



INFUSION NOTES
WHEN ONLY THE BEST WILL DO

RAJASTHAN

COMPUTER INSTRUCTOR

(RAJASTHAN STAFF SELECTION BOARD (RSSB))

(ENGLISH MEDIUM)



PART – 4

COMPUTER STUDY (PART - 2)

Preface

Dear Readers, The presented notes "**Rajasthan Computer Instructor (English Medium)**" have been prepared by a dedicated team of teachers and colleagues, each proficient in their respective subjects. These notes aim to provide comprehensive support to readers appearing for the "**Rajasthan Computer Instructor Recruitment Examination**" conducted by the Rajasthan Staff Selection Board.

Despite careful efforts, there may still be some errors or shortcomings in the notes. Therefore, valuable suggestions from you, the respected readers, are warmly welcomed.

Publisher:-

INFUSION NOTES

Jaipur, 302029 (Rajasthan)

Mob: 9887809083

Email: contact@infusionnotes.com

Website: <http://www.infusionnotes.com>

WhatsApp link - <https://wa.link/yoh401>

Online Order link - <https://shorturl.at/mhX4l>

PRICE: - ₹

EDITION: - LATEST

क्र. सं.	अध्याय	पृष्ठ सं.
1.	Programming Fundamental & Languages <ul style="list-style-type: none"> • <i>Introduction to C</i> • <i>C++</i> • <i>Java</i> • <i>OOPs (Object-Oriented Programming)</i> • <i>NET</i> • <i>Python</i> • <i>AI (Artificial Intelligence)</i> • <i>Machine Learning</i> • <i>Block chain</i> • <i>Principles and Programming Techniques</i> • <i>Integrated Development Environment (IDE)</i> 	1-163

Chapter - 1

Programming fundamentals

1. What is C

- C is a Structured Oriented Middle Level Programming Language, which is basically used to develop various Application Softwares and Systems Softwares of Computer.
- C Language was developed by Dennis Ritchie. After that in 1978 Denis Ricchie and Brain Kernighan published the first version of C language, **The C Programming Language**.
- C language is also called Mother Language, why whatever language was created after C language for example (Java, PHP, C#, or C++) the concept of C language has been used in all these languages.

History of C Language

C Language was published by Brian Kernighan and Dennis Ritchie in 1978. So far many versions of C language have come, which are -

- **K & R** - This is the original language of C language. This version was introduced in 1978. And in this version a function like Standard I/O library was available.
- **ANSI C and ISO C** - This version is called American National Standards Institute (ANSI) and it was published by the International Organization for Standardization (ISO) in 1989/1990.
- **C99** - This version was published in 1999. And in this version some new features were added like - inline functions, several new data types, long int etc.
- **C11** - This version was published in 2011. And in this version also some new features were added like - library, including type generic macros, anonymous structures etc.
- **C18** - This version was published in June 2018. **History of C Language (Tabular Form)**

Language	Year	Developed By
Algo	1960	International Group
BCPL	1966	Martin Richard
Traditional C	1969	Ken Thompson

K & R C	1972	Dennis Ritchie
ANSI C	1978	Kernighan & Dennis Ritchie
ANSI/ISO C	1989	ANSI Committee
C90	1990	ISO Committee
C99	1999	ISO Committee
C11	2011	Standardization Committee
C18	2018	Standardization Committee

Features of C language

1. Portability

Programs written in C language are quite portable, meaning programs written in C language can be easily run on different machines or PCs without any changes. The compiler and preprocessor make it possible to run on different PCs.

2. Powerful Programming Language

C programming language is very fast and efficient programming language because it uses Data Types, Function and Control Statements.

3. Simple Programming Language

C language has commands like English, which makes it very easy for the programmer to write and understand the code.

4. Structured Oriented Language

C Language is Structured Oriented Programming Language. Structured Oriented Programming Language reduces the complexity of the code, which leads to a lot of clarity in the programs.

5. Compiler Based

C Language is Compiler Based Programming Language, it means C language programs cannot be executed or run without compiling.

6. Syntax Based Language

C language is Syntax Based Programming Language. Syntax Based Programming Language is the language which follows the rules and regulation very strictly like C Language, C++ Language, Java Language etc. If a programming language does not strictly follow the Rules and Regulation, then such



programming language is loosely called Syntax based programming language like HTML.

7. Efficient Use of Pointers

Pointer is a variable that points to the address of another variable.

8. Middle Level Language

C language is a Middle Level Programming Language, meaning low level programming can be done in it and high level programming can also be done, due to which both Application Software and System Software can be made very easily with the help of C Language.

9. Case Sensitive

C language is a case sensitive programming language. We understand the meaning of Case Sensitive with an example, if we write printf like this in one place in C language and write PRINTF like this in another place, then both mean different in C language.

