UNIT I

Introduction

C Programming Language is a very popular computer programming language through which users and computers can communicate

What is C?

C is a computer programming language used to design computer software's and applications.

Why do we use C?

C programming language is very popular computer language used to design computer software's and applications.

Who invented C?

- C Programming Language was invented in the year of 1972 by Dennis Ritchie (Dennis Macalister Ritchie).
- He was an American Computer Scientist worked at Bell Labs as researcher along with Ken Thompson. He was born on 9th September 1941 and lived till 12th October 2011.
- He is said to be the Father of C.

Father of C

Software's used to create and execute C Program?

Following are the software's and applications used to create and execute C programs...

1. Turbo C -
2. Turbo C++ -
3. GNU C -
4. Code Blocks -
5. Net Beans

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

Computer Systems:

What is Computer?

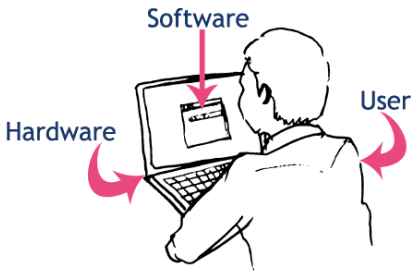
Computer is an electronic device which operates under the control of instructions stored in its own memory. A computer can take data from the user through input devices (Input), process the user given data (Processing), produces the result to the user through output devices (Output)and stores data (Information) for future use. Computer can be defined as follows...

Computer is an electronic device which operates under the control of instructions stored in its own memory and it takes the data from the user, process that data, gives the result and stores the result for future use.

What does Computer consists of?

Every computer mainly consists of three things and those are...

- Hardware
- Software
- User



Here the user interacts with the software, and the software makes the computer hardware parts to work for the user.

What is Computer Hardware?

All physical components of the computer are called as computer hardware. A user can see, touch and feel every hardware of the computer. All hardware components perform any task based on the instructions given by the computer software.

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

The computer hardware is the physical part of a computer.

The computer hardware components are as follows...

1. **Input Devices.**
2. **Output Devices.**
3. **Storage Devices**
4. **Devices Drives**
5. **Cables**
6. **Other Devices**

Input Devices

Computer input devices are the physical components of the computer which are used to give the data given by the user to the computer. Using input devices the user can give the data to the computer.

Example:


Keyboard


Mouse


Joy Stick


Mic


Barcode Reader


Stylus/Pen


Web Camera


Touch pad


Touch Screen


Finger Print reader

Output Devices:

Computer output devices are the physical components of the computer which are used to give the computer result to the User. Using output devices, the user can see the computer generated result.

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

Example:

Storage Devices:

Computer storage devices are the physical components of the computer which are used to store data internally or externally.

Example:

Device Drives:

Computer Device drives are the physical components of the computer which are used to read and write data of the storage devices.

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

Example:

CD/DVD Drive

Floppy Drive

Computer Cables

In a Computer, various cables are used to make connections among the various hardware components of the computer.

Example:

Other Devices:

Other devices of the computer are shown below...

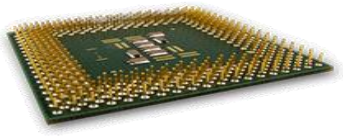
P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

CPU (Processor)

Front



Back

SMPS:

CPU Fan (Heat Sinck):

Cooling Fan

CPU Cooling Fan

Mother Board:

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

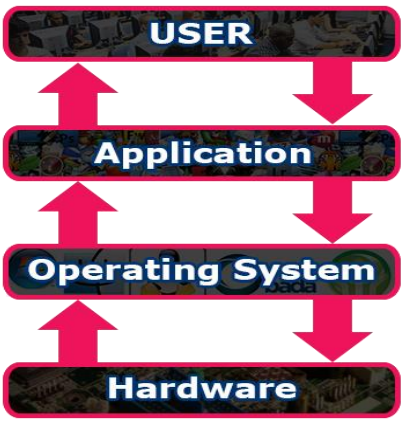
UNIT I

How does Computer works?

When a user wants to communicate with the computer, the user interacts with an application. The application interacts with the operating system, and the operating system makes hardware components to work according to the user given instructions. The hardware components sends the result back to the operating system, then the operating system forwards the same to the application and the application shows the result to the user.

By using input devices, the user interacts with the application and the application uses output devices to show the result. All input and output devices work according to the instructions given by the operating system.

The working process of a computer is shown in the following figure...



P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

Computing Environments

What is Computing Environment?

When we want to solve a problem using computer, the computer makes use of various devices which work together to solve that problem. There may be various number of ways to solve a problem. We use various number of computer devices arranged in different ways to solve different problems. The arrangement of computer devices to solve a problem is said to be computing environment. The formal definition of computing environment is as follows...

Definition:

Computing Environment is a collection of computers which are used to process and exchange the information to solve various types of computing problems.

Types of Computing Environments

The following are the various types of computing environment's...

1. **Personal Computing Environment**
2. **Time Sharing Computing Environment**
3. **Client Server Computing Environment**
4. **Distributed Computing Environment**
5. **Grid Computing Environment**
6. **Cluster Computing Environment**

Personal Computing Environment

- Personal computing is a stand-alone machine.
- In personal computing environment, the complete program resides on stand-alone machine and executed from the same machine.

Example:

- Laptops, mobile devices, printers, scanners and the computer systems we use at home, office are the examples for personal computing environment.

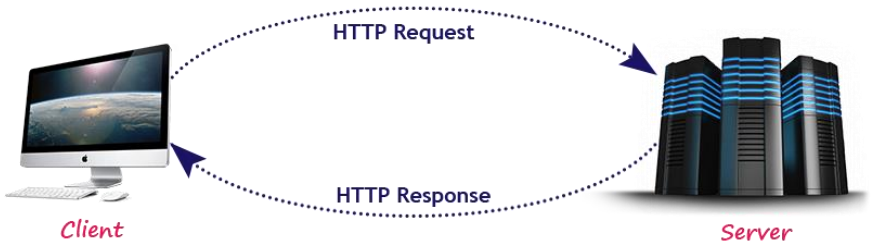
UNIT I

Time Sharing Computing Environment

Time sharing computing environment is standalone computer in which a single user can perform multiple operations at a time by using multitasking operating system. Here the processor time is divided among different tasks and this is called "Time sharing". For example, a user can listen to music while writing something in a text editor. Windows 95 and later versions of windows OS, iOS and Linux operating systems are the examples for this computing environment.

Client Server Computing Environment

- The client server environment contains two machines (Client machine and Server machine).
- These both machines will exchange the information through an application.
- Here Client is a normal computer like PC, Tablet, Mobile, etc., and Server is a powerful computer which stores huge data and manages huge amount of file and emails, etc.,
- In this environment, client requests for data and server provides data to the client. In the client server environment, the communication between client and server is performed using HTTP (Hyper Text Transfer Protocol).



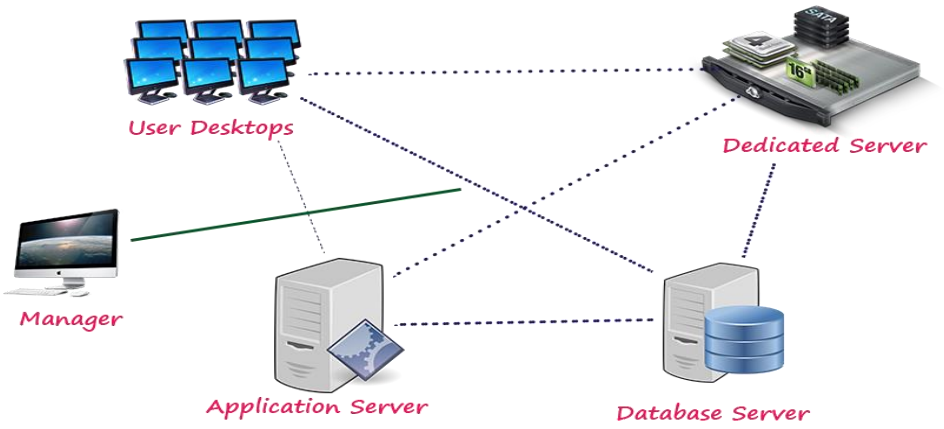
Distributed Computing Environment

- In the distributed computing environment, the complete functionality of a software is not on single computer but is distributed among multiple computers.
- Here we use a method of computer processing in which different programs of an application run simultaneously on two or more computers.
- These computers communicate with each other over a network to perform the complete task.
- In distributed computing environment, the data is distributed among different systems and that data is logically related to each other.

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I



Grid Computing Environment

Grid computing is a collection of computers from different locations. All these computers work for a common problem. A grid can be described as distributed collection of large number of computers working for a single application.

Cluster Computing Environment

Cluster computing is a collection of inter connected computers. These computers work together to solve a single problem. In cluster computing environment, a collection of systems work together as a single system.

Computer Languages:

What is Computer Language?

Generally, we use languages like English, Hindi, Telugu etc., to make communication between two persons. That means, when we want to make communication between two persons we need a language through which persons can express their feelings. Similarly, when we want to make communication between user and computer or between two or more computers we need a language through which user can give information to computer and vice versa. When user wants to give any instruction to the computer the user needs a specific language and that language is known as computer language.

