



NUMBER SYSTEMS

CONTENTS

1. Introduction
2. Decimal number system
3. Binary number systems
4. Octal number system
5. Hexadecimal number system
6. Conversions from decimal to binary
7. Conversions from binary to decimal
8. Binary arithmetics

INTRODUCTION

- When we type some letters or words, the computer translates them in numbers as computers can understand only numbers.
- The value of each digit in a number can be determined using –
 - **The digit**
 - **The position of the digit in the number**
 - **The base of the number system**

BASE , WEIGHTING FACTOR

BASE or Radix

- In mathematical numeral systems, **the radix or base is the number of unique digits**, including zero, used to represent numbers in a positional numeral system.
- For example, for the decimal system (the most common system in use today) the radix is **ten**, because it uses the **ten** digits from 0 through 9.
 - **Base 2 (Binary): 0 and 1**
 - **Base 8 (Octal) : 0, 1, 2, 3, 4, 5, 6, 7**
 - **Base 10(Decimal) : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9**
 - **Base 16(Hexadecimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E , F**
- **Weighting factor = base raised to the power of position (value)**
- Weight of a respective position. In decimal form, weight of each digit increases by a factor of 10 as one moves to the left.
- **$754 = 7*100 + 5 *10 + 4$**

DECIMAL NUMBER SYSTEMS

Decimal Number System

- Decimal number system has base 10 as it uses 10 digits from 0 to 9.
- In decimal number system, the successive positions to the left of the decimal point represent units, tens, hundreds, thousands, and so on.
- Each position represents a specific power of the base (10). For example, the decimal number 1234 consists of the digit 4 in the units position, 3 in the tens position, 2 in the hundreds position, and 1 in the thousands position. Its value can be written as

1234

$$\mathbf{=1000 + 200 + 30 + 4}$$

$$\mathbf{= (1 \times 1000)+ (2 \times 100)+ (3 \times 10)+ (4 \times 1)}$$

$$\mathbf{=(1 \times 10^3)+ (2 \times 10^2)+ (3 \times 10^1)+ (4 \times 10^0)}$$

BINARY NUMBER SYSTEM

Characteristics of the binary number system are as follows –

- Uses two digits, 0 and 1
- Also called as base 2 number system
- First position in a binary number represents a **0** power of the base (2). Example 2^0
 - **101010 binary**
 - **$= (32*1) + (16*0) + (8*1) + (4*0) + (2*1) + 1*0$**
 - **$= 32 + 8 + 2 = 42$**

1	0	1	0	1	0
2^5	2^4	2^3	2^2	2^1	2^0
32	16	8	4	2	1

BINARY NUMBER SYSTEM

Example

- Binary Number: 10101_2
- Calculating Decimal Equivalent –

Step	Binary Number	Decimal Number
Step 1	10101_2	$((1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0))_{10}$
Step 2	10101_2	$(16 + 0 + 4 + 0 + 1)_{10}$
Step 3	10101_2	21_{10}

□ **Note** – 10101_2 is normally written as 10101.

OCTAL NUMBER SYSTEM

Characteristics of the octal number system are as follows

–

- Uses eight digits, 0,1,2,3,4,5,6,7
- Also called as base 8 number system
- First position in an octal number represents a **0** power of the base (8). Example 8^0

2	3	5
$2*8^2$	$3*8^1$	$5*8^0$
$2*64$	$3*8$	$5*1$
128	24	5

Octal 235

= decimal (128 + 24 + 5)

= decimal 157

OCTAL NUMBER SYSTEM

Example

- Octal Number: 12570_8
- Calculating Decimal Equivalent –

Step	Octal Number	Decimal Number
Step 1	12570_8	$((1 \times 8^4) + (2 \times 8^3) + (5 \times 8^2) + (7 \times 8^1) + (0 \times 8^0))_{10}$
Step 2	12570_8	$(4096 + 1024 + 320 + 56 + 0)_{10}$
Step 3	12570_8	5496_{10}