This is a special feature of C language.

If in a programming language the words written in Lower Case and Upper Case have different meanings, then such programming language is called Case Sensitive Programming Language like - C, C++, JAVA, .NET Programming Language is Case Sensitive Programming Language and such. Programming language in which there is no difference between words written in lower case and upper case, such programming language is called case insensitive programming language like - HTML, SQL language

10. Modularity

In C language, we can divide our code into blocks, which makes it easier to read and write code, this is a technique of designing a software in which a whole program is divided into many functions or blocks.

11. Dynamism

There is also the concept of DMA Dynamic Memory Allocation in C language which helps in using and managing the memory very efficiently.

12. Rich Library

C language provides us with pre-built libraries that speed up the creation of programs.

13. Statically Typed Programming Language

C language is a statically typed programming language. Which means the data type of the variable is checked at the time of compilation of the program in a language, but not at run time. This means whenever the programmer will create a program and use any variable in it, every time the programmer will have to declare the data type of that variable.

14. General Purpose Language:

From system programming to photo editing software, the C programming language is used to create various applications. Some common applications where C programming language is used are:

Operating System -: Windows, Linux, Android, OXS, iOS

Database -: PostgreSQL, Oracle, MySQL, MS SQL Server etc.

15. Recursion

In C language, we can call a function within a function. This feature allows the code to be reused for each function. We'll read more about recursion in the section with more functions.

16. Fast Speed

C programming language is much faster than other programming languages because it is a compiler-based language, so it is faster than other programming languages like java, python, which are interpreter based.

17. Extendable

C language can be changed over time. C language has the ability to easily adopt new improvements.

Application of C Language

- I. C language is used to create computer application software such as database, spread sheet etc.
- II. C language is used to write Embedded Software.
- III. C language is used to create system software such as Operating System.
- IV. To create applications related to graphics such as computer and mobile games.
- V. Unix kernel is completely developed in C language.
- VI. C language is used to create network devices and device drivers.

VII. C language is used to make compiler. Compiler converts high level code into low level code or machine code.

Installation of C on your system

There are many compilers available for C language. You can download any one. Here, we are going to install **Turbo C++**. We can use it for both C and C++. To install the Turbo C software, you need to follow following steps.

1. Download Turbo C++ online.
2. Create turboc directory inside C drive and extract the tc zip inside c:\turboc
3. Double click on install.exe file and installation will start.
4. Click on the tc (shortcut symbol) application file located inside c:\TC\BIN to write the c program

First C Program

Before starting the program of C language, you need to learn how to write, compile and run the program in c console.

How to create first c program:-

Before making the first program of c, you have to take care of some important things. Which is the following:-

- I. It is necessary to include the header file.
- II. Header file has to start with # tag.
- III. It is very important to have a main function in your program.

IV. Capital and small letters have to be taken care of while writing statements in c.

V. Almost all statements in c must be terminated with a semicolon.

Open the C console and write the following code:

```
#include <stdio.h>
int main(){
printf("Wellcome to Infusion notes");
return 0;
}
```

Output :-

Wellcome to infusion notes

- I. #include <stdio.h> includes the standard input output library functions.
- II. The printf() function is defined in stdio.h .
- III. int main() The main() function is the entry point of every program in c language.

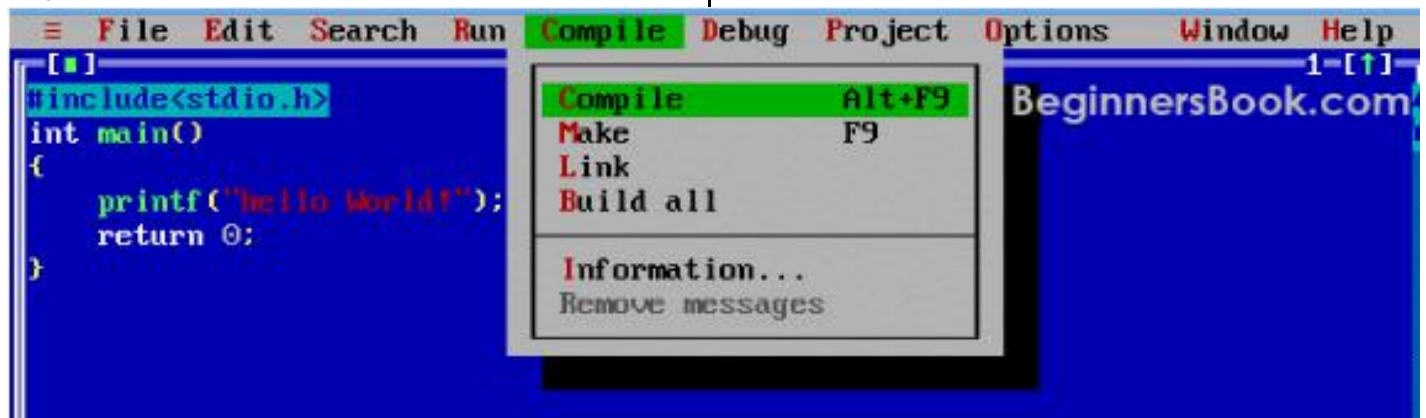
How to compile and run the c program

There are 2 ways to compile and run a C program, by menu and by shortcut.

c program output by Menu

Now click on compile menu and then compile sub menu to compile C program.