UNIT I

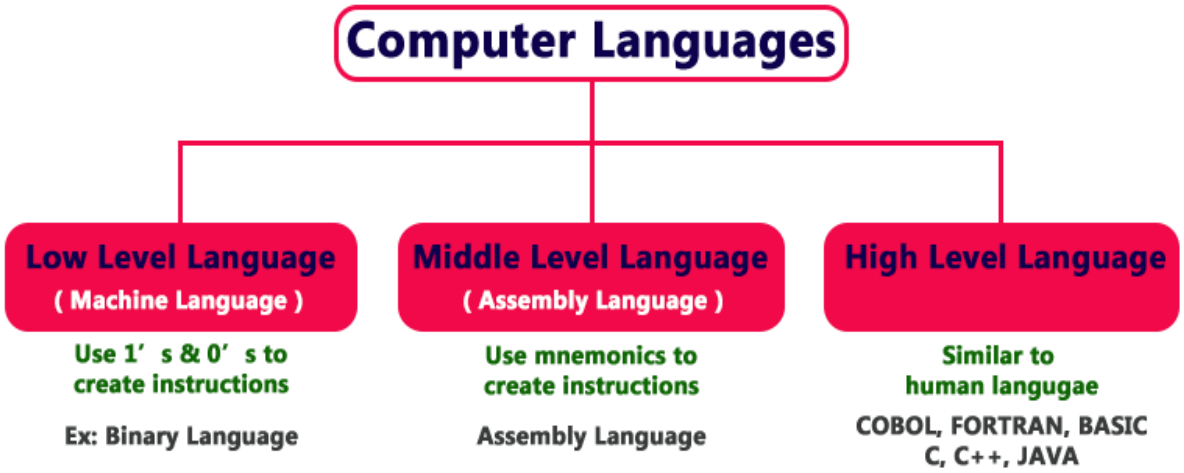
User interacts with the computer using programs and that programs are created using computer programming languages like C, C++, Java etc.,

Computer languages are the languages through which user can communicate with the computer by writing program instructions.

Every computer programming language contains a set of predefined words and a set of rules (syntax) that are used to create instructions of a program.

Computer Languages Classification

Over the years, computer languages have been evolved from Low Level to High Level Languages. In the earliest days of computers, only Binary Language was used to write programs. The computer languages are classified as follows...



Low Level Language (Machine Language):

- Low Level language is the only language which can be understood by the computer. Binary Language is an example of low level language.
- Low level language is also known as Machine Language.
- The binary language contains only two symbols 1 & 0.
- All the instructions of binary language are written in the form of binary numbers 1's & 0's. A computer can directly understand the binary language.
- Machine language is also known as Machine Code.

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

- As the CPU directly understands the binary language instructions, it does not requires any translator.

- CPU directly starts executing the binary language instructions, and takes very less time to execute the instructions as it does not requires any translation. Low level language is considered as the First Generation Language (1GL).

Advantages

- A computer can easily understand the low level language.
- Low level language instructions are executed directly without any translation.
- Low level language instructions require very less time for their execution.

Disadvantages

- Low level language instructions are very difficult to use and understand.
- Low level language instructions are machine dependent, that means a program written for a particular machine does not executes on other machine.
- In low level language, there is more chance for errors and it is very difficult to find errors, debug and modify.

Middle Level Language (Assembly Language):

- Middle level language is a computer language in which the instructions are created using symbols such as letters, digits and special characters.
- Assembly language is an example of middle level language.
- In assembly language, we use predefined words called **mnemonics**.
- Binary code instructions in low level language are replaced with mnemonics and operands in middle level language.
- But computer can not understand mnemonics, so we use a translator called **Assembler** to translate mnemonics into binary language.
- Assembler is a translator which takes assembly code as input and produces machine code as output. That means, computer can not understand middle level language, so it needs to be translated into low level language to make it understandable by the computer. Assembler is used to translate middle level language to low level language.

UNIT I

Advantages

- Writing instructions in middle level language is easier than writing instructions in low level language.
- Middle level language is more readable compared to low level language.
- Easy to understand, find errors and modify.

Disadvantages

- Middle level language is specific to a particular machine architecture that means it is machine dependent.
- Middle level language needs to be translated into low level language.
- Middle level language executes slower compared to low level language.

High Level Language

- High level language is a computer language which can be understood by the users.
- High level language is very similar to the human languages and have a set of grammar rules that are used to make instructions more easily.
- Every high level language have a set of predefined words known as Keywords and a set of rules known as Syntax to create instructions.
- High level language is more easier to understand for the users but the computer can not understand it. High level language needs to be converted into low level language to make it understandable by the computer. We use **Compiler or interpreter** to convert high level language to low level language.

Example:

Languages like COBOL, FORTRAN, BASIC, C, C++, JAVA etc., are the examples of high level languages. **Advantages**

- Writing instructions in high level language is moreeasier.
- High level language is more readable and understandable.
- The programs created using high level language runs on different machines with little change or no change.
- Easy to understand, create programs, find errors and modify.

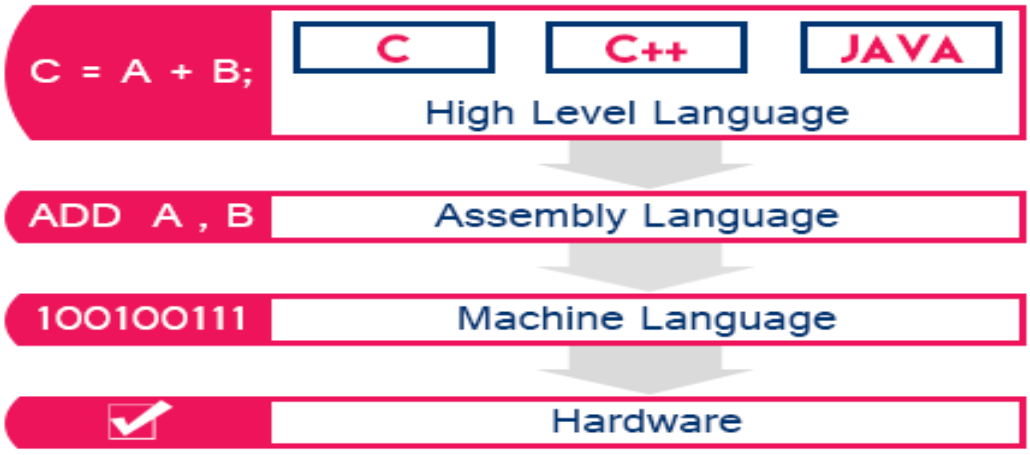
Disadvantages

- High level language needs to be translated to low level language.
- High level language executes slower compared to middle and low level languages.

UNIT I

Understanding Computer Languages

The follwing figure provides few key points related to the computer languages.



From the above figure, we can observe the following key points...

- The programming languages like C, C++, Java etc., are written in High level language which is more comfortable for the developers.
- High level language is more closer to the users.
- Low level language is more closer to the computer. Computer hardware can understand only the low level language (Machine Language).
- The program written in high level language needs to be converted to low level language to make communication between the user and the computer.
- Middle level language is not closer to both user and computer. We can consider it as a combination of both high level language and low level language.

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

Creating and Running C Program

Generally, the programs created using programming languages like C, C++, Java etc., are written using high level language like English. But, computer cannot understand the high level language. It can understand only low level language. So, the program written in high level language needs to be converted into low level language to make it understandable for the computer. This conversion is performed using either Interpreter or Compiler.

Popular programming languages like C, C++, Java etc., use compiler to convert high level language instructions into low level language instructions. Compiler is a program that converts high level language instructions into low level language instructions. Generally, compiler performs two things, first it verifies the program errors, if errors are found, it returns list of errors otherwise it converts the complete code into low level language.

To create and execute C programs in Windows Operating System, we need to install Turbo C software. We use the following steps to create and execute C programs in Windows OS…

Step 1: Creating Source Code

Source code is a file with C programming instructions in high level language. To create source code, we use any text editor to write the program instructions. The instructions written in the source code must follow the C programming language rules. The following steps are used to create source code file in Windows OS…

1. Click on Start button
2. Select Run

3. Type cmd and press Enter
4. Type cd c:\TC\bin in the command prompt and press Enter
5. Type TC press Enter
6. Click on File -> New in C Editor window
7. Type the program
8. Save it as FileName.c (Use shortcut key F2 to save)

Step 2: Compile Source Code (Alt + F9)

Compilation is the process of converting high level language instructions into low level language instructions. We use the shortcut key Alt + F9 to compile a C program in Turbo C.

Compilation is the process of converting high level language instructions into low level language instructions.

Whenever we press Alt + F9, the source file is going to be submitted to the Compiler. On receiving a source file, the compiler first checks for the Errors. If there are any Errors then compiler returns List of Errors, if there are no errors then the source code is converted into object code and stores it as file with .obj extension. Then the object code is given to the Linker. The Linker combines both the object code and specified header file code and generates an Executable file with .exe extension.

Step 3: Executing / Running Executable File (Ctrl + F9)

After completing compilation successfully, an executable file is created with .exe extension. The processor can understand this .exe file content so that it can perform the task specified in the source file.

We use a shortcut key Ctrl + F9 to run a C program. Whenever we press Ctrl + F9, the .exe file is submitted to the CPU. On receiving .exe file, CPU performs the task according to the instruction written in the file. The result generated from the execution is placed in a window called User Screen.

