Data Structures and Program Design in C

Topic 1 : Course Administration and Introduction

Dr Manjusri Wickramasinghe

Outline

- Course Administration
- What is meant by Computers, Programming Languages, and Programming?
- The C Programming Language
  - The first C program

Course Administration …(1)

- Course Rubric
  - Continuous Assessment – 40%
  - Final Examination – 60%

- Final Examination
  - Two (2) hour paper
  - 35 MCQ questions (2 marks per question = 70 marks)
  - One (1) essay question (30 marks)

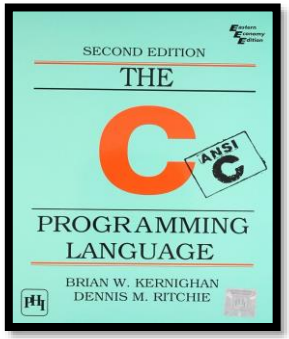
Course Administration …(2)

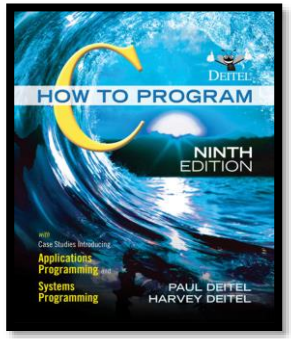
- This is a four (4C = 3L + 1P) credit course module
  - Three (3) lecture hours per week – Thursday 8.00 AM – 12.00 NN @ S 104
  - Two (2) practical hours per week

- Tools
  - C Programming language
  - GNU Compiler Collection (gcc)
  - VIM text editor
  - Linux based OS environment
  - Strictly no Integrated Development Environments (IDEs)

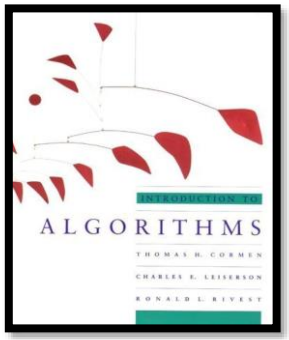
Course Administration …(3)

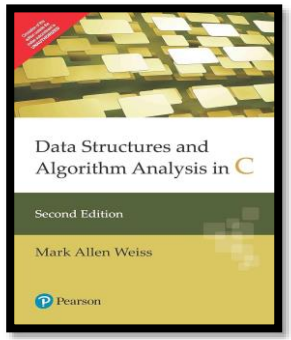
- Recommended Reference Books
  - Kernighan, B.W. and Ritchie, D.M., 2002. *The C programming language*.
  - Deitel, H.M. and Deitel, H. *C: how to program*. Prentice-Hall, Upper Saddle.





  - Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C., 2022. *Introduction to algorithms*. MIT press.
  - Weiss, M.A., 1997. *Data Structures and Algorithm Analysis in C*. Pearson Education India.





Course Administration …(4)

- Semester Plan
  - C Programming Language (1 – 6 weeks)
  - Linear Data Structures (9 – 12 weeks)
  - Algorithm Analysis (13 – 15 weeks)

What is a Computer?

- Is it mix of different minerals such as silica, iron, aluminum, copper, lead, zinc, nickel, tin, selenium, manganese, arsenic, and cadmium wired together?
- What language does a computer understand?
- Does it really understand what it does?
- Classical Definition → “A programmable usually electronic device that can store, retrieve, and process data” ~ Merriam Webster

What is a Programming Language?

- A programming language is a set of constructs which enable a programmer combine in various ways to deliver instructions to the computer to perform and accomplish a task.



Generations to Programming Languages …(1)

- First (1st) generation languages
  - Low-level language more like resembling machine code
  - Machine dependent

- Second (2nd) generation languages
  - Contains human readable notation
  - Converted to machine language using an assembler

Generations to Programming Languages …(2)

- Third (3rd) generation languages
  - Consists of the use of a series of English-like words that humans can understand easily.
  - Translated into machine code using translator or interpreter.

