C PROGRAMMING

FYBSC SEM 2

INTRODUCTION

- A general purpose language, closely associated with UNIX that developed in BELL laboratories.
- Most of the programs of UNIX is written with the help of C.
- C was originally developed at Bell Labs by Dennis Ritchie between 1972 and 1973 to construct utilities running on Unix.
- From beginning C was useful for busy programmers to get things done easily because C is powerful, dominant and flexible language.

About C

- It is a structured and disciplined approach to program design.
- It supports functions that enables easy maintainability of code by breaking large file into smaller modules.
- Comments in C provides easy readability.
- The C program is built from:

1. Variables and type declarations.
2. Functions
3. Statements.
4. Expressions.

Cont.

- C language is also known as “Compiled Language” as the source code must first be compiled in order to run.

Structure of C program

1. Documentation Section.
2. Link Section.
3. Global Declaration section.
4. Main()

{

Declaration part1

Declaration part2

}

5. Subprogram section.

Function 1

Function 2

.

.

Function n

Header and Body.

- The **header** includes the name of the function and tells us (and the compiler) what type of data it expects to receive (the *parameters*) and the type of data it will return (*return value type*) to the calling function or program.
- Eg: #include<stdio.h>
- The body of the function contains the instructions to be executed.

Basic syntax

```
main( )
{
/*.....printing begins.....*/
    printf("I see, I remember");
/*.....printing ends.....*/
}
```

Fig. 1.2 *A program to print one line of text*

Use of Comments

- The comments are basic simple lines which are just used to read the code or understand the code.
- This comment lines are not executed by the compiler as it is only for reading purpose only.
- These are used to enhance its readability and understanding.
- There are two types of comment line:

1. Single line comment.
2. Multi line comment.

Cont.

1. A single line comment is used to comment a single line.

Eg: //Single line comment.

2. A multi-line comment is used to comment multiple lines in code.

Eg: /* this is multi line

.

.

comment ends */

Interpreters Vs Compiler

- Compliers and interpreters are programs that help convert the high level language (Source Code) into machine codes to be understood by the computers.
- A high level language is one that can be understood by humans.
- However, computers cannot understand high level languages as we humans do. So, the source codes must be converted into machine language and here comes the role of compilers and interpreters.

Cont.

Interpreter translates just one statement of the program at a time into machine code.	Compiler scans the entire program and translates the whole of it into machine code at once.
An interpreter takes very less time to analyze the source code. However, the overall time to execute the process is much slower.	A compiler takes a lot of time to analyze the source code. However, the overall time taken to execute the process is much faster.
An interpreter does not generate an intermediary code. Hence, an interpreter is highly efficient in terms of its memory.	A compiler always generates an intermediary object code. It will need further linking. Hence more memory is needed.
Keeps translating the program continuously till the first error is confronted. If any error is spotted, it stops working and hence debugging becomes easy.	A compiler generates the error message only after it scans the complete program and hence debugging is relatively harder while working with a compiler.
Interpreters are used by programming languages like Ruby and Python for example.	Compliers are used by programming languages like C and C++ for example.

Python Vs C

- Python is a general-purpose, high-level programming language that was developed by Guido Rossum in 1989. What makes Python amazing is its simple syntax that is almost similar to the English language and dynamic typing capability. The straightforward syntax allows for easy code readability.
- Being an interpreted language, Python is an ideal language for scripting and rapid application development on most platforms and is so popular with the developers. Scripting languages incorporate both interactive and dynamic functionalities via web-based applications.

Cont.

What is the Difference Between Python and C Language?

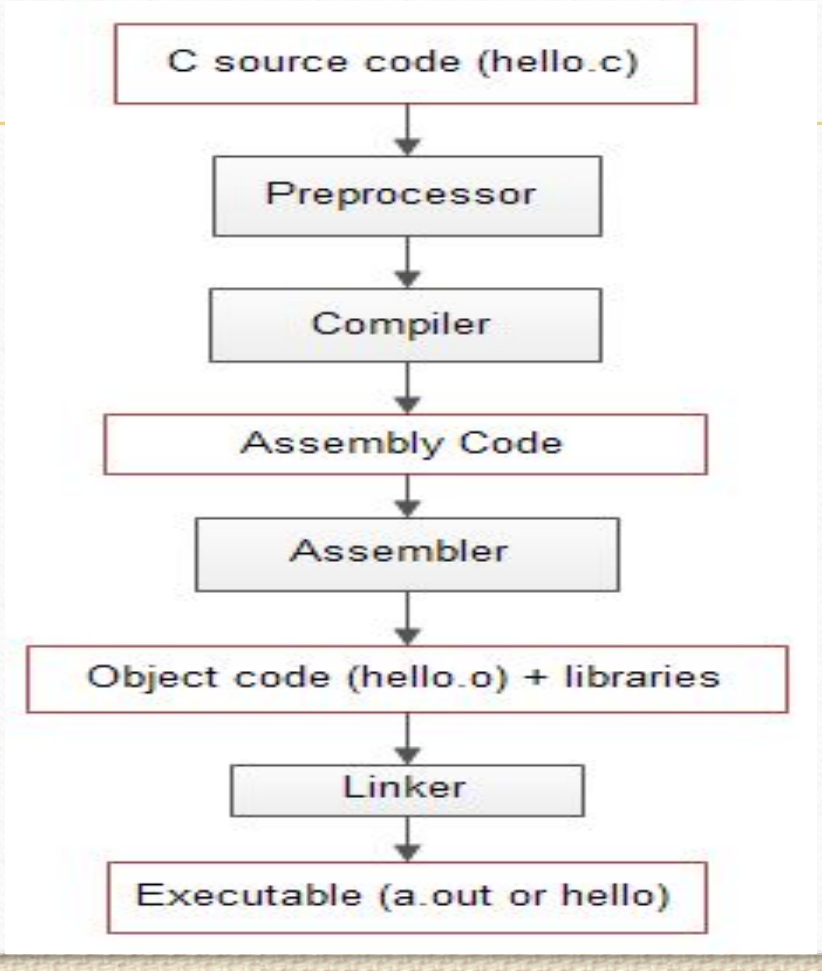
Python vs C Language	
Python is a multi-paradigm. It mainly supports Object-oriented programming, Procedural programming, Functional programming.	C is a Structured programming language.
Language Type	
Python is an interpreter based language. The interpreter reads the code line by line.	C is a compiled language. The complete source code is converted into machine language.
Memory Management	
Python use automatic garbage collector for memory management.	In C, Programmer has to do memory management on his own.
Applications	
Python is a General-Purpose programming language.	C is mainly used for hardware related applications.
Speed	
Python is slow.	C is fast.
Variable Declaration	
In Python, no need to declare variable type.	In C, it is compulsory to declare variable type.
Complexity	
Python programs are easier to learn, write and read.	C program syntax is harder than Python.
Testing and Debugging	
Testing and debugging is easier in Python.	Testing and debugging is harder in C.