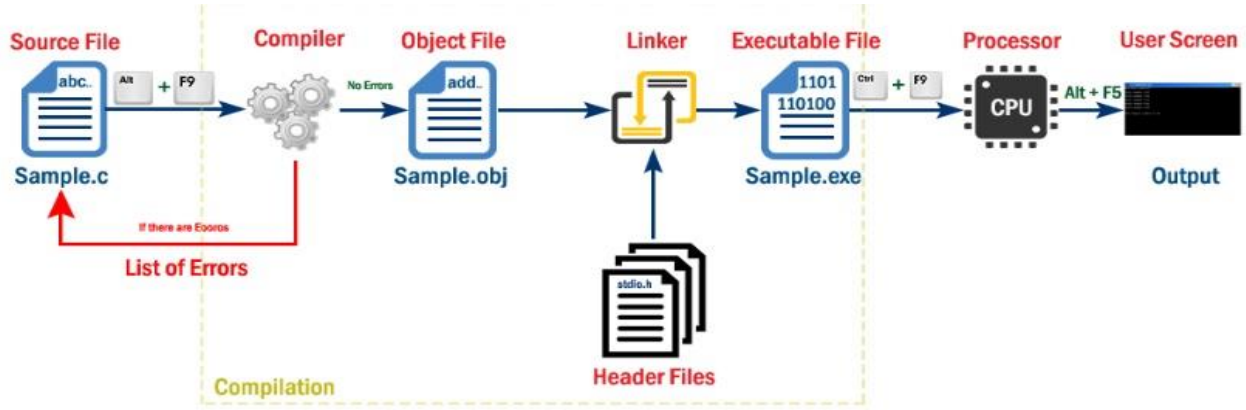
Step 4: Check Result (Alt + F5)

After running the program, the result is placed into User Screen. Just we need to open the User Screen to check the result of the program execution. We use the shortcut key Alt + F5 to open the User Screen and check the result.

UNIT I

Execution Process of a C Program

When we execute a C program it undergoes with following process…



The file which contains c program instructions in high level language is said to be source code. Every c program source file is saved with .c extension, for example Sample.c.

Whenever we press Alt + F9 the source file is submitted to the compiler. Compiler checks for the errors, if there are any errors, it returns list of errors, otherwise generates object code in a file with name Sample.obj and submit it to the linker. Linker combines the code from specified header file into object file and generates executable file as Sample.exe. With this compilation process completes.

Now, we need to Run the executable file (Sample.exe). To run a program we press Ctrl + F9. When we press Ctrl + F9 the executable file is submitted to the CPU. Then CPU performs the task according to the instructions written in that program and place the result into UserScreen.

Then we press Alt + F5 to open UserScreen and check the result of the program.

Important Points

- C program file (Source file) must save with .c extension.
- Compiler converts complete program at a time from high level language to low level language.
- Input to the compiler is .c file and output from the compiler is .exe file, but it also generates .obj file in this process.
- Compiler converts the file only if there are no errors in the source code.
- CPU places the result in User Screen window.

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

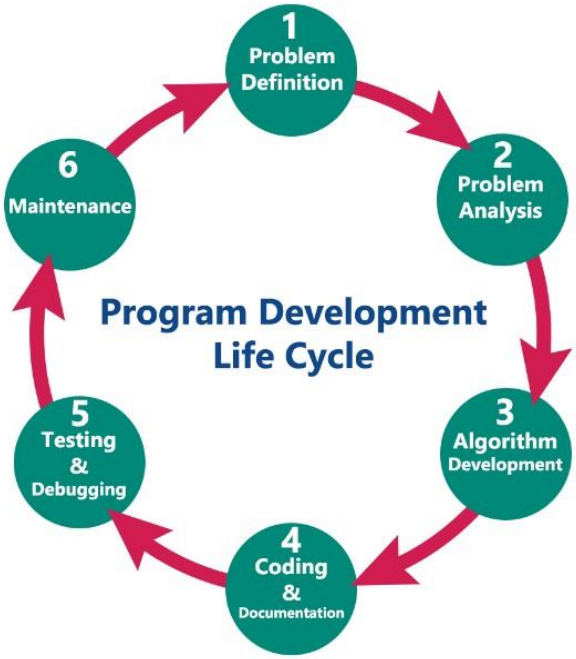
UNIT I

Program Development Life Cycle

When we want to develop a program using any programming language, we follow a sequence of steps. These steps are called phases in program development. The program development life cycle is a set of steps or phases that are used to develop a program in any programming language.

Generally, program development life cycle contains 6 phases, they are as follows….

1. **Problem Definition**
2. **Problem Analysis**
3. **Algorithm Development**
4. **Coding & Documentation**
5. **Testing & Debugging**
6. **Maintenance**



1. Problem Definition

In this phase, we define the problem statement and we decide the boundaries of the problem. In this phase we need to understand the problem statement, what is our requirement, what should be the output of the problem solution. These are defined in this first phase of the program development life cycle.

UNIT I

2. Problem Analysis

In phase 2, we determine the requirements like variables, functions, etc. to solve the problem. That means we gather the required resources to solve the problem defined in the problem definition phase. We also determine the bounds of the solution.

3. Algorithm Development

During this phase, we develop a step by step procedure to solve the problem using the specification given in the previous phase. This phase is very important for program development. That means we write the solution in step by step statements.

4. Coding & Documentation

This phase uses a programming language to write or implement actual programming instructions for the steps defined in the previous phase. In this phase, we construct actual program. That means we write the program to solve the given problem using programming languages like C, C++, Java etc.,

5. Testing & Debugging

During this phase, we check whether the code written in previous step is solving the specified problem or not. That means we test the program whether it is solving the problem for various input data values or not. We also test that whether it is providing the desired output or not.

6. Maintenance

During this phase, the program is actively used by the users. If any enhancements found in this phase, all the phases are to be repeated again to make the enhancements. That means in this phase, the solution (program) is used by the end user. If the user encounters any problem or wants any enhancement, then we need to repeat all the phases from the starting, so that the encountered problem is solved or enhancement is added.

UNIT I

C Background

C is a structured programming language. It is also known as function orientated programming language. C programming language was developed in the year of 1972 by Dennis Ritchie at Bell Laboratories in USA (AT & T).

In the year of 1968, research was started by Dennis Ritchie on programming languages like BCPL, CPL. The main aim of his research was to develop a new language to create an OS called UNIX. After four years of research, a new programming language was created with solutions for drawbacks in languages like BCPL & CPL. In the year of 1972, the new language was introduced with the name "**Traditional C".**

- The name 'c' was selected from the sequence of previous language 'B' (BCPL), because most of the features of 'c' was derived from BCPL (B language).
- The first outcome of the c language was UNIX operating system. The initial UNIX OS was completely developed using 'c' programming language.

The founder of the 'C' language, Dennis Ritchie is known as "Father of C" and also "Father of UNIX".

- C programming language is a very popular programming language, because it is reliable, simple and easy to use and it is the base for almost all the other programming languages.
- The following are the language before 'c' & various versions of 'c'.

1. CPL (Common Programming Language)

The CPL was invented by Martin Richards at the University of Cambridge in the early of 1960's.

UNIT I

2. BCPL (Basic Combined Programming Language)

The BCPL was invented by Martin Richards at the University of Cambridge in the year of 1966. It was a popular programming language at that time. BCPL allows the user, direct access to the computer memory. BCPL is the extension of CPL.

3. B Language

B language is derived from BCPL. It was introduced in the year of 1969 by Ken Thompson and Dennis Ritchie at Bell Laboratory, USA. The B language is similar to the BCPL.

4. C Language

C language is derived from B language. It was introduced in the year of 1972 by Dennis Ritchie at Bell Laboratory, USA. The C language was mainly developed to create a operating system called UNIX. The name C is given based on the previous language B and BCPL. Ninety percent of the UNIX operating system code is written in C language. During 1970's, the C language became very popular programming language. Many universities and organizations began creating their own version of C language for their respective projects. So C language has got many variants at that time. Later it was standardized.

5. ANSI C (C89)

In the year of 1983, the ANSI (American National Standards Institute) formed a committee to frame standard specifications for the C language. In the year of 1989, this committee introduced a standard version of C with the name "ANSI C" with standard library files. The ANSI C is also called as C89 in short form.

6. C90

In the year of 1990, the ANSI C was got ISO (International Organization for Standardization) standardization with the inclusion of few new features like new library files, new processor commands. And it was also added with keywords const, volatile and signed etc... ISO standardized the ANSI C as ISO/IEC 9899:1990. This version is called as C90 in short form.

7. C99

In the year of 1995, many new features were added to the C90 to create a new version of it. This new version of C was got ISO standardization in the year of 1999 with the name ISO/IEC 9899:1999. In the short form it is called as C99. Later C99 became the official standard version of C.