Then click on run menu and then run sub menu to run C program.



From shortcut

Or, press ctrl + f9 keys and run the program directly.

- You will see the following output on the user screen.
- You can view the user screen at any time by pressing Alt + F5 keys.
- Now press Esc to go back to Turbo c++ console



C Language – Flow Chart

- Introduction to flowchart
- Symbols of flowchart
- Examples of flowchart




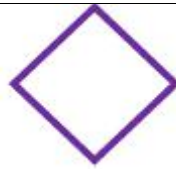

Introduction of Flowchart

Flow chart is the graphical representation of our program. Whenever someone writes a program in C language, he writes without flow chart. If we want to understand our program, then the program will

not understand. That is why we can through our flow chart represent the program graphically. So that anyone can easily understand the program's logic.

Symbols of Flowchart

We have to use some symbols to represent graphically the program. This is the symbol we provide. These are called flow chart symbol.

Name	Symbols	Explanation
Start/End		This symbol is called oval symbol. And also called terminal symbol. It is used to start and end flow chart
Processing		This symbol is called processing symbol. Which is used for processing.
Input/Output		This symbol is called parallelogram symbol. It is used to denote symbol input and output.
Decision		This is called diamond symbol. Which uses to denote decision making statements.
Arrow (Flow)		This is called arrow symbol. Which is used arrow symbol to show the flow of the program

Every symbol in flow chart has a meaning .Which the flyer can easily understand. The below flow chart symbols are being told

Character Set

1. Characters are letters that are used to make words in any language. Sentences are formed from words, which reveal the semantic meaning of that word.

II. Computer language is a language using which we can tell the computer about our needs.

III. Therefore this language also has grammar like other languages.

IV. The very first step in learning any language is to know its character set which is called character set in English.

V. Like - When we learn any language, we are first taught a, b, c, d of that language or while learning English a, b. In the same way, before learning C language, you need to know about its characters. The character set in the language is divided into four parts, which are as follows.

- Letters
- Digit
- Special Character
- White Space

Letters: The English language alphabet A to Z and 3 to 2 are its letters which are used to write programs. "C" is case sensitive language. So "C" treats both a and A as separate letters. Identifiers, Keyword String etc. are formed by adding these letters.

Digit: Digit is also used in C language, which are used for identifiers numeric value. There are only ten (0,1, 2,3,4,5,6,7,8, 9) in this number, but you can also use it by grouping them. 123,345,764 etc.

Special Character There are special types of characters in C language, which are used only in special places, let us know which special characters are used in the language.

~	Tilde
!	Exclamation mark
#	Number sign
\$	Dollar sign
%	Percent sign
:	Colon
"	Quotation mark
;	Semicolon
<	Opening angle bracket

>	Closing angle bracket
?	Question mark
,	Comma
.	Period
/	Slash
+	Plus sign
	Vertical bar
\	Backslash
'	Apostrophe
-	Minus sign
=	Equal to sign
{	Left brace
}	Right brace
[Left bracket
]	Right bracket
^	Caret
&	Ampersand
*	Asterisk
(Left parenthesis
)	Right parenthesis
_	Underscore

Variable

- I. In C language, when we want to use some data value in our program, we can store it in a memory space and name the memory space so that it becomes easier to access it.
- II. The naming of an address is known as variable. Variable is the name of memory location.
- III. Unlike constant, variables are changeable, we can change value of a variable during execution of a program.
- IV. A programmer can choose a meaningful variable name. Example : average, height, age, total etc.



Datatype of Variable

A variable in C language must be given a type, which defines what type of data the variable will hold.

- I. char: Can hold/store a character in it.
- II. int: Used to hold an integer.
- III. float: Used to hold a float value.
- IV. double: Used to hold a double value.
- V. Void

Rules to name a Variable

- I. Variable name must not start with a digit.
- II. Variable name can consist of alphabets, digits and special symbols like underscore _.
- III. Blank or spaces are not allowed in variable name.
- IV. Keywords are not allowed as variable name.
- V. Upper and lower case names are treated as different, as C is case-sensitive, so it is suggested to keep the variable names in lower case.

Declaring, Defining and initializing a variable

Declaration of variables must be done before they are used in the program. Declaration does the following things.