Compilation

- The compilation is a process of converting the source code into object code. It is done with the help of the compiler. The compiler checks the source code for the syntactical or structural errors, and if the source code is error-free, then it generates the object code.
- The c compilation process converts the source code taken as input into the object code or machine code. The compilation process can be divided into four steps, i.e., Pre-processing, Compiling, Assembling, and Linking.
- In the pre-processor, The source code is the code which is written in a text editor and the source code file is given an extension ".c". This source code is first passed to the preprocessor, and then the preprocessor expands this code. After expanding the code, the expanded code is passed to the compiler.

- In the compiler, The code which is expanded by the preprocessor is passed to the compiler. The compiler converts this code into assembly code. Or we can say that the C compiler converts the pre-processed code into assembly code.
- Then the assembly code is converted into object code by using an assembler. The name of the object file generated by the assembler is the same as the source file. If the name of the source file is **'hello.c',** then the name of the object file would be 'hello.obj'.
- The linking is the process of putting together other program files and functions that are required by the program. For eg: if the program is using exp() function then the object code of this function should be brought from the math library.

Diagram.


C source code (hello.c)
Preprocessor
Compiler
Assembly Code
Assembler
Object code (hello.o) + libraries
Linker
Executable (a.out or hello)

Formatted I/O

- C provides standard functions scanf() and printf(), for performing formatted input and output .These functions accept, as parameters, a format specification string and a list of variables.
- The format specification string is a character string that specifies the data type of each variable to be input or output and the size or width of the input and output.

For Input

- Scanf(): To read data in from standard input (keyboard), we call the **scanf** function. The basic form of a call to scanf is: scanf(*format_string*, *list_of_variable_addresses*);
- If **x** is a variable, then the expression **&x** means "address of x"

For Output

- Printf(): The function printf() is used for formatted output to standard output based on a format specification. The format specification string, along with the data to be output, are the parameters to the printf() function.
- Basic syntax: printf(format, data1, data2,……..);
- The character specified after % is called a conversion character because it allows one data type to be converted to another type and printed.
- The conversion character used in C are:

1. %d : the data is converted into Decimal(integer).
2. %s : The data is a string and character from the string , are printed until a NULL, character is reached.
3. %c : The data is taken as a character.
4. %f : The data is output as float or double with a default Precision 6.

Data

- 1. **Variables**: A variable are names used to refer to some memory locations of computer.
- The piece of information stored at this location is referred as value of variable.
- The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because C is case-sensitive.
- For eg: int a=10; (Here “int” is datatype and “a” is the variable name)

Constant

2. Constants: It is a value that cannot be changed during the execution of the program.

- Constants are treated just like regular variables except that their values cannot be modified after their definition.
- For Eg: #define PI=3.14

Difference.

Constants	Variable
A value that can not be altered throughout the program	A storage location paired with an associated symbolic name which has a value
It is similar to a variable but it cannot be modified by the program once defined	A storage area holds data
Can not be changed	Can be changed according to the need of the programmer
Value is fixed	Value is varying

Data Types

- Data types specify how we enter data into our programs and what type of data we enter. C language has some predefined set of data types to handle various kinds of data that we can use in our program. These datatypes have different storage capacities.
- The data types includes are:

1. Int:
2. Float:
3. Char:
4. Void:
5. Double
6. Short
7. Long
8. Signed and unsigned

Example.

Data Type	Range	Bytes	Format
signed char	-128 to + 127	1	%c
unsigned char	0 to 255	1	%c
short signed int	-32768 to +32767	2	%d
short unsigned int	0 to 65535	2	%u
signed int	-32768 to +32767	2	%d
unsigned int	0 to 65535	2	%u
long signed int	-2147483648 to +2147483647	4	%ld
long unsigned int	0 to 4294967295	4	%lu
float	-3.4e38 to +3.4e38	4	%f
double	-1.7e308 to +1.7e308	8	%lf
long double	-1.7e4932 to +1.7e4932	10	%Lf