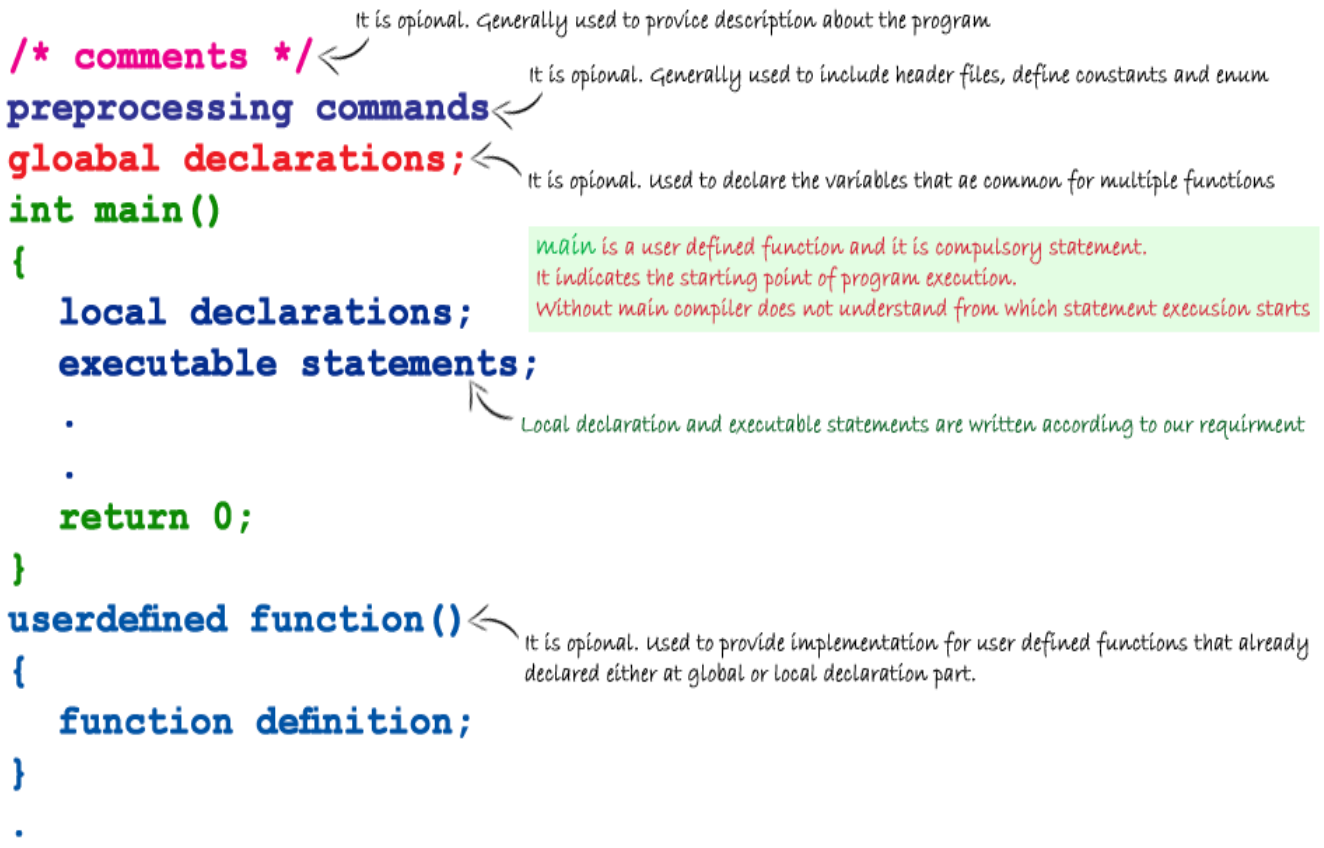
P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

C Program Basics

- C is a structured programming language. Every c program and its statements must be in a particular structure. Every c program has the following general structure...



Line 1: Comments - They are ignored by the compiler

This section is used to provide small description of the program. The comment lines are simply ignored by the compiler that means they are not executed. In C, there are two types of comments.

Single Line Comments: Single line comment begins with **// symbol**. We can write any number of single line comments.

Multiple Lines Comments: Multiple lines comment begins with **/* s**ymbol and ends with ***/.** We can write any number of multiple lines comments in a program.

In a C program, the comment lines are optional. Based on the requirement, we write the comments. All the comment lines in a C program just provide the guidelines to understand the program and its code.

Line 2: Preprocessing Commands

Preprocessing commands are used to include header files and to define constants. We use #include statement to include header file into our program. We use #define statement to define a constant. The preprocessing statements are used according to the requirement. If we don't need any header file, then no need to write #include statement. If we don't need any constant, then no need to write #define statement.

Line 3: Global Declaration

Global declaration is used to define the global variables, which are common for all the functions after its declaration. We also use the global declaration to declare functions. This global declaration is used based on the requirement.

Line 4: intmain()

Every C program must write this statement. This statement (main) specifies the starting point of the C program execution. Here, main is a user defined method which tells the compiler that this is the starting point of the program execution. Here, int is a data type of a value that is going to return to the Operating System after completing the main method execution. If we don't want to return any value, we can use it as void.

Line 5: Open Brace ({)

The open brace indicates the beginning of the block which belongs to the main method. In C program, every block begins with '{' symbol.

Line 6: Local Declaration

In this section, we declare the variables and functions that are local to the function or block in which they are declared. The variables which are declared in this section are valid only within the function or block in which they are declared.

Line 7: Executable statements

In this section, we write the statements which perform tasks like reading data, displaying result, calculations etc., all the statements in this section are written according to the requirement.

Line 9: Closing Brace (})

The close brace indicates the end of the block which belongs to the main method. In C program every block ends with '}' symbol.

Line 10, 11, 12, .: User defined function()

This is the place where we implement the user defined functions. The user defined function

Implementation can also be performed before the main method. In this case, the user defined function need not to be declared. Directly it can be implemented, but it must be before the main method. In a program, we can define as many user defined functions as we want. Every user defined function needs a function call to execute its statements.

UNIT I

C Tokens

- Every C program is a collection of instructions and every instruction is a collection of some individual units.
- Every smallest individual unit of a c program is called token.
- Every instruction in a c program is a collection of tokens.
- Tokens are used to construct c programs and they are said to the basic building blocks of a c program.

In a c program tokens may contain the following...

1. Keywords
2. Identifiers
3. Operators
4. Special Symbols
5. Constants
6. Strings
7. Data values

In a C program, collection of all the keywords, identifiers, operators, special symbols, constants, strings and data values are called as tokens.

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

1. **C Keywords**

- As every language has words to construct statements, C programming also has words with specific meaning which are used to construct c program instructions.
- In C programming language, keywords are **special words** with predefined meaning. Keywords are also known as **reserved words** in C programming language.

- In C programming language, there are **32 keywords**. All the 32 keywords has their own meaning which is already known to the compiler.

Definition:

Keywords are the reserved words with predefined meaning which already known to the compiler

- Whenever C compiler come across a keyword, automatically it understands its meaning.

Properties of Keywords

- All the keywords in C programming language are defined as **lowercase letters** so they must be used only in lowercase letters
- Every keyword has a specific meaning, users **can not change that meaning.**
- Keywords **cannot be used** as **user defined names** like variable, functions, arrays, pointers etc...
- Every keyword in C programming language, represents something or specifies some kind of action to be performed by the compiler.

The following table specifies all the 32 keywords with their meaning...

UNIT I

32 Keywords in C Programming Language with their Meaning

S.No	Keyword	Meaning
1	auto	Used to represent automatic storage class
2	break	Unconditional control statement used to terminate swicth & looping statements
3	case	Used to represent a case (option) in switch statement
4	char	Used to represent character data type
5	const	Used to define a constant
6	continue	Unconditional control statement used to pass the control to the begining of looping statements
7	default	Used to represent a default case (option) in switch statement
8	do	Used to define do block in do-while statement
9	double	Used to present double datatype
10	else	Used to define FALSE block of if statement
11	enum	Used to define enumarated datatypes
12	extern	Used to represent external storage class
13	float	Used to represent floating point datatype
14	for	Used to define a looping statement
15	goto	Used to represent unconditional control statement
16	if	Used to define a conditional control statement
17	int	Used to represent integer datatype
18	long	It is a type modifier that alters the basic datatype
19	register	Used to represent register storage class
20	return	Used to terminate a function execution
21	short	It is a type modifier that alters the basic datatype
22	signed	It is a type modifier that alters the basic datatype
23	sizeof	It is an operator that gives size of the memory of a variable
24	static	Used to create static variables - constants
25	struct	Used to create structures - Userdefined datatypes
26	switch	Used to define switch - case statement
27	typedef	Used to specify temporary name for the datatypes
28	union	Used to create union for grouping different types under a name
29	unsigned	It is a type modifier that alters the basic datatype
30	void	Used to indicate nothing - return value, parameter of a function
31	volatile	Used to creating volatile objects
32	while	Used to define a looping statement

- All the keywords are in lowercase letters
- Keywords can't be used as userdefined name like variable name, function name, lable, etc...
- Keywords are also called as Reserved Words

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

C Identifiers

In C programming language, programmers can specify their own name to variable, array, pointer, function, label etc...

Identifier is a collection of characters which acts as name of variable, function, array, pointer, structure, label etc... In other words, an identifier can be defined as user defined name to identify an entity uniquely in c programming language that name may be of variable name, function name, array name, pointer name, structure name or a label.

Definition:

Identifier is a user defined name of an entity to identify it uniquely during the program execution

Example

intmarks;s

charstudentName[30];

- Here, marks and studentName are identifiers.

Rules for Creating Identifiers:

- An identifier can contain letters (UPPERCASE and lowercase), numeric & underscore symbol only.
- An identifier should not start with numerical value. It can start with a letter or an underscore.
- We should not use any special symbols in between the identifier even whitespace. However, the only underscore symbol is allowed.
- Keywords should not be used as identifiers.
- There is no limit for length of an identifier. However, compiler consider first 31 characters only.
- An identifier must be unique in its scope.

UNIT I

Rules for Creating Identifiers for better programming

- The following are the commonly used rules for creating identifiers for better programming...
- The identifier must be meaningful to describe the entity.
- Since, starting with underscore may create conflict with system names, so we avoid starting an identifier with underscore.
- We, start every identifier with a lowercase letter. If identifier contains more than one word then first word starts with lowercase letter and second word onwards first letter is used as UPPERCASE letter. We can also use an underscore to separate multiple words in an identifier.

