**Introduction to C**

 C is a programming language developed at AT & T’s Bell Laboratories of USA in 1972. It was designed and written by a man named Dennis Ritchie. In the late seventies C began to replace the more familiar languages of that time like PL/I, ALGOL, etc

 ANSI C standard emerged in the early 1980s, this book was split into two titles: The original was still called Programming in C, and the title that covered ANSI C was called Programming in ANSI C. This was done because it took several years for the compiler vendors to release their ANSI C compilers and for them to become ubiquitous. It was initially designed for programming UNIX operating system. Now the software tool as well as the C compiler is written in C. Major parts of popular operating systems like Windows, UNIX, Linux is still written in C. This is because even today when it comes to performance (speed of execution) nothing beats C. Moreover, if one is to extend the operating system to work with new devices one needs to write device driver programs. These programs are exclusively written in C. C seems so popular is because it is reliable, simple and easy to use. often heard today is – “C has been already superceded by languages like C++, C# and Java.

**PROGRAMME LEARNING**

There is a close analogy between learning English language and learning C language. The classical method of learning English is to first learn the alphabets used in the language, then learn to combine these alphabets to form words, which in turn are combined to form sentences and sentences are combined to form paragraphs. Learning C is similar and easier. Instead of straight-away learning how to write programs, we must first know what alphabets, numbers and special symbols are used in C, then how using them constants, variables and keywords are constructed, and finally how are these combined to form an instruction. A group of instructions would be combined later on to form a program.



**#WHAT IS COMPUTER PROGRAMME?**

a computer program is just a collection of the instructions necessary to solve a specific problem. The basic operations of a computer system form what is known as the computer’s instruction set. And the approach or method that is used to solve the problem is known as an algorithm

**# TYPES OF PROGRAMMING LANGUAGE :**

programming language concern these are of two types.

1) Low level language

2) High level language

**# Low level language:**

 Low level languages are machine level and assembly level language. In machine level language computer only understand digital numbers i.e. in the form of 0 and 1. So, instruction given to the computer is in the form binary digit, which is difficult to implement instruction in binary code. This type of program is not portable, difficult to maintain and also error prone. The assembly language is on other hand modified version of machine level language. Where instructions are given in English like word as ADD, SUM, MOV etc. It is easy to write and understand but not understand by the machine. So the translator used here is assembler to translate into machine level.

**# High level language:**

These languages are machine independent, means it is portable. The language in this category is Pascal, Cobol, Fortran etc. High level languages are understood by the machine. So it need to translate by the translator into machine level.

**# Translator:**

A translator is software which is used to translate high level language as well as low level language in to machine level language

**# Types of Translator:**

Three types of translator are there:

1. Compiler
2. Interpreter
3. Assembler

Compiler and interpreter are used to convert the high level language into machine level language. The program written in high level language is known as source program and the corresponding machine level language program is called as object program. Both compiler and interpreter perform the same task but there working is different. Compiler read the program at-a-time and searches the error and lists them. If the program is error free then it is converted into object program. When program size is large then compiler is preferred. Whereas interpreter read only one line of the source code and convert it to object code. If it check error, statement by statement and hence of take more time.

**Integrated Development Environments (IDE)**:

 The process of editing, compiling, running, and debugging programs is often managed by a single integrated application known as an Integrated Development Environment, or IDE for short. An IDE is a windows-based program that allows us to easily manage large software programs, edit files in windows, and compile, link, run, and debug programs.

**# Structure of C Language program:**

1 ) Comment line

 2) Preprocessor directive

3 ) Global variable declaration

4) main function( )

 {

Local variables;

Statements;

 }

 User defined function

}

}

**Comment line**

It indicates the purpose of the program. It is represented as /\*……………………………..\*/ Comment line is used for increasing the readability of the program. It is useful in explaining the program and generally used for documentation. It is enclosed within the decimeters. Comment line can be single or multiple line but should not be nested. It can be anywhere in the program except inside string constant & character constant.

**Preprocessor Directive:**

#include tells the compiler to include information about the standard input/output library. It is also used in symbolic constant such as #define PI 3.14(value). The stdio.h (standard input output header file) contains definition &declaration of system defined function such as printf( ), scanf( ), pow( ) etc. Generally printf() function used to display and scanf() function used to read value

**Global Declaration:**

This is the section where variable are declared globally so that it can be access by all the functions used in the program. And it is generally declared outside the function :

 **main()**

 It is the user defined function and every function has one main() function from where actually program is started and it is encloses within the pair of curly braces.

 The main( ) function can be anywhere in the program but in general practice it is placed in the first position.

 Syntax :

main()

 { ……..

 ……..