- I. It tells the compiler what the variable name is.
- II. It specifies what type of data the variable will hold.
- III. Until the variable is defined the compiler doesn't have to worry about allocating memory space to the variable.
- IV. Declaration is more like informing the compiler that there exist a variable with following datatype which is used in the program.
- V. A variable is declared using the extern keyword, outside the main() function.

Expressions

- I. An expression is a combination of variables, constants, operators and function call.
- II. It can be arithmetic, logical and relational for example:-
`int z= x+y // arithmetic expression`
`a>b //relational`
`a==b // logical`
`func(a, b) // function call`

Keywords in C

- I. Like in our language Hindi the words 'laugh, cry, run' already have a meaning and therefore we do not use these words for any other work, just like keywords are there in programming.
- II. Keywords are reserved words which already have specific meaning in C programming.
- III. Now as you have understood that the meaning and use of C keywords is already fixed, so you cannot use the keywords for any other work.
- IV. There are 32 keywords in C programming language and C keywords are always written in lowercase, a list of all 32 keywords is given below.

1.	auto
2.	do
3.	else
4.	if
5.	long
6.	static
7.	switch
8.	while
9.	break
10.	default
11.	enum
12.	case
13.	continue
14.	double
15.	for
16.	int
17.	signed
18.	struct
19.	void
20.	unsigned
21.	const
22.	extern
23.	float
24.	char
25.	short
26.	union
27.	return



28.	goto
29.	register
30.	sizeof
31.	typedef
32.	volatile

What is the Meaning of Identifiers in C:

- I. When we develop the program, then we have to input different types of data into the computer's memory and do different types of processing on it.
- II. No matter what kind of process we want to do with the data in the computer, we need to first store every data in the computer's memory.
- III. Without storing any data in the memory of the computer, we cannot do any kind of process with that data.
- IV. Every location of memory in the computer has a unique address.
- V. When we input any data in the computer to process, then that data gets stored by going to some location of the memory.
- VI. But we can never know in a normal way that the data we input is stored on which memory location of the computer, nor can we ever decide on which memory location our data will be stored.
- VII. Because the work of allocating the memory to the data is done by our operating system itself according to its convenience.

To name an identifier, we have to follow the following rules, which are called Identifier Naming Convention:

- I. Any Upper Case and Lower Case character can be used in the name of any identifier.
- II. Underscore can also be used in the name of any identifier.
- III. If we want to use digits in the name of any identifier, then it is necessary to have at least one character or underscore before using the digits.
- IV. Apart from this, any type of special symbol such as Period, Comma, and Blank Space etc. cannot be used in the name of the identifier. Also, we cannot use the name of any Reserve Word or any Built-In Function in the name of the identifier.
- V. No name can start with a digit.

VI. %, @, - characters symbol is not used in identifiers.

VII. C language is a case sensitive language. Therefore num and Num will be known as two different identifiers.

VIII. We cannot use operators in identifiers.

IX. Identifiers are not started with Digits. Identifiers can be started with a character or an underscore.

Examples identifiers -

_ram // right.

Ram-123 // wrong.

ram_45 // right.

4ram // wrong.

Data types

Whenever we coding in C language. So we have to create variables. And the types of these variables are called data types.

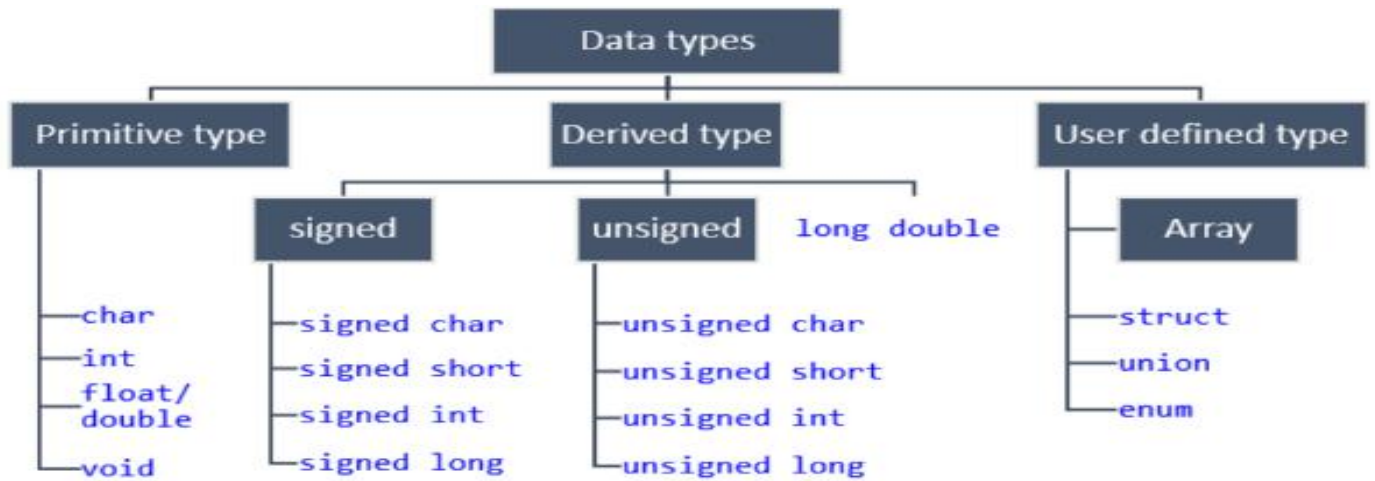
We cannot create any program without declaring the data type, whether it is C language or C++. If we want to make any kind of program in c language. So for that we have to create variables.