Data types

Data used in c program is classified into different types based on its properties.

In c programming language, data type can be defined as a set of values with similar characteristics. All the values in a data type have the same properties.

- **Data types in c programming language are used to specify what kind of value can be stored in a variable**.
- The memory size and type of value of a variable are determined by variable data type.
- In a c program, each variable or constant or array must have a data type and this data type specifies how much memory is to be allocated and what type of values are to be stored in that variable or constant or array. The formal definition of data type is as follows...

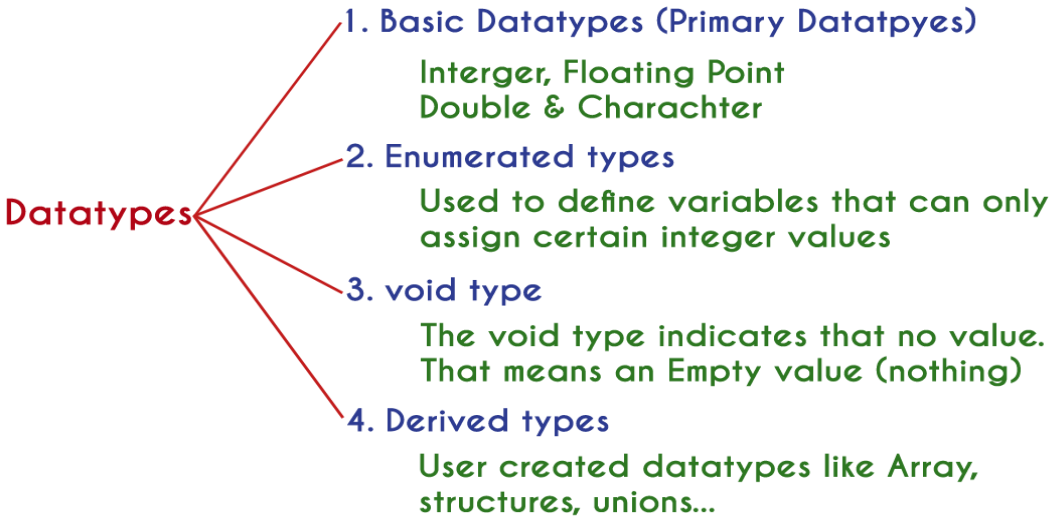
Definition:

Data type is a set of value with predefined characteristics. Data types are used to declare variable, constants, arrays, pointers and functions.

In c programming language, data types are classified as follows...

1. Primary Data types (Basic Data types OR Predefined Data types)
2. Derived Data types (Secondary Data types OR User defined Data types)
3. Enumeration Data types
4. Void Data type

UNIT I

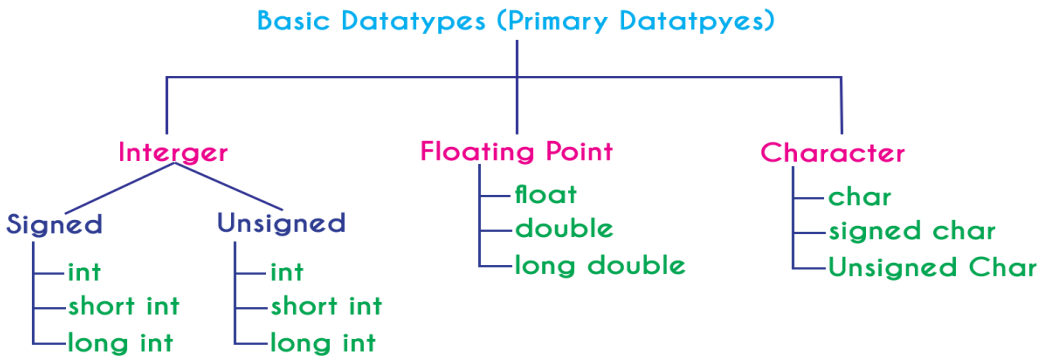


Primary Data types:

- The primary data types in C programming language are the basic data types.
- All the primary data types are already defined in the system. Primary data types are also called as Built-In data types.
-

The following are the primary data types in c programming language...

1. **Integer Data type**
2. **Floating Point Data type**
3. **Double Data type**
4. **Character Data type**



P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

Integer Data type:

- Integer data type is a set of whole numbers.
- Every integer value does not have the decimal value.
- We use the keyword "int" to represent integer data type in c.
- We use the keyword int to declare the variables and to specify return type of a function.
- The integer data type is used with different type modifiers like short, long, signed and unsigned. The following table provides complete details about integer data type.

Type	Size (Bytes)	Range	Specifier
int (signed short int)	2	-32768 to +32767	%d
short int (signed short int)	2	-32768 to +32767	%d
long int (signed long int)	4	-2,147,483,648 to +2,147,483,647	%d
unsigned int (unsigned short int)	2	0 to 65535	%u
unsigned long int	4	0 to 4,294,967,295	%u

Floating Point Data types:

Floating point data types are set of numbers with decimal value. Every floating point value must contain the decimal value.

The floating point data type has two variants...

1. float
2. double

- We use the keyword "float" to represent floating point data type and "double" to represent double data type in c.
- Both float and double are similar but they differ in number of decimal places. The float value contains 6 decimal places whereas double value contains 15 or 19 decimal places. The following table provides complete details about floating point data types.

UNIT I

Type	Size (Bytes)	Range	Specifier
float	4	1.2E - 38 to 3.4E + 38	%f
double	8	2.3E-308 to 1.7E+308	%ld
long double	10	3.4E-4932 to 1.1E+4932	%ld

Character Data type:

Character data type is a set of characters enclosed in single quotations. The following table provides complete details about character data type.

Type	Size (Bytes)	Range	Specifier
char (signed char)	1	-128 to +127	%c
unsigned char	1	0 to 255	%c

The following table provides complete information about all the data types in c programming language...

	Integer	Floating Point	Double	Character
What is it?	Numbers without decimal value	Numbers with decimal value	Numbers with decimal value	Any symbol enclosed in single quotation
Keyword	int	float	double	char
Memory Size	2 or 4 Bytes	4 Bytes	8 or 10 Bytes	1 Byte
Range	-32768 to +32767 (or) 0 to 65535 (Incase of 2 bytes only)	1.2E - 38 to 3.4E + 38	2.3E-308 to 1.7E+308	-128 to + 127 (or) 0 to 255
Type Specifier	%d or %i or %u	%f	%ld	%c or %s
Type Modifier	short, long signed, unsigned	No modifiers	long	signed, unsigned
Type Qualifier	const, volatile	const, volatile	const, volatil	const, volatile

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

Void Data type:

The void data type means nothing or no value. Generally, void is used to specify a function which does not return any value. We also use the void data type to specify empty parameters of a function.

Enumerated Data type:

An enumerated data type is a user-defined data type that consists of integer constants and each integer constant is given a name. The keyword "enum" is used to define enumerated data type.

Derived Data types:

Derived data types are user-defined data types. The derived data types are also called as user defined data types or secondary data types. In c programming language, the derived data types are created using the following concepts...

- **Arrays**
- **Structures**
- **Unions**
- **Enumeration**

Variables

Variables in c programming language are the named memory locations where user can store different values of same data type during the program execution.

That means, variable is a name given to a memory location in which we can store different values of samedata type. In other words a variable can be defined as a storage container to hold values of same data type during the program execution. The formal definition of data type is as follows...

Definition:

Variable is a name given to a memory location where we can store different values of same data type during the program execution.

Every variable in c programming language must be declared in the declaration section before it is used.

UNIT I

Every variable must have a data type that determines the range and type of values to be stored and size of the memory to be allocated.

A variable name may contain letters, digits and underscore symbol.

The following are the rules to specify a variable name...

- Variable name should not start with digit.
- Keywords should not be used as variable names.
- Variable name should not contain any special symbols except underscore (_).
- Variable name can be of any length but compiler considers only the first 31 characters of the variable name.

Syntax:

datatypevariablename:

Example

intnumber;

The above declaration tells to the compiler that allocate 2 bytes of memory with the name number and allows only integer values into that memory location.

Constants in C

In C programming language, a constant is similar to the variable but constant holds only one value during the program execution. That means, once a value is assigned to the constant, that value can't be changed during the program execution. Once the value is assigned to the constant, it is fixed throughout the program. A constant can be defined as follows...

Definition:

A constant is a named memory location which holds only one value throughout the program execution.

In C programmig language, constant can be of any datatype like integer, floating point, character, string and double etc.,

Integer constants

- An integer constant can be a decimal integer or octal integer or hexa decimal integer.
- A decimal integer value is specified as direct integer value whereas octal integer value is prefixed with 'o' and hexa decimal value is prefixed with 'OX'.
- An integer constant can also be unsigned type of integer constant or long type of integer constant. Unsigned integer constant value is suffixed with 'u' and long integer constant value is suffixed with 'l' whereas unsigned long integer constant value is suffixed with 'ul'.

Examples:

125 -----> Decimal Integer Constant

O76 -----> Octal Integer Constant

OX3A ----->Hexa Decimal Integer Constant

50u -----> Unsigned Integer Constant

30l -----> Long Integer Constant

100ul -----> Unsigned Long Integer Constant

Floating Point constants

A floating point constant must contain both integer and decimal parts. Sometimes it may also contain exponent part. When a floating point constant is represented in exponent form, the value must be suffixed with 'e' or 'E'.