…….. }

The main( ) function return value when it declared by data type as

 int main( )

{

return 0

 }

The main function does not return any value when void (means null/empty) as void main(void ) or void

main()

 {

printf (“C language”);

}

 Output: C language

 The program execution start with opening braces and end with closing brace.

 And in between the two braces declaration part as well as executable part is mentioned. And at the end of each line, the semi-colon is given which indicates statement termination.

**/\*First c program with return statement\*/**

 #include <stdio.h>

int main (void)

 {

printf ("welcome to c Programming language.\n");

 return 0;

}

Output: welcome to c programming language.

**# Steps for Compiling and executing the Programs :**

****

**# Character set**

 A character denotes any alphabet, digit or special symbol used to represent information. Valid alphabets, numbers and special symbols allowed in C are

****

**# Identifiers**

 Identifiers are user defined word used to name of entities like variables, arrays, functions, structures etc. Rules for naming identifiers are:

 1) name should only consists of alphabets (both upper and lower case), digits and underscore (\_) sign.

 2) first characters should be alphabet or underscore

 3) name should not be a keyword

 4) since C is a case sensitive, the upper case and lower case considered differently, for example code, Code, CODE etc. are different identifiers.

5) identifiers are generally given in some meaningful name such as value, net\_salary, age, data etc.

 An identifier name may be long, some implementation recognizes only first eight characters, most recognize 31 characters. ANSI standard compiler recognize 31 characters. Some invalid identifiers are 5cb, int, res#, avg no etc

**# Keyword**

 There are certain words reserved for doing specific task, these words are known as reserved word or keywords. These words are predefined and always written in lower case or small letter. These keywords cann’t be used as a variable name as it assigned with fixed meaning. Some examples are int, short, signed, unsigned, default, volatile, float, long, double, break, continue, typedef, static, do, for, union, return, while, do, extern, register, enum, case, goto, struct, char, auto, const etc

**# Data Types**

 Data types refer to an extensive system used for declaring variables or functions of different types before its use. The type of a variable determines how much space it occupies in storage and how the bit pattern stored is interpreted. The value of a variable can be changed any time.

C has the following 4 types of data types

basic built-in data types: int, float, double, char

Enumeration data type: enum

Derived data type: pointer, array, structure, union

 Void data type: void

****

**# Constants:**

 Constant is a any value that cannot be changed during program execution. In C, any number, single character, or character string is known as a constant. A constant is an entity that doesn’t change whereas a variable is an entity that may change. For example, the number 50 represents a constant integer value.

The character string "Programming in C is fun.\n" is an example of a constant character string.

 C constants can be divided into two major categories:

 Primary Constants

 Secondary Constants

 These constants are further categorized as



**Numeric constant:**

 Numeric constant consists of digits. It required minimum size of 2 bytes and max 4 bytes. It may be positive or negative but by default sign is always positive. No comma or space is allowed within the numeric constant and it must have at least 1 digit. The allowable range for integer constants is -32768 to 32767. Truly speaking the range of an Integer constant depends upon the compiler. For a 16-bit compiler like Turbo C or Turbo C++ the range is –32768 to 32767. For a 32-bit compiler the range would be even greater. Mean by a 16-bit or a 32- bit compiler, what range of an Integer constant has to do with the type of compiler.

It is categorized a integer constant and real constant. An integer constants are whole number which have no decimal point.

Types of integer constants are:

 **Decimal constant**: 0-------9(base 10)

**Octal constant:** 0-------7(base 8)

**Hexa decimal constant**: 0----9, A------F(base 16) \

**Real constant** is also called floating point constant. To construct real constant we must follow the rule of , -real constant must have at least one digit.

 -It must have a decimal point.

 -It could be either positive or negative. -Default sign is positive.

 -No commas or blanks are allowed within a real constant.

 Ex.: +325.34 426.0 -32.76

**Character constant :**

Character constant represented as a single character enclosed within a single quote.

These can be single digit, single special symbol or white spaces such as ‘9’,’c’,’$’, ‘ ’ etc. Every character constant has a unique integer like value in machine’s character code as if machine using ASCII (American standard code for information interchange).

Some numeric value associated with each upper and lower case alphabets and decimal integers are as: A------------ Z ASCII value

(65-90) a-------------z ASCII value

 (97-122) 0-------------9 ASCII value

 ; ASCII value (59)

**String constant** :

Set of characters are called string and when sequence of characters are enclosed within a double quote (it may be combination of all kind of symbols) is a string constant. String constant has zero, one or more than one character and at the end of the string null character(\0) is automatically placed by compiler. Some examples are “,sarathina” , “908”, “3”,” ”, “A” etc. In C although same characters are enclosed within single and double quotes it represents different meaning such as “A” and ‘A’ are different because first one is string attached with null character at the end but second one is character constant with its corresponding ASCII value is 65.

