1. OVERVIEW

C is a general-purpose, high-level language that was originally developed by Dennis M. Ritchie to develop the UNIX operating system at Bell Labs. C was originally first implemented on the DEC PDP-11 computer in 1972.

In 1978, Brian Kernighan and Dennis Ritchie produced the first publicly available description of C, now known as the K&R standard.

The UNIX operating system, the C compiler, and essentially all UNIX application programs have been written in C. C has now become a widely used professional language for various reasons:

☐ Easy to learn
☐ Structured language
☐ It produces efficient programs
☐ It can handle low-level activities
☐ It can be compiled on a variety of computer platforms

Facts about C

☐ C was invented to write an operating system called UNIX.
☐ C is a successor of B language which was introduced around the early 1970s.
☐ The language was formalized in 1988 by the American National Standard Institute (ANSI).
☐ The UNIX OS was totally written in C.
☐ Today C is the most widely used and popular System Programming Language.
☐ Most of the state-of-the-art software have been implemented using C.
☐ Today's most popular Linux OS and RDBMS MySQL have been written in C.

Why

C was initially used for system development work, particularly the programs that make-up the operating system. C was adopted as a system development language because it produces code that runs nearly as fast as the code written in assembly language. Some examples of the use of C might be:

☐ Operating Systems

☐ Language Compilers
☐ Assemblers
☐ Text Editors
☐ Print Spoolers
☐ Network Drivers

☐ Modern Programs
☐ Databases
☐ Language Interpreters
☐ Utilities

C Programs

A C program can vary from 3 lines to millions of lines and it should be written into one or more text files with extension ".c"; for example, hello.c. You can use "vi", "vim" or any other text editor to write your C program into a file.

This tutorial assumes that you know how to edit a text file and how to write source code inside a program file.

CHP-2

Try it Option Online

You really do not need to set up your own environment to start learning C programming language. Reason is very simple, we already have set up C Programming environment online, so that you can compile and execute all the available examples online at the same time when you are doing your theory work. This gives you confidence in what you are reading and to check the result with different options. Feel free to modify any example and execute it online.

Try following example using our online compiler option available at http://www.compileonline.com/.

```
#include <stdio.h>
int main()
{
/* my first program in C */
printf("Hello, World! \n");
return 0;
}
```

For most of the examples given in this tutorial, you will find the Try it option in our website code sections at the top right corner that will take you to the online compiler. So just make use of it and enjoy your learning.

Local Environment Setup

If you want to set up your environment for C programming language, you need the following two software tools available on your computer, (a) Text Editor and (b) The C Compiler.

Text Editor

This will be used to type your program. Examples of a few editors include Windows Notepad, OS Edit command, Brief, Epsilon, EMACS, and vim or vi.

2. ENVIORNMENT SETUP

The name and version of text editors can vary on different operating systems. For example, Notepad will be used on Windows, and vim or vi can be used on Windows as well as on Linux or UNIX.

The files you create with your editor are called the source files and they contain the program source codes. The source files for C programs are typically named with the extension ".c".

Before starting your programming, make sure you have one text editor in place and you have enough experience to write a computer program, save it in a file, compile it and finally execute it.

The C Compiler

The source code written in source file is the human readable source for your program. It needs to be "compiled" into machine language so that your CPU can actually execute the program as per the instructions given.

The compiler compiles the source codes into final executable programs. The most frequently used and free available compiler is the GNU C/C++ compiler, otherwise you can have compilers either from HP or Solaris if you have the respective operating systems.

The following section explains how to install GNU C/C++ compiler on various OS. m We keep mentioning C/C++ together because GNU gcc compiler works for both C and C++ programming languages.

Installation on UNIX/Linux

If you are using Linux or UNIX, then check whether GCC is installed on your system by entering the following command from the command line:

```
$ gcc -v
```

If you have GNU compiler installed on your machine, then it should print a message as follows:

```
Using built-in specs.
Target: i386-redhat-linux
Configured with: ../configure --prefix=/usr .......
Thread model: posix
gcc version 4.1.2 20080704 (Red Hat 4.1.2-46)
```

If GCC is not installed, then you will have to install it yourself using the detailed instructions available at http://gcc.gnu.org/install/.

This tutorial has been written based on Linux and all the given examples have been compiled on the Cent OS flavor of the Linux system.

Installation on Mac OS

If you use Mac OS X, the easiest way to obtain GCC is to download the Xcode development environment from Apple's web site and follow the simple installation instructions. Once you have Xcode setup, you will be able to use GNU compiler for C/C++.