Example

The floating point value 3.14 is represented as 3E-14 in exponent form.

Character Constants

A character constant is a symbol enclosed in single quotation. A character constant has a maximum length of one character.

Example

'A','2','+'….

- In C programming language, there are some predefined character constants called **escape sequences.**

Every escape sequence has its own special functionality and every escape sequence is prefixed with '\' symbol. These escape sequences are used in output function called 'printf()'.

String Constants

- A string constant is a collection of characters, digits, special symbols and escape sequences that are enclosed in double quotations.

We define string constant in a single line as follows...

"This is Vamsheedharreddy"

We can define string constant using multiple lines as follows...

" This\

is\

Vamsheedharreddy "

We can also define string constant by separating it with white space as follows...

"This" "is" "Vamsheedharreddy"

All the above three defines the same string constant.

Creating constants in C…..

In c programming language, constants can be created using two concepts...

1. **Using 'const' keyword**
2. **Using '#define' preprocessor**

Using 'const' keyword

We create a constant of any datatype using 'const' keyword. To create a constant, we prefix the variable declaration with 'const' keyword.

The general syntax for creating constant using 'const' keyword is as follows...

constdatatypeconstantName ;

OR

constdatatypeconstantName = value ;

Example:

constint x = 10 ;

Here, 'x' is a integer constant with fixed value 10.

Example Program:

```
#include <stdio.h>

void main()

{

inti = 9 ;

constint x = 10 ;

i = 15 ;

x = 100 ; // creates an error

printf("i = %d\nx = %d", i, x ) ;

}
```

The above program gives an error because we are trying to change the constant variable value (x = 100).

Using '#define' preprocessor

We can also create constants **using '#define' preprocessor** directive.

When we create constant using this preprocessor directive it must be defined at the beginning of the program (because all the preprocessor directives must be written before the **gloabal declaration**).

We use the following syntax to create constant using '#define' preprocessor directive...

#define CONSTANTNAME value

Example

#define PI 3.14

Here, PI is a constant with value 3.14

Example Program:

```
#include <stdio.h>
#defien  PI  3.14
void main(){
int r, area ;
printf("Please enter the radius of circle : ") ;
scanf("%d", &r) ;
area = PI * (r * r) ;
printf("Area of the circle = %d", area) ;
}
```

UNIT I

Operators in C

An operator is a symbol used to perform mathematical and logical operations in a program. That means an operator is a special symbol that tells to the compiler to perform mathematical or logical operations. C programming language supports rich set of operators that are classified as follows...

1. **Arithematic Operators**
2. **Relational Operators**
3. **Logical Operators**
4. **Increment & Decrement Operators**
5. **Assignment Operators**
6. **Bitwise Operators**
7. **Conditional Operator**
8. **Special Operators**
9.

Arithematic Operators (+, -, *, /, %)

The arithematic operators are the symbols that are used to perform basic mathematical operations like addition, subtraction, multiplication, division and percentage modulo. The following table provides information about arithematic operators...

Operator	Meaning	Example
+	Addition	10 + 5 = 15
-	Subtraction	10 - 5 = 5
*	Multiplication	5 * 10 = 50
/	Division	10 / 5 = 2
%	Remaider of Division	5 % 2 = 1

⇒The addition operator can be used with numerical data types and character data type. When it is used with numerical values, it performs mathematical addition and when it is used with character data type values, it performs concatination (appending).

⇒The remainder of division operator is used with integer data type only.

UNIT I

Relational Operators (<, >, <=, >=, ==, !=)

The relational operators are the symbols that are used to compare two values.

That means, the relational operators are used to check the relationship between two values. Every relational operator has two results TRUE or FALSE. In simple words, the relational operators are used to define conditions in a program. The following table provides information about relational operators...

Operator	Meaning	Example
<	Returns TRUE if first value is smaller than second value otherwise returns FALSE	10 < 5 is FALSE
>	Returns TRUE if first value is larger than second value otherwise returns FALSE	10 > 5 is TRUE
<=	Returns TRUE if first value is smaller than or equal to second value otherwise returns FALSE	10 <= 5 is FALSE
>=	Returns TRUE if first value is larger than or equal to second value otherwise returns FALSE	10 >= 5 is TRUE
==	Returns TRUE if both values are equal otherwise returns FALSE	10 == 5 is FALSE
!=	Returns TRUE if both values are not equal otherwise returns FALSE	10 != 5 is TRUE

Logical Operators (&&, ||, !)

The logical operators are the symbols that are used to combine multiple conditions into one condition. The following table provides information about logical operators...

Operator	Meaning	Example
&&	**Logical AND** - Returns TRUE if all conditions are TRUE otherwise returns FALSE	10 < 5 && 12 > 10 is FALSE
||	**Logical OR** - Returns FALSE if all conditions are FALSE otherwise returns TRUE	10 < 5 || 12 > 10 is TRUE
!	**Logical NOT** - Returns TRUE if condition is FLASE and returns FALSE if it is TRUE	!(10 < 5 && 12 > 10) is TRUE

⇒ Logical AND - Returns TRUE only if all conditions are TRUE, if any of the conditions is FALSE then complete condition becomes FALSE.

⇒ Logical OR - Returns FALSE only if all conditions are FALSE, if any of the conditions is TRUE then complete condition becomes TRUE.

UNIT I

Increment & Decrement Operators (++ & --)

The increment and decrement operators are called as **unary operators** because, both needs only one operand.

The increment operators **adds one** to the existing value of the operand and **the decrement operatorsubtracts one** from the existing value of the operand. The following table provides information about increment and decrement operators...

Operator	Meaning	Example
++	**Increment** - Adds one to existing value	int a = 5; a++; ⇒ a = 6
--	**Decrement** - Subtracts one from existing value	int a = 5; a--; ⇒ a = 4

The increment and decrement operators are used infront of the operand (++a) or after the operand (a++).

- **If it is used infront of the operand, we call it as pre-increment or pre-decrement and**
- **if it is used after the operand, we call it as post-increment or post-decrement.**
-

Pre-Increment or Pre-Decrement

In case of pre-increment, the value of the variable is increased by one before the expression evaluation. In case of pre-decrement, the value of the variable is decreased by one before the expression evaluation. That means, when we use pre increment or pre decrement, first the value of the variable is incremented or decremented by one, then modified value is used in the expression evaluation

Example Program:

```
#include <stdio.h>

void main(){

inti = 5,j;

    j = ++i; // Pre-Increment

printf("i = %d, j = %d",i,j);

}
```

Post-Increment or Post-Decrement

In case of post-increment, the value of the variable is increased by one after the expression evaluation. In case of post-decrement, the value of the variable is decreased by one after the expression evaluation. That means, when we use post-increment or post-decrement, first the expression is evaluated with existing value, then the value of the variable is incremented or decremented by one.

Example Program:

```
#include <stdio.h>

void main(){

inti = 5,j;

    j = i++;  // Post-Increment

printf("i = %d, j = %d",i,j);

}
```

Assignment Operators (=, +=, -=, *=, /=, %=)

The assignment operators are used to assign right hand side value (Rvalue) to the left hand side variable (Lvalue).

The assignment operator is used in different variants along with arithematic operators. The following table describes all the assignment operators in C programming language.

Operator	Meaning	Example
=	Assign the right hand side value to left hand side variable	A = 15
+=	Add both left and right hand side values and store the result into left hand side variable	A += 10 ⇒ A=A+10
-=	Subtract right hand side value from left hand side variable value and store the result into left hand side variable	A -= B ⇒ A=A-B
*=	Multiply right hand side value with left hand side variable value and store the result into left hand side variable	A *= B ⇒ A=A*B
/=	Divide left hand side variable value with right hand side variable value and store the result into left hand side variable	A /= B ⇒ A=A/B
%=	Divide left hand side variable value with right hand side variable value and store the remainder into left hand side variable	A %= B ⇒ A=A%B

UNIT I

Bitwise Operators (&, |, ^, ~, >>, <<)

The bitwise operators are used to perform bit level operations in c programming language. When we use the bitwise operators, the operations are performed based on the binary values. The following table describes all the bitwise operators in C programming language.

Let us consider two variables A and B as A = 25 (11001) and B = 20 (10100)

Operator	Meaning	Example
&	the result of **Bitwise AND** is 1 if all the bits are 1 otherwise it is 0	A & B ⇒ 16 (10000)
|	the result of **Bitwise OR** is 0 if all the bits are 0 otherwise it is 1	A | B ⇒ 29 (11101)
^	the result of **Bitwise XOR** is 0 if all the bits are same otherwise it is 1	A ^ B ⇒ 13 (01101)
~	the result of **Bitwise once complement** is nagation of the bit (Flipping)	~A ⇒ 6 (00110)
<<	the **Bitwise left shift** operator shifts all the bits to the left by specified number of positions	A<<2 ⇒ 100 (1100100)
>>	the **Bitwise right shift** operator shifts all the bits to the right by specified number of positions	A>>2 ⇒ 6 (00110)

Conditional Operator (?:)

The conditional operator is also called as **ternary operator** because it requires three operands. This operator is used for decision making.

In this operator, first we verify a condition, then we perform one operation out of the two operations based on the condition result. If the condition is TRUE the first option is performed, if the condition is FALSE the second option is performed. The conditional operator is used with the following syntax...

Condition ? TRUE Part : FALSE Part ;

Example:

A = (10<15) ?100 : 200 ; ⇒ A value is 100

UNIT I

Special Operators (sizeof, pointer, comma, dot etc.)