Datatype Modifiers

- I. As the name implies, datatype modifiers are used with the built-in data types to modify the length of data that a particular data type can hold.
- II. Modifiers are keywords in c which changes the meaning of basic data type in c.
- III. It specifies the amount of memory space to be allocated for a variable.
- IV. Modifiers are prefixed with basic data types to modify the memory allocated for a variable.
- V. There are five data type modifiers in C Programming Language:

- long
- short
- signed
- unsigned
- long long

There are mainly three types of data types found in C language. First is **int (integer)** and second is **float** and third is **char**.



Types of Data Types in C Language

There are mainly three types of data types in C language -:

1. Pre-defined (Primitive data)data types
2. Derived Data Types
3. User-defined data type

I. Primitive data types

Data Types	Memory Size	Range
char	1 byte	-128 to 127
signed char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255
short	2 byte	-32,768 to 32,767
signed short	2 byte	-32,768 to 32,767
unsigned short	2 byte	0 to 65,535
int	2 byte	-32,768 to 32,767
signed int	2 byte	-32,768 to 32,767
unsigned int	2 byte	0 to 65,535
short int	2 byte	-32,768 to 32,767
signed short int	2 byte	-32,768 to 32,767
unsigned short int	2 byte	0 to 65,535
long int	4 byte	-2,147,483,648 to 2,147,483,647
signed long int	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long int	4 byte	0 to 4,294,967,295
float	4 byte	1.2E-38 to 3.4E+38
double	8 byte	2.3E-308 to 1.7E+308
long double	10 byte	3.4E-4932 to 1.1E+4932

Integer Type

- I. The "int" keyword is used to create a variable of type Integer.

These data types are built-in or predefined data types and can be used directly by the user to declare variables.

Examples of Pre-defined Data Types

- Integer Type - int, long int, short int, unsigned int
- Character Type - char
- Floating Point - Float , double

- II. We store numeric value in the variable of Integer type.
- III. Variable made of "int" data type can store 2 byte, 4 byte and 8 byte data. But this thing depends on the compiler.



```
printf("%d x %d = %d\n",num,a,num*a);
a++;
if(a<=5)
goto table;
}
```

Output:

Enter the number whose table you want to print=10

10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40

When should we use goto?

The only condition in which it is better to use goto is when we need to break multiple loops using the same statement at the same time. Consider the following example.

```
include <stdio.h>
int main()
{
int a, b, k;
for(a=0;a<10;a++)
{
for(b=0;b<5;b++)
{
for(k=0;k<3;k++)
{
printf("%d %d %d\n",a,b,k);
if(b == 3)
{
goto out;
}
}
}
}
out:
printf("out of the loop");
}
```

Output -

0 0 0
0 0 1
0 0 2
0 1 0
0 1 1
0 1 2
0 2 0
0 2 1
0 2 2
0 3 0
out of the loop

Array (array)

Array (array) is a group of data items of the same type. These data items are related to each other. These groups are recognized by the same name. All the data items of the array are stored in the same place in the memory which is called Contiguous Memory Allocation.

To declare an array, after the name, write a value in the square bracket, which is called the size of the array.

```
int n [10];
```

"1" is an Array Variable in which 10 different numbers can be stored. To access these numbers, a value is required which is called Index Value which tells the position of that data item in this array. The index value in the array starts with 0 (zero).

Some important things of Array Declaration which will be used in further programming

- (1) Array is a group of same type of data items.
- (2) The position of the first data item of the array starts from 0 (Zero).
- (3) Before using Array it is necessary to decide its type and size.
- (4) All the data items of the array are stored in the same place in the memory, which are used with the help of an index variable.
- (5) Array is also called sub-script variable.

Types of Arrays in c language - Array has the following types-

- (i) One Dimensional Array

(ii) Two Dimensional Array

(iii) Multi Dimensional Array

(I) One-Dimensional Array- One dimensional array is also called single dimensional array. It is a linear list in which similar type and related data items are stored. All the data items in memory are stored one after the other in contiguous memory allocations.

