not initialized when declared

Reading/Printing Input/Output

- Reading: `scanf` statement of C standard I/O library

```
scanf("%d", &i); /* reads an integer; stores into i*/
scanf("%f", &x); /* reads a flaot; stores into x*/
```

- Writing: `printf` statement of C standard I/O library

```
printf("%d", i); /* prints an integer*/
printf("%f", x); /* prints a float*/
```

Basic Data Types I

- Numeric: integer and floating point
  - Value of an integer type are whole numbers, while values of a floating type call have a fractional part
- Character: Used to store a single character

A Simple Program

```
#include<stdio.h>

int main(void){

    int height, length, width, volume, weight;
    int density = 3;

    printf("Enter height of box: ");
    scanf("%d", &height);
    printf("Enter length of box: ");
    scanf("%d", &length);
    printf("Enter width of box: ");
    scanf("%d", &width);

    volume = height * length * width;
    weight = volume * density;

    printf("Volume: %d\n", volume);
    printf("Weight: %d\n", weight);

    return(0);
}
```