The following are the special operators in c programming language...

sizeof operator :

This operator is used to find the size of the memory (in bytes) allocated for a variable. This operator is used with the following syntax...

sizeof(variableName);

Example

sizeof(A); ⇒ result is 2 if A is an integer

Pointer operator (*):

This operator is used to define pointer variables in c programming language.

Comma operator (,):

This operator is used to separate variables while they are declaring, separate the expresions in function calls etc..

Dot operator (.):

This operator is used to access members of sructure or union.

UNIT I

Condition And Control Statements

What is Decision Making Statement?

In c programming language, the program execution flow is, line by line from top to bottom. That means the c program is executed line by line from the main method. But this type of execution flow may not be suitable for all the program solutions. Sometimes, we make some decisions or we may skip the execution of one or more lines of code. Consider a situation, where we write a program to check whether a student has passed or failed in a particular subject. Here, we need to check whether the marks are greater than the pass marks or not. If marks are greater, then we take the decision that the student has passed otherwise failed. To solve such kind of problems in c we use the statements called decision making statements.

Definition:

Decision making statements are the statements that are used to verify a given condition and decides whether a block of statements gets executed or not based on the condition result.

In c programming language, there are two decision making statements they are as follows...

1. **if statement**
2. **switch statement**

if statement in c:

In c, if statement is used to make decisions based on a condition. The if statement verifies the given condition and decides whether a block of statements are executed or not based on the condition result. In c,

if statement is classified into four types as follows...

1. **Simple if statement**
2. **if - else statement**
3. **Nested if statement**
4. **if-else-if statement (if-else ladder)**

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

Simple if statement

Simple if statement is used to verify the given condition and executes the block of statements based on the condition result.

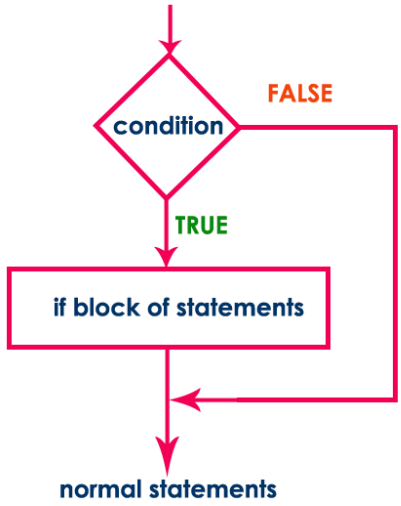
The simple if statement evaluates specified condition. If it is TRUE, it executes the next statement or block of statements. If the condition is FALSE, it skips the execution of the next statement or block of statements.

The general syntax and execution flow of the simple if statement is as follows...

Syntax

```
if ( condition )
{
   ....
   block of statements;
   ....
}
```

Execution flow diagram



Simple if statement is used when we have only one option that is executed or skipped based on a condition.

Example Program | Test whether given number is divisible by 5.

```
#include <stdio.h>
#include<conio.h>
void main(){
int n ;
clrscr() ;
printf("Enter any integer number: ") ;
scanf("%d", &n) ;
if ( n%5 == 0 )
printf("Given number is divisible by 5\n") ;
printf("statement does not belong to if!!!") ;
}
```

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

Output 1:

Enter any integer number: 100

Given number is divisible by 5

Statement does not belong to if!!!

Output 2:

Enter any integer number: 99

Statement does not belong to if!!!

if - else statement

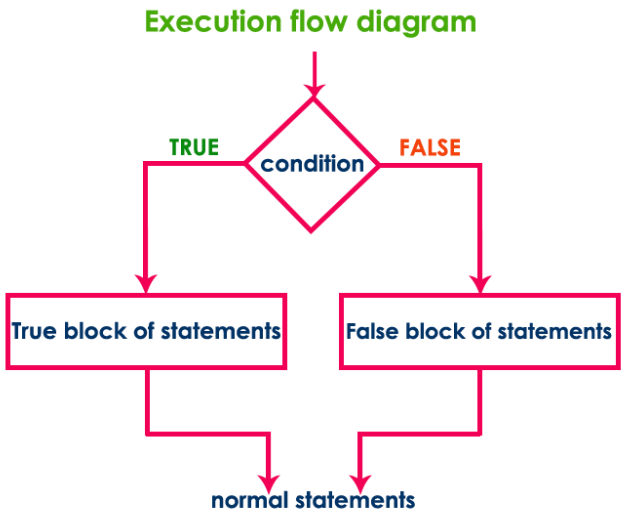
The if - else statement is used to verify the given condition and executes only one out of the two blocks of statements based on the condition result.

The if-else statement evaluates the specified condition. If it is TRUE, it executes a block of statements (True block). If the condition is FALSE, it executes another block of statements (False block).

The general syntax and execution flow of the if-else statement is as follows...

Syntax

```
if ( condition )
{
   ....
   True block of statements;
   ....
}
else
{
   ....
   False block of statements;
   ....
}
```



The if-else statement is used when we have two options and only one option has to be executed based on a condition result (TRUE or FALSE).

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

Example Program | Test whether given number is even or odd.

```
#include <stdio.h>
#include<conio.h>
void main(){
int n ;
clrscr() ;
printf("Enter any integer number: ") ;
scanf("%d", &n) ;
if ( n%2 == 0 )
printf("Given number is EVEN\n") ;
else
printf("Given number is ODD\n") ;
}
```

Output 1:

Enter any integer number: 100

Given number is EVEN

Output 2:

Enter any integer number: 99

Given number is ODD.

Nested if statement

Writing a if statement inside another if statement is called nested if statement. The general syntax of the nested if statement is as follows...

Syntax

```
if ( condition1 )
{
   if ( condition2 )
   {
      ....
      True block of statements 1;
   }
   ....
}
else
{
    False block of condition1;
}
```

The nested if statement can be defined using any combination of simple if & if-else statements.

Example Program | Test whether given number is even or odd if it is below 100.

```
#include <stdio.h>
#include<conio.h>
void main(){
int n ;
clrscr() ;
printf("Enter any integer number: ") ;
scanf("%d", &n) ;
if ( n < 100 )
  {
printf("Given number is below 100\n") ;
if( n%2 == 0)
printf("And it is EVEN") ;
else
printf("And it is ODD") ;
  }
else
printf("Given number is not below 100") ;
}
```

Output 1:
Enter any integer number: 55
Given number is below 100
And it is ODD

Output 2:
Enter any integer number: 999
Given number is not below 100

if - else - if statement (if-else ladder)

Writing a if statement inside else of a if statement is called if - else - if statement. The general syntax of the if-else-if statement is as follows...

Syntax

```
if ( condition1 )
{
    ....
    True block of statements1;
    ....
}
else if ( condition2 )
{
    False block of condition1;
    &
    True block of condition2
}
```

The if-else-if statement can be defined using any combination of simple if & if-else statements.

Example Program | Find the largest of three numbers.

```
#include <stdio.h>
#include<conio.h>
void main(){
int a, b, c ;
clrscr() ;

printf("Enter any three integer numbers: ") ;
scanf("%d%d%d", &a, &b, &c) ;

if( a>=b && a>=c)
printf("%d is the largest number", a) ;

else if (b>=a && b>=c)
printf("%d is the largest number", b) ;

else
printf("%d is the largest number", c) ;
}
```

Output 1:

```
Enter any three integer numbers: 55 60 20
60 is the largest number
```

UNIT I

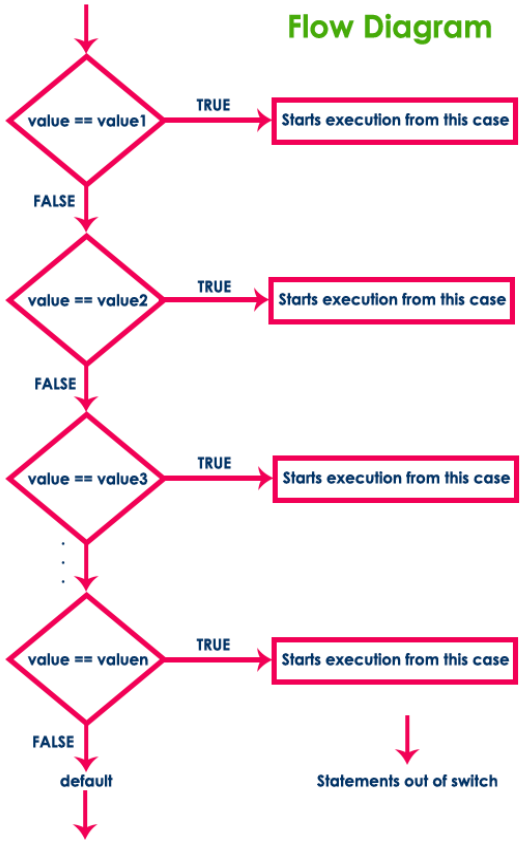
switch statement in C

Consider a situation in which we have more number of options out of which we need to select only one option that is to be executed. Such kind of problems can be solved using nested if statement. But as the number of options increases, the complexity of the program also gets increased. This type of problems can be solved very easily using switch statement. Using switch statement, one can select only one option from more number of options very easily. In switch statement, we provide a value that is to be compared with a value associated with each option. Whenever the given value matches with the value associated with an option, the execution starts from that option. In switch statement every option is defined as a case.

The switch statement has the following syntax and execution flow diagram...