- Fourth (4th) generation languages
  - More high-level than 3rd generation. More specific to a discipline such as data and database manipulations.
  - They are more efficient that 3G/L

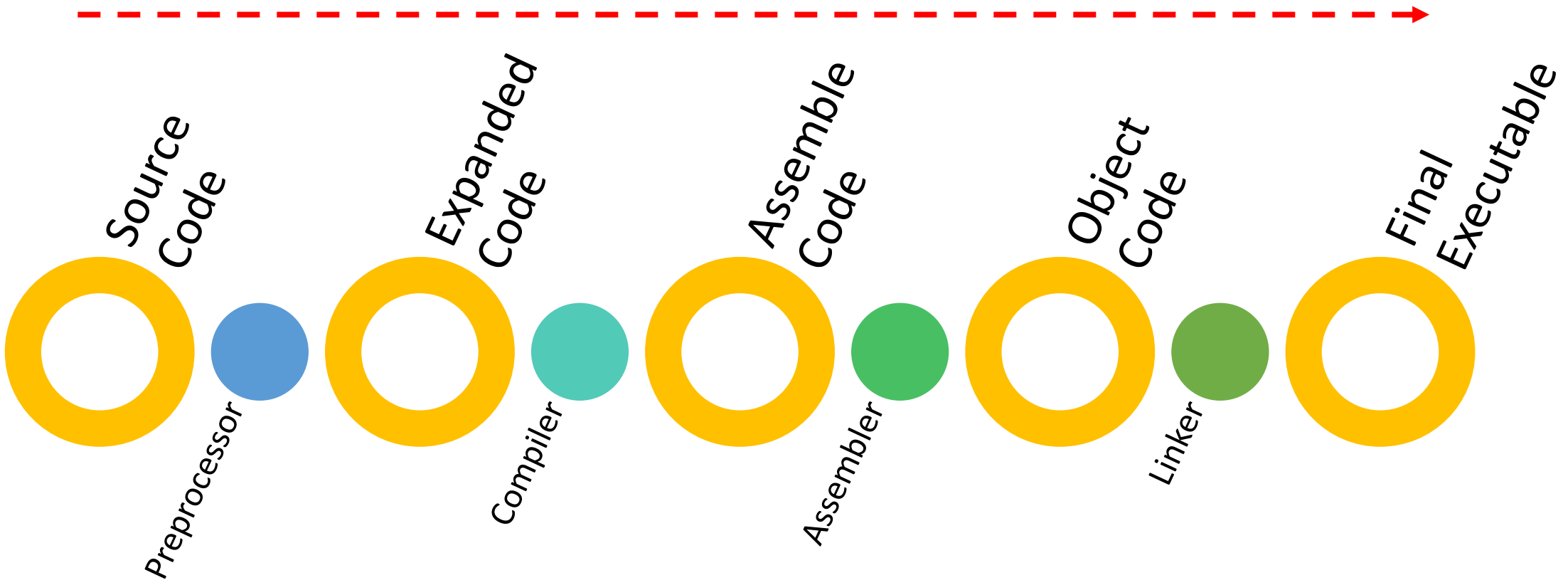
- Fifth (5th) generation languages

What is an Operating System?

- Is a software that manages hardware and software resources of a computer and provides a common services for computer programs.

- What are common services?
  - File management
  - Input/output management
  - Error handling
  - Processes
  - Communication between process

What is Compilation? …(1)



What is Compilation? …(2)