An index variable is used to access or read all the data items of a one-dimensional array.

Declaration:

Rules for Declaring One Dimensional Array

- I. An array variable must be declared before being used in a program.
- II. The declaration must have a data type (int, float, char, double, etc.), variable name, and subscript.
- III. The subscript represents the size of the array. If we declared the size is as 10, programmers can store 10 elements.
- IV. An array index always starts from 0. Example- if an array variable is declared as `t[10]`, then it ranges from 0 to 9.
- V. Each array element stored in a separate memory location.

Initialization of One-Dimensional Array in C

An array can be initialized at either following states:

1. At compiling time (static initialization)
2. Dynamic Initialization

Compiling time initialization:

The compile-time initialization means the array of the elements are initialized at the time the program is written or array declaration.

Syntax : `datatype name [size];`

Run time initialization:

In Run time initialization the array can be initialized at runtime. That means array elements are initialized after the compilation of the program.

Program:

For Example: `int a[5]={1,2,3,4,5};`

During compilation, 5 contiguous memory locations are reserved by the compiler for the variable `a` and all these locations are initialized as shown in below figure.

a[0]	a[1]	a[2]	a[3]	a[4]
10	15	1	3	20
1000	1002	1004	1006	1008

ii) Two-dimensional array- Two-dimensional array is also called Matrix Presentation of Data Items. Two-dimensional array is used to arrange data items in rows and columns in tabular form. In this, two index values are used to display a particular data item for the first row and for the second column.

Declaration:

`datatype name [no. of Rows] [no. of Columns];`

example: Storing elements in a matrix

```
#include <stdio.h>
void main ()
{
    int arr[3][3],i,j;
    for (i=0;i<3;i++)
    {
        for (j=0;j<3;j++)
        {
            printf("Enter a[%d][%d]: ",i,j);
            scanf("%d",&arr[i][j]);
        }
    }
    printf("\n printing the elements ....\n");
    for(i=0;i<3;i++)
    {
        printf("\n");
        for (j=0;j<3;j++)
        {
            printf("%d\t",arr[i][j]);
        }
    }
}
```



```
}
}
```

Output

Enter a[0][0]: 45

Enter a[0][1]: 12

Enter a[0][2]: 74

Enter a[1][0]: 12

Enter a[1][1]: 10

Enter a[1][2]: 85

Enter a[2][0]: 96

Enter a[2][1]: 83

Enter a[2][2]: 41

printing the elements

45	12	74
12	10	85
96	83	41

(ii) Multi-dimensional array- Two-dimensional array is considered as a type of multi-dimensional array. However, it is used to create a 3-Dimensional, 4-Dimensional array.

e.g., int M[2][3][2];

Addition of 2 matrix

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

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

```
void main()
```

```
{
```

```
int
```

```
a[2][3], b[2][3], c[2][3], i, j;
```

```
clrscr();
```

```
printf("\nENTER VALUES
```

```
FOR MATRIX A:\n");
```

```
for(i=0; i<2; i++)
```

```
for(j=0; j<3; j++)
```

```
scanf("%d", &a[i][j]);
```

```
printf("\nENTER VALUES
```

```
FOR MATRIX B:\n");
```

```
for(i=0; i<2; i++)
```

```
for(j=0; j<3; j++)
```

```
scanf("%d", &b[i][j]);
```

```
for(i=0; i<2; i++)
```

```
for(j=0; j<3; j++)
```

```
c[i][j] = a[i][j] + b[i][j];
```

```
printf("\nTHE VALUES OF  
MATRIX C ARE:\n");
```

```
for(i=0; i<2; i++)
```

```
{
```

```
for(j=0; j<3; j++)
```

```
printf("%5d", c[i][j]);
```

```
printf("\n");
```

```
}
```

```
getch();
```

```
}
```

Use of array in c programming -

- I. Array has many uses in 'C' programming. like
- II. Array is used to store multiple data-items of the same type.
- III. Array is used to manage the list of Number or String.
- IV. Array is used to perform various types of matrix operations.
- V. Like- Addition & Multiplication of two matrixes, etc.
- VI. Array is also used in Recursive function.
- VII. Array is also used for CPU Scheduling in the computer.

Points to Remember About Array in C:

1. An array is a derived data type in C that is constructed from the fundamental data type of C language.
2. An array is a collection of similar types of values in a single variable.
3. In implementation when we required the 'n' number of the variable of a similar data type then we need to go for the array.
4. When we are working with an array always memory will be created in the contiguous memory location, so randomly we can access the data.