HEXADECIMAL NUMBER SYSTEM

Characteristics of the hexadecimal number system are as follows

- Uses 10 digits and 6 letters, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- Letters represent the numbers starting from 10. A = 10. B = 11, C = 12, D = 13, E = 14, F = 15
- Also called as base 16 number system
- First position in a hexadecimal number represents a **0** power of the base (16). Example, 16^0

HEXADECIMAL NUMBER SYSTEM

Hexadecimal Number: $19FDE_{16}$

Calculating Decimal Equivalent –

Step	Hexa Number	Decimal Number
Step 1	$19FDE_{16}$	$((1 \times 16^4) + (9 \times 16^3) + (F \times 16^2) + (D \times 16^1) + (E \times 16^0))_{10}$
Step 2	$19FDE_{16}$	$((1 \times 16^4) + (9 \times 16^3) + (15 \times 16^2) + (13 \times 16^1) + (14 \times 16^0))_{10}$
Step 3	$19FDE_{16}$	$(65536 + 36864 + 3840 + 208 + 14)_{10}$
Step 4	$19FDE_{16}$	106462_{10}

□ **Note** – 12570_8 is normally written as 12570..

BINARY ARITHMETIC

BINARY ADDITION

- There are four rules of binary addition.
- In fourth case, a binary addition is creating a sum of $(1 + 1 = 10)$ i.e. 0 is written in the given column and a carry of 1 over to the next column.

	Rule#1	Rule#2	Rule#3	Rule#4
A	0	0	1	1
+ B	0	1	0	1
A+B	0	1	1	0 (CARRY 1)

BINARY ADDITION

	Rule#1	Rule#2	Rule#3	Rule#4
A	0	0	1	1
B	0	1	0	1
A+B	0	1	1	0 (CARRY 1)

0011010 + 001100 = 00100110

$$\begin{array}{r} 11 \text{ carry} \\ 0011010 \\ +0001100 \\ \hline 0100110 \end{array} \begin{array}{l} = 26_{10} \\ = 12_{10} \\ = 38_{10} \end{array}$$

BINARY SUBTRACTION

- **Subtraction and Borrow**, these two words will be used very frequently for the binary subtraction. There are four rules of binary subtraction.

	Rule#1	Rule#2	Rule#3	Rule#4
A	0	1	1	0
-B	0	1	0	1
A - B	0	0	1	1 (BORROW 1)

BINARY SUBTRACTION

BORROW		1	1	1	
A	1	0	1	0	1
-B	0	1	1	1	0
A - B	0	0	1	1	1

TRANSLATOR, LINKER LOADER

□ Translator : three types

- **Assembler** (assembly language to machine language)
- **Compiler** (translates the source to object code at once)
- **Interpreter** (translates each line of source code, slower)

Depending on implementation , high level language uses either compiler or interpreter (Basic) or both (Java).

- For debugging, interpreted language is better than compiled language.

TRANSLATOR, LINKER LOADER

Linker

- High level language provides libraries so that certain common operation can be reused.
- In C, all subprograms and common variables need to be linked to form an useful execution unit.
- Linking makes the address of the programs known to each other or to the main program so that transfer of control from one point to another takes place during execution.

TRANSLATOR, LINKER LOADER

Loader

- Brings program from secondary memory to primary memory
- So that it can run. Initiates execution.
- Ex : Bootstrap loader is an absolute loader which is executed when computer is switched on or restarted.
- The main program is loaded into main memory and gets executed. When a function or subroutine is called, checks if it is already loaded is not then loads into the memory.

DEVELOPING PROGRAM IN C

1. Writing the program :

- use text editor
- Save file with .c extension

2. Compiling the program

3. Executing program –

1. Each statement is sequentially executed.
2. If program requests data from user, waits for input.
3. Or program can wait for an event like mouse click.
4. Results or output displays on the screen
5. Results can also be stored in a file.