**Symbolic constant:**

 Symbolic constant is a name that substitute for a sequence of characters and, characters may be numeric, character or string constant. These constant are generally defined at the beginning of the program as #define name value , here name generally written in upper case for example

 #define MAX 10

 #define CH ‘b’

#define NAME “sony

**Variables**:

 Variable is a data name which is used to store some data value or symbolic names for storing program computations and results. The value of the variable can be change during the execution. The rule for naming the variables is same as the naming identifier. Before used in the program it must be declared. Declaration of variables specify its name, data types and range of the value that variables can store depends upon its data types.

 Syntax: int a; char c; float f;

**Variable initialization:**

When we assign any initial value to variable during the declaration, is called initialization of variables. When variable is declared but contain undefined value then it is called garbage value. The variable is initialized with the assignment operator such as

 Data type variable name=constant;

Example:

int a=20; Or

int a;

 a=20;

**Expressions**:

 An expression is a combination of variables, constants, operators and function call. It can be arithmetic, logical and relational for

example:-

 int z= x+y // arithmatic expression

 a>b //relational

a==b // logical

 func(a, b) // function call

**Operator**:

 This is a symbol use to perform some operation on variables, operands or with the constant. Some operator required 2 operand to perform operation or Some required single operation.

 Several operators are there those are, arithmetic operator, assignment, increment , decrement, logical, conditional, comma, size of , bitwise and others

1. **Arithmatic Operator**

 This operator used for numeric calculation. These are of either Unary arithmetic operator, Binary arithmetic operator. Where Unary arithmetic operator required 25 \*Under revision only one operand such as +,-, ++, --,!, tiled.

 And these operators are addition, subtraction, multiplication, division. Binary arithmetic operator on other hand required two operand and its operators are +(addition), -(subtraction), \*(multiplication), /(division), %(modulus). But modulus cannot applied with floating point operand as well as there are no exponent operator in c.

Unary (+) and Unary (-) is different from addition and subtraction.

1. **Assignment Operator**

A value can be stored in a variable with the use of assignment operator. The assignment operator(=) is used in assignment statement and assignment expression. Operand on the left hand side should be variable and the operand on the right hand side should be variable or constant or any expression. When variable on the left hand side is occur on the right hand side then we can avoid by writing the compound statement.

For example,

 int x= y;

int Sum=x+y+z;

1. **Increment and Decrement**

The Unary operator ++, --, is used as increment and decrement which acts upon single operand. Increment operator increases the value of variable by one .Similarly decrement operator decrease the value of the variable by one. And these operator can only used with the variable, but cann't use with expression and constant as ++6 or ++(x+y+z).

EXAMPLE

 let y=12;

z= ++y;

 y= y+1;

 z= y;

1. **Relational Operator**

 It is use to compared value of two expressions depending on their relation. Expression that contain relational operator is called relational expression. Here the value is assign according to true or false value.

a.(a>=b) || (b>20)

b.(b>a) && (e>b)

c. 0(b!=7)

 **5. Conditional Operator**

It sometimes called as ternary operator. Since it required three expressions as operand and it is represented as (? , :).

 SYNTAX exp1 ? exp2 :

exp3 Here exp1 is first evaluated. It is true then value return will be exp2 . If false then exp3. EXAMPLE

void main()

 {

 int a=10, b=2;

 int s= (a>b) ? a:b;

 printf(“value is:%d”);

 }

 Output: Value is:10

1. **Comma Operator** Comma operator is use to permit different expression to be appear in a situation where only one expression would be used. All the expression are separator by comma and are evaluated from left to right.

EXAMPLE

int i, j, k, l;

 for(i=1,j=2;i<=5;j<=10;i++;j++)

1. **Sizeof Operator**

 Size of operator is a Unary operator, which gives size of operand in terms of byte that occupied in the memory. An operand may be variable, constant or data type qualifier. Generally it is used make portable program(program that can be run on different machine) .

It determines the length of entities, arrays and structures when their size are not known to the programmer. It is also use to allocate size of memory dynamically during execution of the program.

EXAMPLE

main( )

{

 int sum;

float f;

printf( "%d%d" ,size of(f), size of (sum) );

 printf("%d%d", size of(235 L), size of(A)); }

1. **Bitwise Operator**

Bitwise operator permit programmer to access and manipulate of data at bit level. Various bitwise operator enlisted are

one's complement (~)

bitwise AND (&)

 bitwise OR (|)

bitwise XOR (^)

left shift (<>)

 These operator can operate on integer and character value but not on float and double.

In bitwise operator the function showbits( ) function is used to display the binary representation of any integer or character value.