Xcode is currently available at developer.apple.com/technologies/tools/.

Installation on Windows

To install GCC on Windows, you need to install MinGW. To install MinGW, go to the MinGW homepage, www.mingw.org, and follow the link to the MinGW download page. Download the latest version of the MinGW installation program, which should be named MinGW-<version>.exe.

While installing MinGW, at a minimum, you must install gcc-core, gcc-g++, binutils, and the MinGW runtime, but you may wish to install more.

Add the bin subdirectory of your MinGW installation to your PATH environment variable, so that you can specify these tools on the command line by their simple names.

After the installation is complete, you will be able to run gcc, g++, ar, ranlib, dlltool, and several other GNU tools from the Windows command line.

3. PROGRAM STRUCTURE

Before we study the basic building blocks of the C programming language, let us look at a bare minimum C program structure so that we can take it as a reference in the upcoming chapters.

Hello World Example

A C program basically consists of the following parts:

☐ Preprocessor Commands
☐ Functions
☐ Variables
☐ Statements & Expressions
☐ Comments

Let us look at a simple code that would print the words "Hello World":

```
#include <stdio.h>
int main()
{
/* my first program in C */
printf("Hello, World! \n");
return 0;
}
```

Let us take a look at the various parts of the above program:

1. The first line of the program #include <stdio.h> is a preprocessor command, which tells a C compiler to include stdio.h file before going to actual compilation.
2. The next line int main() is the main function where the program execution begins.
3. The next line /*...*/ will be ignored by the compiler and it has been put to add additional comments in the program. So such lines are called comments in the program.

4. The next line printf(...) is another function available in C which causes the message "Hello, World!" to be displayed on the screen.
5. The next line return 0; terminates the main() function and returns the value 0.

Compile

Let us see how to save the source code in a file, and how to compile and run it. Following are the simple steps:

1. Open a text editor and add the above-mentioned code.
2. Save the file as hello.c
3. Open a command prompt and go to the directory where you have saved the file.
4. Type gcc hello.c and press enter to compile your code.
5. If there are no errors in your code, the command prompt will take you to the next line and would generate a.out executable file.
6. Now, type a.out to execute your program.
7. You will see the output "Hello World" printed on the screen.

```
$ gcc hello.c
$ ./a.out
Hello, World!
```

Make sure the gcc compiler is in your path and that you are running it in the directory containing the source file hello.c.

You have seen the basic structure of a C program, so it will be easy to understand other basic building blocks of the C programming language.

Tokens in C

A C program consists of various tokens and a token is either a keyword, an identifier, a constant, a string literal, or a symbol. For example, the following C statement consists of five tokens:

```
printf("Hello, World! \n");
```

The individual tokens are:

```
printf
(
"Hello, World! \n"
)
;
```

Semicolons

In a C program, the semicolon is a statement terminator. That is, each individual statement must be ended with a semicolon. It indicates the end of one logical entity.

Given below are two different statements:

```
printf("Hello, World! \n");
return 0;
```

Comments

Comments are like helping text in your C program and they are ignored by the compiler. They start with /* and terminate with the characters */ as shown below:

```
/* my first program in C */
```

You cannot have comments within comments and they do not occur within a string or character literals.

Identifiers

A C identifier is a name used to identify a variable, function, or any other user-defined item. An identifier starts with a letter A to Z, a to z, or an underscore '_' followed by zero or more letters, underscores, and digits (0 to 9).

C does not allow punctuation characters such as @, $, and % within identifiers. C is a case-sensitive programming language. Thus, Manpower and manpower are two different identifiers in C. Here are some examples of acceptable identifiers:

mohd zara abc move_name a_123
myname50 _temp j a23b9 retVal

Keywords

The following list shows the reserved words in C. These reserved words may not be used as constants or variables or any other identifier names.

auto
break
case
char
const
continue
default
do
double
else
enum
extern
float
for
goto
if
int
long
register
return
short
signed
sizeof
static
struct
switch
typedef
union
unsigned
void
volatile
while
_Packed

Whitespace in C

A line containing only whitespace, possibly with a comment, is known as a blank line, and a C compiler totally ignores it.

Whitespace is the term used in C to describe blanks, tabs, newline characters and comments. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement, such as int, ends and the next element begins. Therefore, in the following statement:

```
int age;
```

there must be at least one whitespace character (usually a space) between int and age for the compiler to be able to distinguish them. On the other hand, in the following statement:

```
fruit = apples + oranges; // get the total fruit
```

no whitespace characters are necessary between fruit and =, or between = and apples, although you are free to include some if you wish to increase readability.