BASIC STRUCTURE OF C PROGRAM

Basic Structure of C Programs	
Documentation Section	
Link Section	
Definition Section	
Global Declaration Section	
main() Function Section	
{	
Declaration Part	
Executable Part	
}	
Subprogram Section	
Function 1	
Function 2	
Function 3	
-	
-	
-	
Function n	

BASIC STRUCTURE OF C

BASIC STRUCTURE OF A 'C' PROGRAM:

Documentation section [Used for Comments]
Link section
Definition section
Global declaration section [Variable used in more than one function]
main() { Declaration part Executable part }
Subprogram section [User-defined Function] Function1 Function 2 : : Function n

Example:

→ `//Sample Prog Created by:Bsource`

→ `#include<stdio.h>`
`#include<conio.h>`

→ `void fun();`

→ `int a=10;`

→ `void main()`
`{`
`clrscr();`
`printf("a value inside main(): %d",a);`
`fun();`
`}`

→ `void fun()`
`{`
`printf("\na value inside fun(): %d",a);`
`}`

Example

```
/*-----Multiple Line -----  
-----Comments -----*/  
// C program to illustrate components  
#include <stdio.h> //preprocessor Directives<Header File> (It will expand the input and output function)  
#include <conio.h> //preprocessor Directives<Header File>  
#define LIMIT 5    //Macro definition  
Int a = 10;        //global variable  
  
void main() //Program Execution will start from here  
{    //Opening of the function  
    int b, c, sum; //Local variable  
    printf("Enter the value of b and c"); //Displays Output  
    scanf("b=%d,c=%d",&b,&c); //Takes Input from Keyboard  
    sum = b+c;  
    printf("Sum of two numbers = %d", sum);  
    printf("The value of LIMIT is %d", LIMIT);  
    printf("Value of a inside main() is %d\n", a);  
} //Closing of the function
```

COMPILING A C PROGRAM – FOUR STEPS

Step-1

Preprocessing

- It processes include files, conditional compiler instructions and macros
- Preprocessor directives are there in the source code which specifies instructions on how to modify the source code

Step-2

Compilation



- Converts the high level language to machine level language.
- Generates the assembly source code
- Checks syntaxes and semantics, generates error. Program has to be recompiled.