5. In an array, all elements will share the same name with a unique identification value called index.
6. Always array index will start from 0 and ends with size - 1.
7. When we are working with an array compile-time memory management will occur i.e. static memory allocation.
8. Always size of the array must be an unsigned integer value which should be greater than zero.

Function

Functions in c:-

- I. A function is a block of statements that performs a specific task. Let's say you are building an application in C language and in one of your programs, you need to do the same task more than once. In such a case you have two options -
 - Use the same set of statements each time you want to perform the task
 - Create a function to perform that task and just call it every time to perform that task.
- II. Function is a piece of code, in other words, it works like a sub-program in a program.
- III. While making a program, sometimes we need that we have to use the result obtained from the execution of some code again and again in the program, in which case the code is not written again and again but as a function `main()` is defined outside the function and used in one place. And to use the result obtained from the function in the program, that function is called.
- IV. The advantage of functions is that it saves both our time and space.

There are two types of functions in the programming language "c".

1:- Built-in-functions

2:- user defined functions

1:- Built-in-function:-

Built-in functions are those functions whose prototypes are preserved in the header file of the programming language "C". These functions are called by simply writing their name in the program and they get executed in the program. Examples of this:- `scanf()`, `printf()`, `strcat()`; e.t.c.

These functions are also called library functions of the "C" programming language. All these functions are related to a specific "C" library file. These special library files are called header files and their extension name is `.h`. Therefore, all those files located in the library of "C" whose extended name is `.h`, are called header files.

There are many header files in "C", some of them are as follows:-

- `stdio.h`
- `Math.h`
- `string.h`
- `conio.h`
- `time.h`
- `ctype.h`

Syntax of a function

```
return_type function_name (argument list)
{
```

Set of statements - Block of code

```
}
```

return_type: The return type can be of any data type such as `int`, `double`, `char`, `void`, `short` etc. Don't worry you will be able to understand these terms better after going through the examples given below.

function_name: It can be anything, however it is advisable to have a meaningful name for the functions, so that only the purpose of the function is easy to understand.

Argument list: The argument list contains the variable names along with their data types. These arguments are the types of input to the function. For example - a function that is used to add two integer variables will have two integer arguments.

Block of Code: Set of C statements that will be executed whenever the function is called.

User Defined Functions

- I. C allows you to define functions according to your need. These functions are known as user-defined functions.
- II. Suppose, you need to create a circle and color it depending upon the radius and color. You can create two functions to solve this problem:

Commonly Used std::string Functions:

Function	Description	Example
length() or size()	Returns the number of characters in the string.	<pre>std::string str = "Hello"; std::cout << str.length(); // 5</pre>
empty()	Checks if the string is empty. Returns true if the string is empty, otherwise returns false.	<pre>std::string str = ""; std::cout << str.empty(); // true</pre>
append()	Appends a string or character to the end of the string.	<pre>std::string str = "Hello"; str.append("World"); std::cout << str; // Hello World</pre>
push_back()	Adds a single character to the end of the string.	<pre>std::string str = "Hello"; str.push_back('!'); std::cout << str; // Hello!</pre>
substr()	Returns a substring starting from a given position with an optional length.	<pre>std::string str = "Hello World"; std::cout << str.substr(0, 5); // Hello</pre>
find()	Finds the first occurrence of a substring or character. Returns -1 if not found.	<pre>std::string str = "Hello World"; std::cout << str.find("World"); // 6</pre>
erase()	Removes a portion of the string, starting at a specified position for a specified length.	<pre>std::string str = "Hello World"; str.erase(5, 6); std::cout << str; // Hello</pre>
insert()	Inserts a string or character at a specific position.	<pre>std::string str = "Hello"; str.insert(5, "World"); std::cout << str; // Hello World</pre>
replace()	Replaces a portion of the string with another string or character.	<pre>std::string str = "Hello World"; str.replace(6, 5, "C++"); std::cout << str; // Hello C++</pre>
at()	Returns the character at the specified position. Throws an exception if the index is out of range.	<pre>std::string str = "Hello"; std::cout << str.at(1); // e</pre>
front()	Returns the first character of the string.	<pre>std::string str = "Hello"; std::cout << str.front(); // H</pre>
back()	Returns the last character of the string.	<pre>std::string str = "Hello"; std::cout << str.back(); // o</pre>
clear()	Removes all characters from the string, making it empty.	<pre>std::string str = "Hello"; str.clear(); std::cout << str.empty(); // true</pre>

Function	Description	Example
c_str()	Returns a C-style string (const char array) representation of the string.	std::string str = "Hello"; const char* cstr = str.c_str();
compare()	Compares the string with another string. Returns 0 if the strings are equal, a positive number if the first is greater, and a negative number if the second is greater.	std::string str1 = "Hello"; std::string str2 = "World"; std::cout << str1.compare(str2); // Negative
swap()	Swaps the contents of the current string with another string.	std::string str1 = "Hello"; std::string str2 = "World"; str1.swap(str2);
resize()	Resizes the string to the specified length. If the new size is larger, it will be padded with the specified character.	std::string str = "Hello"; str.resize(8, '!'); std::cout << str; // Hello!!!
to_string()	Converts numeric values to a string (available in C++11 and above).	int num = 123; std::string str = std::to_string(num); std::cout << str; // "123"

Preprocessor -- This is a program, which is executed even before the source code is compiled.