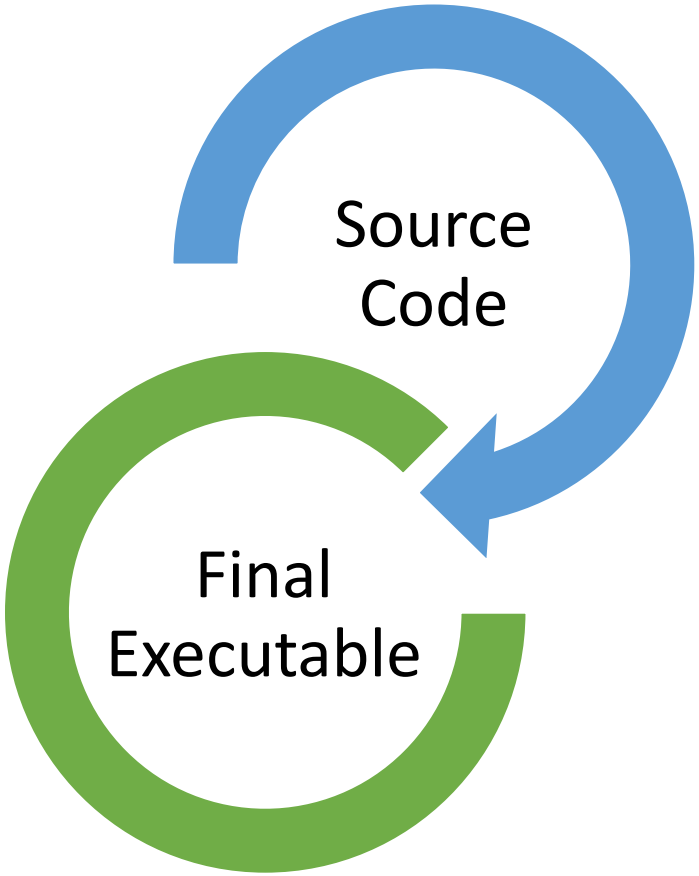
- Examples for compiled programming languages are C, C++, and C#.



- Compiled languages are faster and provides a better performance.

- Compiled languages converts the source language into a targeted language.

What is Interpretation? …(1)



- Interpreter

What is Interpretation? …(2)

- Examples for interpreted languages are Python, JavaScript, and Perl.


- Much slower than compiled languages in performance and execution but however provides more facilities during execution to the programmer.

- Java is a compiled and an interpreted language which is a separate category called hybrid implementation.

What is Programming?

- Is the act of writing a sequence of instructions using a programming language that the computers can follow and perform.
- The task is also called coding and is both an art is much as a science.
- The sequence of instructions are also called programs.

The C Programming Language

- C Programming language was invented in 1972 by Dennis Ritchie.
- C was evolved from two previous (no longer in common use) languages known as BCPL (Basic Combined Programming Language) and B.
- C became famous as the implementation language of UNIX and to date many of operating systems are written using C or its derivative C++.

Why Learn C Programming Language?

- C is widely used and is one of the most efficient high-level languages invented. Following are some domains the C language is prevalent:
  - Operating Systems
  - Real Time Systems
  - Embedded Systems

- Enables a beginner to better understand underlying programming concepts such as memory management, pointers than any other programming language.

- C has a **steep** learning curve.

Current C Standard

- The Standard C version was approved in the United States through the American National Standards Institute (ANSI), then worldwide through the International Standards Organization (ISO) in 1989.

- The latest C standard (referred to as **C11**), which was approved in 2011 updated with bug fixes in 2018 (referred to as **C18**).

- The current C standard document is referred to as *ISO/IEC 9899:2018*.

First C Program …(1)

```
1. #include <stdio.h>
2.
3. //Main begins the execution of program
4. int main()
5. {
6.       printf("Hello World!");
7.       return 0;
8. }
```

First C Program …(2)

- To compile the program on UNIX systems

```
$gcc -std=c18 hello.c [-std=c18 is optional]
$gcc -o [output name] hello.c
```

- The above step creates an output file **a.out,** which is directly executable.

- To execute the program

```
$./a.out
```

`#include` Preprocessor Directive …(1)

- `#include` represent a preprocessor directive.
- Any line in a C program that starts with a # at the beginning of a source file is a preprocessor directive.
- These directives are processed by the prepossessing step.
- **Line 1** includes the contents of the standard input and output header file (**stdio.h**) to our source program.

`#include` Preprocessor Directive …(2)

- To see the preprocessor output

```
$gcc –E hello.c > hello.p (can be any name)
```

Red part will redirect the output to the file ‘hello.p’

- The option –E when used stops the compilation process after the preprocessing stage.

- The output is in the form of preprocessed source code, which is sent to the standard output.

Comments …(1)

- The comments in C language is two(2) forms as:
  - single line comments
  - multiline comments

- Single line comments *(Line 2 of the first program)*
  - E.g. `//This is a comment`

- Multiline comments
  - E.g.
```
/*
 * This is a comment
 */
```

Comments ...(2)

- Comments are programmer aids and does not effect the execution of the program.
- The comments are ignored by the compiler when the executable is being created.
- Comments should be used by programmers to explain an important aspect regarding the program and **not** the same thing as the program syntax.