COMPILING A C PROGRAM – FOUR STEPS

Step-3

Assembly

Takes assembly code and generates **object code (binary format)**

Object code is written into another file on the system

filename.c  *filename.o*

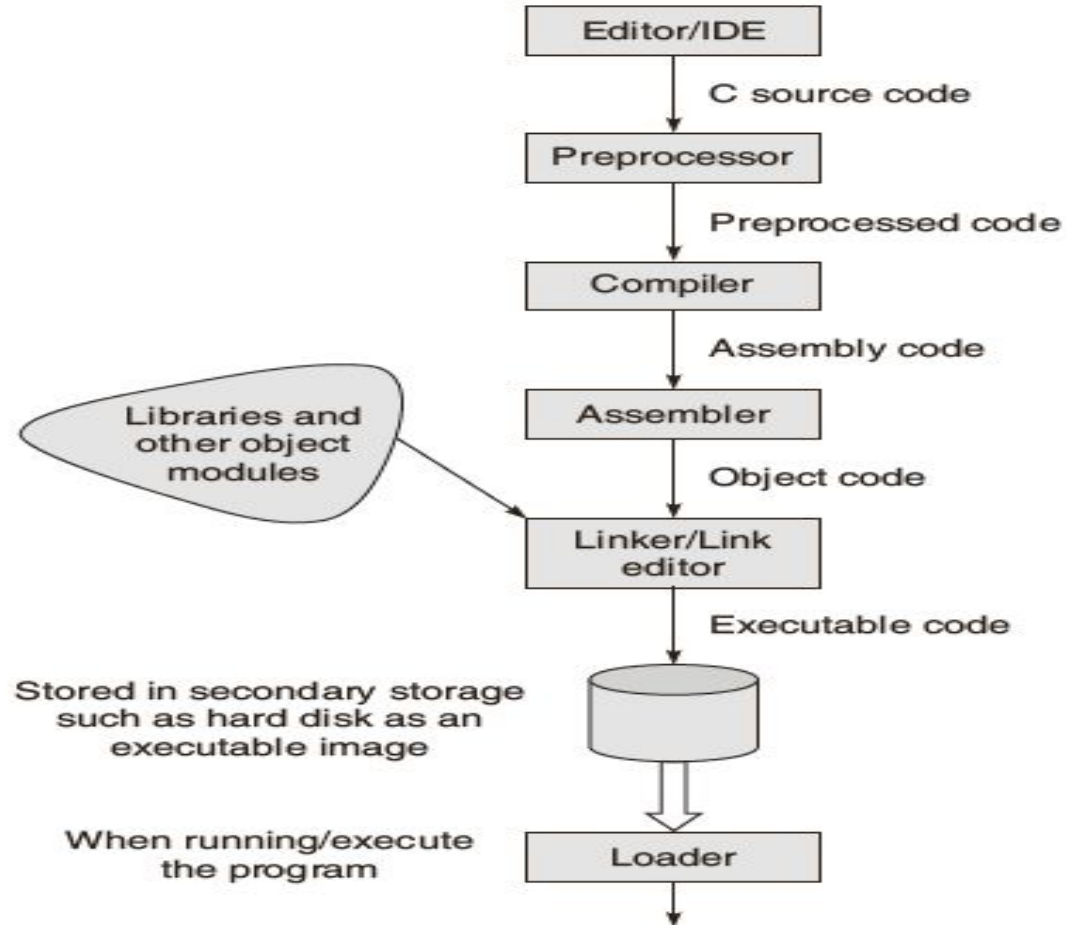
Step-4

Linking



□ Purpose of linking phase is to get the program in final form for execution.

COMPILING A C PROGRAM – EXAMPLE



ERROR

- **Compilation error**

- Given by compiler and prevent program from running
 - Missing parenthesis
 - Missing semicolons

- **Linking error**

- Given by linker or at run time
- If part of program is missing or non existent library component
 - Wrong prototype (Main instead of main)
 - Incorrect header file

- **Run time error**

- Given by operating system
 - Division by 0

Terminologies of 'C'

1. Keywords
2. Identifiers
3. Variables
4. Constants
5. Special Symbols
6. Operators
7. Character & String

1. Keywords

- Keywords are the reserved words whose meaning has already been explained to the C compiler.
- C has 32 keywords.
- These keywords combined with a formal syntax form a C programming language.
- Rules to be followed for all programs written in C:
 - ➡ All keywords are lower-cased.
 - ➡ C is case sensitive, do-while is different from DO WHILE.
 - ➡ Keywords cannot be used as a variable or function name.

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

2. Identifiers

- Identifiers refer to the name of variables, functions and arrays.
- These are user-defined names and consist of sequence of letters and digits, with a letter as a first character.
- Both uppercase and lowercase letters are permitted, although lowercase letters are commonly used .
- The underscore character is also permitted in identifiers. It is usually used as a link between two words in long identifiers.

Identifier Names

- ✧ Some correct identifier names are -

arena, s_count
marks40
class_one



- ✧ Some erroneous identifier names are -

1stsst
oh!god
start....end



- ✧ The number of characters in the variable that are recognized differs from compiler to compiler
- ✧ An identifier cannot be the same as a C keyword

3. Variables

- Variables are named locations in memory that are used to hold a value that may be modified by the program.
- Unlike constants that remain unchanged during the execution of a program.
- A variable may take different values at different times during execution.
- The syntax for declaring a variable is –
 DataType IdentifierName ;
- Example- int num;
 long int sum , a;

Variables : Attributes

- **All variables have three important attributes:**
 - A **data type** that is established when the variable is defined, e.g., integer, real, character. Once defined , the type of a C variable cannot be changed.
 - A **name** of the variable.
 - A **value** that can be changed by assigning a new value to the variable. The kind of values a variable can assume depends on its type. For example, an integer variable can only take integer values, e.g., 2, 100, -12.