Syntax

```
switch ( expression or value )
{
      case  value1:  set of statements;
                     ....
      case  value2:  set of statements;
                     ....
      case  value3:  set of statements;
                     ....
      case  value4:  set of statements;
                     ....
      case  value5:  set of statements;
                     ....
      .
      .
      default: set of statements;
}
```



- The switch statement contains one or more number of cases and each case has a value associated with it.
- At first switch statement compares the first case value with the switchValue, if it gets matched the execution starts from the first case.
- If it doesn't match the switch statement compares the second case value with the switchValue and if it is matched the execution starts from the second case.
- This process continues until it finds a match.
- If no case value matches with the switchValue specified in the switch statement, then a special case called default is executed.

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

- When a case value matches with the switchValue, the execution starts from that particular case. This execution flow continues with next case statements also.
- To avoid this, we use "break" statement at the end of each case. That means the break statement is used to terminate the switch statement. However it is optional.

Example Program | Display pressed digit in words.

```
#include <stdio.h>
#include<conio.h>
void main(){
int n ;
clrscr() ;

printf("Enter any digit: ") ;
scanf("%d", &n) ;

switch( n )
  {
        case 0: printf("ZERO") ;
        break ;
        case 1: printf("ONE") ;
        break ;
        case 2: printf("TWO") ;
        break ;
        case 3: printf("THREE") ;
        break ;
        case 4: printf("FOUR") ;
        break ;
        case 5: printf("FIVE") ;
        break ;
        case 6: printf("SIX") ;
        break ;
        case 7: printf("SEVEN") ;
        break ;
        case 8: printf("EIGHT") ;
        break ;
        case 9: printf("NINE") ;
        break ;
        default: printf("Not a Digit") ;
  }
getch() ;
}
```

UNIT I

Output 1:
Enter any digit: 5
FIVE
Output 2:
Enter any digit: 15
Not a Digit

Looping Statements:

Consider a situation in which we execute a single statement or block of statements repeatedly for required number of times. Such kind of problems can be solved using looping statements in C. For example, assume a situation where we print a message for 100 times. If we want to perform that task without using looping statements, we have to either write 100 printf statements or we have to write the same message for 100 times in a single printf statement. Both are complex methods. The same task can be performed very easily using looping statements.

Definition:

The looping statements are used to execute a single statement or block of statements repeatedly until the given condition is FALSE.

C language provides three looping statements...

- **while statement**
- **do-while statement**
- **for statement**

while Statement

The while statement is used to execute a single statement or block of statements repeatedly as long as the given condition is TRUE. The while statement is also known as Entry control looping statement.

The while statement has the following syntax...

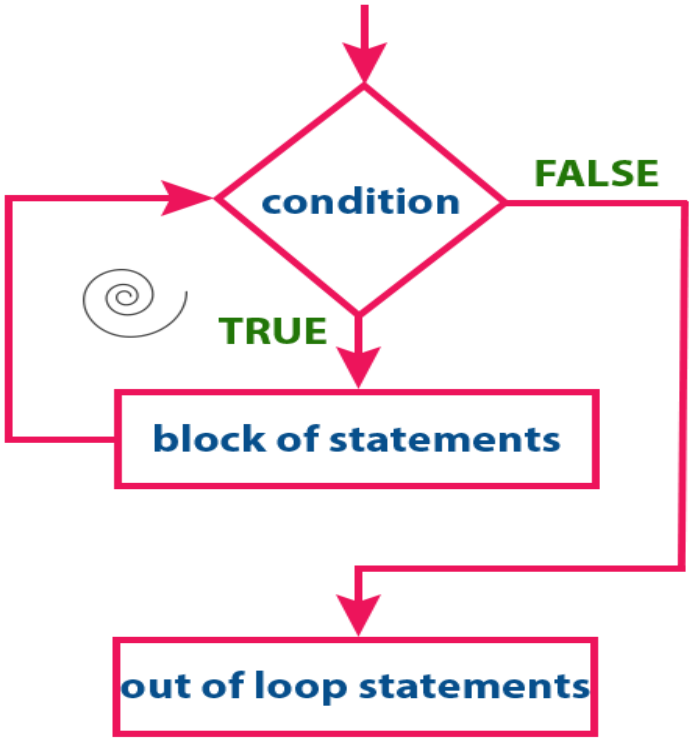
Syntax:

```
while( condition )
{
    ...
    block of statements;
    ...
}
```

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

UNIT I

The while statement has the following execution flow diagram...



At first, the given condition is evaluated. If the condition is TRUE, the single statement or block of statements gets executed. Once the execution gets completed the condition is evaluated again. If it is TRUE, again the same statements gets executed. The same process is repeated until the condition is evaluated to FALSE. Whenever the condition is evaluated to FALSE, the execution control moves out of the while block.

Example Program | Program to display even numbers upto 10.

```
#include <stdio.h>
#include<conio.h>
void main(){
int n = 0;
clrscr() ;
printf("Even numbers upto 10\n");

while( n <= 10 )
  {
if( n%2 == 0)
printf("%d\t", n) ;
    n++ ;
  }
```

```
getch() ;
}
```

Output 1:
Even numbers upto 10
0 2 4 6 8 10

do-while Statement in C

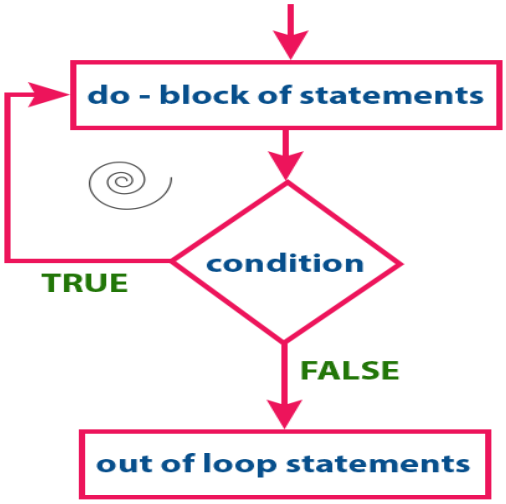
The do-while statement is used to execute a single statement or block of statements repeatedly as long as given the condition is TRUE. The do-while statement is also known as Exit control looping statement.

The do-while statement has the following syntax...

Syntax:

```
do
{
    ...
    block of statements;
    ...
} while( condition ) ;
```

The do-while statement has the following execution flow diagram...



- At first, the single statement or block of statements which are defined in do block are executed. After execution of do block, the given condition gets evaluated.
- If the condition is evaluated to TRUE, the single statement or block of statements of do block are executed again. Once the execution gets completed again the condition is evaluated. If it is TRUE, again the same statements are executed.
- The same process is repeated until the condition is evaluated to FALSE. Whenever the condition is evaluated to FALSE, the execution control moves out of the while block.

Example Program | Program to display even numbers upto 10.

```
#include <stdio.h>
#include<conio.h>
void main(){
int n = 0;
clrscr() ;
printf("Even numbers upto 10\n");

do
  {
if( n%2 == 0)
printf("%d\t", n) ;
    n++ ;
}while( n <= 10 ) ;

getch() ;
}
```

Output 1:
Even numbers upto 10
0 2 4 6 8 10

UNIT I

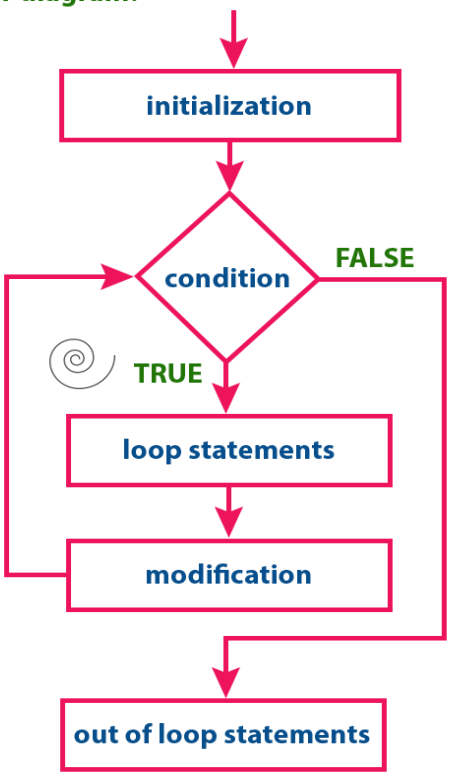
for Statement in C

The for statement is used to execute a single statement or a block of statements repeatedly as long as the given condition is TRUE. The for statement has the following syntax and execution flow diagram..

Syntax:

```
for( initialization ; condition ; modification )
{
    ...
    block of statements;
    ...
}
```

Execution flow diagram:



- At first, the for statement executes initialization followed by condition evaluation.
- If the condition is evaluated to TRUE, the single statement or block of statements of for statement are executed.
- Once the execution gets completed, the modification statement is executed and again the condition is evaluated. If it is TRUE, again the same statements are executed.
- The same process is repeated until the condition is evaluated to FALSE. Whenever the condition is evaluated to FALSE, the execution control moves out of the for block.

P.VAMSHEEDHAR REDDY
(Asst.Prof,CSE DEPT)

(If any data needed to add into this material please write to : pvamsheedharreddy@gmail.com)

UNIT I

Example Program | Program to display even numbers upto 10.

```
#include <stdio.h>
#include<conio.h>
void main(){
int n ;
clrscr() ;
printf("Even numbers upto 10\n");

for( n = 0 ; n <= 10 ; n++ )
   {
if( n%2 == 0)
printf("%d\t", n) ;
   }

getch() ;
}
```

Output 1:
Even numbers upto 10
0 2 4 6 8 10