The `main()` function

- Main function or the main method is a part of every C program.
- This is the main entry point to a C program.
- General form is:

```
int main()
{
    //your code goes here
    return 0;
}
```

`printf(…)` function

- Is a function that displays anything between the parenthesis.

- The **'f'** at the end of the name in `printf` stands for **"formatted"**. Which means `printf(…)` produces formatted output to be displayed on the screen.

- `printf` function have number of parameters that can be combined in various ways to get different formatted outputs.

Intermediate Phases in Compilation …(1)

- To see the output after the compilation, before the assembler is run

```
$gcc –S hello.c
```

- The above step will result in an assembly file named “hello.s”

- To see the output after the execution of the assembler and before the linker

```
$gcc –c hello.s
```

- The above step will result in an object file named “hello.o”

Intermediate Phases in Compilation …(2)

- To get the final executable after the linking process

```
$gcc hello.o
```

- The above will create an executable file named “a.out”

Variables…(1)

- In general a variable means a value that can change.
- In programing variables are containers for storing data values.
- In C, variable is declared using the following syntax
  - `<data type> <variable name>;`
  - `<data type> <variable name> = <initial value>;`

Variables…(2)

- Variable name in C language
  - can contain letters, digits and underscores
  - must begin with a letter or an underscore (_)
  - are case-sensitive
  - cannot contain whitespaces or special characters like !, #, %, etc.
  - Cannot be any reserved word (such as int)

Data Types in C …(1)

- The data types in C can be categorized in to three (3) groups as:
  - Primitives
  - Derived Types
  - User Defined Types

Data Types in C …(2)

- There are four primitive data types as `int`, `float`, `double`, and `char`.
- Integer (int) types can also be used as `short int`, `long int` and unsigned variation of those types.
- Character (char) types allow to store a single character in a char variable.
- Maximum and minimum possible values of each data type above is defined macros in **`<limits.h>`**

Data Types in C …(3)

- Numbers with floating point values are stored using float and double data types.
- Floats are single precision floating point numbers while double is called double precision floating point numbers.
- When representing a float the ‘f’ literal has to be appended to the end of the assigned value.
- Maximum and minimum possible values of each data type above is defined macros in **`<float.h>`**

Data Types in C …(3)

- The **`sizeof(…)`** operator can be used to find the sizes of each data type or a given variable in memory.

- The value returned is in bytes.

printf(…) …(1)

- The printf function is used to print formatted output to the console screen.

- Syntax

```
printf("formatted string", arguments_list);
```

`printf(...)` ...(2)

- The formatted string of the `printf` function begin with a percentage symbol '%' and take the following form

```
%[flags][width][.precision][length]specifier
```

- Specifier represent the character representing the data type to be printed. Common specifiers are %d, %f, %c, %s, and %p.

`printf(...)` ...(3)

- Width
  - Specifies the number of minimum characters that will be printed in the standard output.
  - If the number of characters are greater than the specified width, all characters will be printed irrespective of the width set.
  - If the number of characters are less than the specified width, whitespace is included to fill the remaining spaces.

`printf(...)` ...(4)

- Precision
  - Has different meanings for different data types.
    - For integer forms it specifies the minimum number of digits to be printed.
    - For floating point numbers it specifies the number of digits to be printed after the precision point.
    - For string types specifies the length of the string to be printed.

- Length
  - Specifies the length of the data type in memory.
    - h, l, and L specifiers are used for this purpose.

`printf(...)` ...(5)

- **Exercise**
  - Find what are the flag specifiers that can be used with printf function.
  - Write C programs to observe the behavior of the flags and other components of the format.
  - What other possibilities are available for specifiers, width, precision and length?

Escape Sequences

Escape Sequences	Description
\n	Moves the cursor to the beginning of the next line.
\t	Mover the cursor to the next horizontal tab stop.
\a	Produces a sound or a visible alert. Does not adjust the current cursor position.
\\ and \”	\ and “ has special meanings. Therefore the backslash has to be used to print them in screen.

Questions?