Variable Type	Keyword	Bytes Required	Range	Format
Character (signed)	Char	1	-128 to +127	%c
Integer (signed)	Int	2	-32768 to +32767	%d
Float (signed)	Float	4	-3.4e38 to +3.4e38	%f
Double	Double	8	-1.7e308 to +1.7e308	%lf
Long integer (signed)	Long	4	2,147,483,648 to 2,147,438,647	%ld
Character (unsigned)	Unsigned char	1	0 to 255	%c
Integer (unsigned)	Unsigned int	2	0 to 65535	%u
Unsigned long integer	unsigned long	4	0 to 4,294,967,295	%lu
Long double	Long double	10	-1.7e932 to +1.7e932	%Lf

4. Constants

- Constants are the fixed values that do not change during the execution of a program.
- C supports several types of constants.
 - Numeric Constants
 - Integer constants
 - Real constants
 - Character Constants
 - Single character constant
 - String Constants

ASCII Value (Character encoding format for text data)

Characters	ASCII Value
A - Z	65 - 90
a - z	97 - 122
0 - 9	48 - 57
Special Symbol	0 - 47, 58 - 64, 91 - 96, 123 - 127

5. Special Symbols

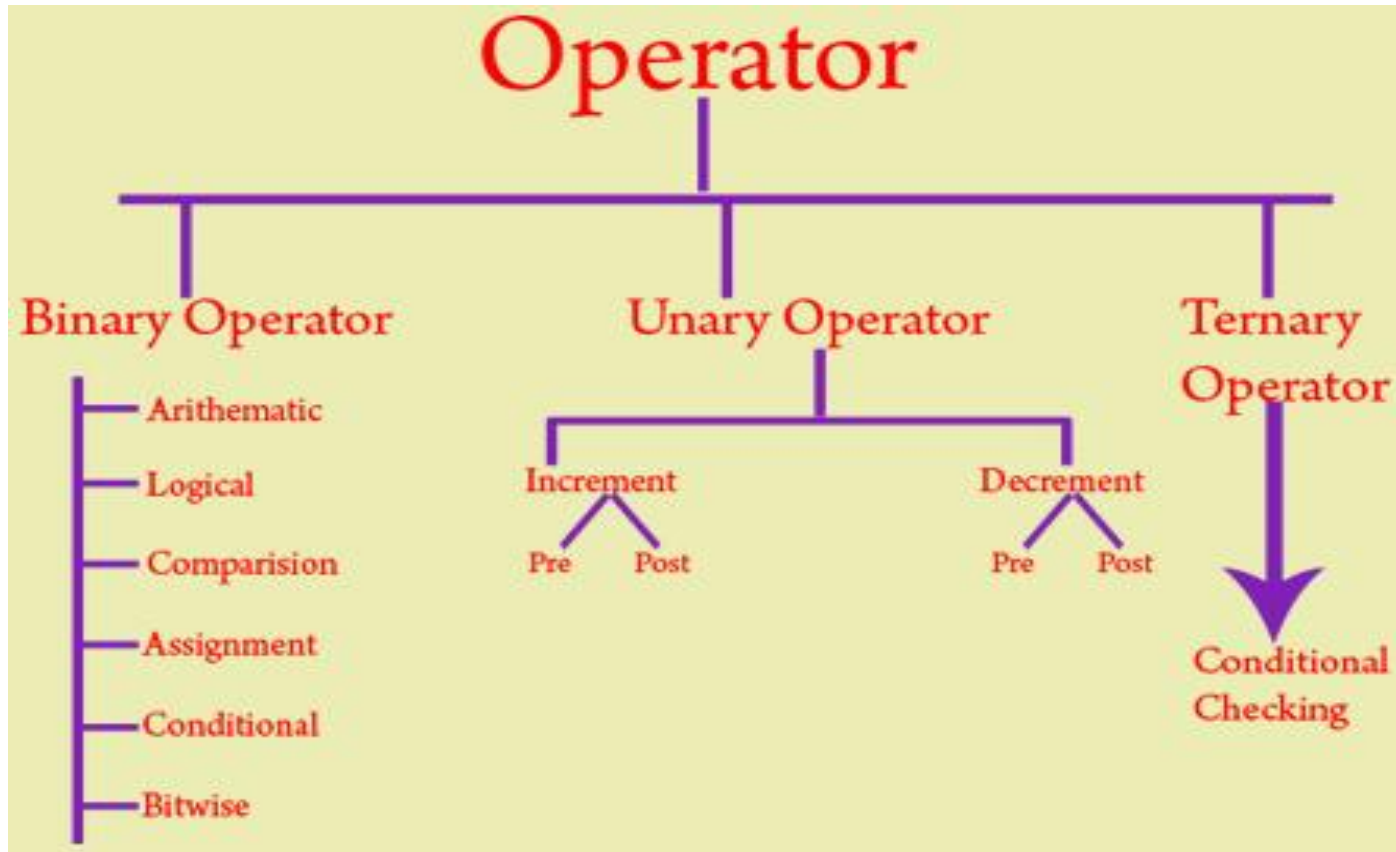
- ! , @ , # , \$, & , * , These all symbols that can be find on Keyboard, are called Special Symbols.
- Every symbol has its special meaning in different respect at different place that's why it is called Special Symbols.

6. Operators

- Operator is a symbol that operates on one or more operands and produces output.

Example - $c = a + b ;$

- In the above Example the symbols $+$ and $=$ are operators that operate on operands a , b , and c .



Unary Operator - Operates on one operand

Binary Operator - Operates on two operands

Ternary operator - Operates on three operands

Operators in C

	Operator	Type
Unary operator →	++, --	Unary operator
Binary operator {	+, -, *, /, %	Arithmetic operator
	<, <=, >, >=, ==, !=	Relational operator
	&&, , !	Logical operator
	&, , <<, >>, ~, ^	Bitwise operator
	=, +=, -=, *=, /=, %=	Assignment operator
Ternary operator →	?:	Ternary or conditional operator

7. Character & String

- The Characters that can be used to form words, numbers and expressions depend upon the computer on which the program is running.
- The characters in C are grouped into the following categories :
 - Letters
 - Digits
 - Special characters
 - White Spaces
- Remember that a character 'a' is not equivalent to the string "a".

Escape Sequence

- `\n` new line
- `\t` tab
- `\r` carriage return
- `\a` alert
- `\\` backslash
- `\"` double quote

Compilation and Execution of a C Programme

- **Step-1:** Create a file named as first.c in gedit editor and write the a program code in it, then save the file and quit from gedit window.
- **Step-2:** Compile the C Program file named as first.c
 - \$ gcc first.c
 - **\$ gcc first.c -o first**
- It compiles the file first.c, if it is error free, then go for execution to get output. Else open the file again in gedit to correct the errors, again compile it till it does not show any errors.
- **Step-3:** To get the output do the following
 - \$./a.out

Some Examples

- WAP to display “IIT” using the character ‘*’.

[illegible]

OUTPUT

```

*** **      *** **      *****
  *          *          *
  *          *          *
  *          *          *
  *          *          *
  *          *          *
*****      *****      *

```

Some Examples

- WAP to display the following message by using multiple printf statement.
A Good End
Can Only Be Achieved
Only By Good Means.

```
#include <stdio.h>
void main()
{
printf("\n A Good End ");
printf("\n Can Only Be Achieved ");
printf("\n Only By Good Means.");
}
```

Some Examples

- WAP to display the following message by using a single printf statement.
A Good End
Can Only Be Achieved
Only By Good Means.

```
#include <stdio.h>
void main()
{
printf("\n A Good End\nCan Only Be Achieved\nOnly By Good Means.");
}
```

WAP to input and display an integer number.

Code

```
// Printing an integer
#include<stdio.h>
void main()
{
    int a;
    printf("Enter an integer number :");
    scanf("%d", &a);
    printf("The entered number is %d", a);
}
```

Output

```
Enter an integer number
35
```


WAP to perform the addition of two integers & display the result.

Code

```
#include<stdio.h>
void main()
{
int a, b, c;
printf("\n Enter two numbers to add :");
scanf("%d%d",&a,&b);
c = a + b;
printf("\n The addition of %d and %d is %d", a,b,c);
}
```

Assignment

- WAP to display the following message by using multiple printf statement.

If The End Is Good,
Then It Is Good,
Whatever Be The Means.

- WAP to display the above message using single printf statement.
- WAP to print your BIO-DATA (Name, Regd.no", Branch, JEE Rank, Gender, Phone no., Address etc.) using printf statement.