Normal Example for Preprocessor

```
#include <iostream.h>

int main ()
{
    cout<<"Hello";

return 0;
}
```

Here Preprocessors are divided into different departments.

- I. Directive Types Preprocessors
- II. Macro #define
- III. File Inclusion #include
- IV. Conditional Compilation #if, #else, #ifdef, #endif, #ifndef, #elif
- V. Other Directives #pragma, #undef
- VI. Macro - #define

Macros are identifiers, in the program where there is an identifier, then it is replaced and the value given there comes.

Predefined Macros

Macro	Description
DATE	It returns the string of the current date.
FILE	This will return the current file name along with its path.
LINE	This macro will return the number of the line on which it is located.
STDC	If the compiler follows ANSI Standard C++, it will return '1'.
TIME	It returns the string of the current time

C++ - File Inclusion #include

- I. #include to include header files in the program makes use of |
- II. There is a reference to #include in every program.

III. Preprocessors are included in two types.

1. #include
2. #include "file_name"

IV. Some header files are predefined and some header files are also user-defined.

V. Predefined header files : `iostream.h`, `conio.h`, `string.h`, `math.h`

VI. User-defined header file : `myheader.h`

VII. Header files contain functions, in which program to use these functions, then header files have to be included.

VIII. `string.h` header files : `strlen()`, `strcat()`

IX. `myheader.h` header file : `myfunction()`

C++ - File Handling Classes Modes and Member Functions

- I. In file handling, the data is stored permanently in the secondary storage device (hard disk).
- II. File Handling is used for open, close, read, write.
- III. Earlier `cin` and `cout` have been using `iostream.h` for both these objects. When `iostream` does not use this header file, `cin` of `istream` class and `cout` of `ostream` class do not have any value. The same is true for File Handling as well.

IV. File Handling in C++ is a topic that has been given a separate header file, its name is `fstream`.

Why use File Handling?

- I. In the program `cin` and `cout`, it stores the memory only for a short time that is, when the programmer closes the program, then all the data of the program is destroyed.
- II. Programmer uses some variables, arrays, structures, unions to store data in the program, but this data is not stored permanently.
- III. File handling is used only to store it permanently.
- IV. Files created during file handling, whether they are of different types (`.txt`, `.doc` etc.), are portable. It is also used in other computers.

Function/Method	Description	Usage Example
<code>open()</code>	Opens a file for reading or writing. You can specify file access mode (e.g., <code>ios::in</code> , <code>ios::out</code> , <code>ios::app</code>)	<code>file.open("example.txt", ios::in)</code>
<code>close()</code>	Closes the file when you're done reading/writing to it.	<code>file.close()</code>
<code>is_open()</code>	Checks whether the file was successfully opened. Returns true if the file is open, false otherwise.	<code>if (file.is_open()) { ... }</code>
<code>eof()</code>	Returns true if the end-of-file (EOF) is reached while reading from a file.	<code>if (file.eof()) { ... }</code>
<code>fail()</code>	Returns true if the last file operation failed (e.g., file not found or invalid operation).	<code>if (file.fail()) { ... }</code>
<code>tellg()</code>	Returns the current position of the get pointer in an input file stream.	<code>position = file.tellg()</code>
<code>tellp()</code>	Returns the current position of the put pointer in an output file stream.	<code>position = file.tellp()</code>

Function/Method	Description	Usage Example
<code>seekg()</code>	Sets the position of the get pointer in an input file stream.	<code>file.seekg(0, ios::beg)</code>
<code>seekp()</code>	Sets the position of the put pointer in an output file stream.	<code>file.seekp(0, ios::beg)</code>
<code>getline()</code>	Reads a line of text from the file and stores it in a string.	<code>getline(file, str)</code>
<code>read()</code>	Reads binary data from the file. Writes data into a memory location (buffer).	<code>file.read(reinterpret_cast<char*>(&num), sizeof(num))</code>
<code>write()</code>	Writes binary data to a file from a buffer.	<code>file.write(reinterpret_cast<char*>(&num), sizeof(num))</code>
<code>flush()</code>	Flushes the output buffer to ensure that all buffered output is written to the file.	<code>file.flush()</code>

Example Code using these functions:

```

cpp
Copy
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    // Example for opening a file
    ofstream outFile("example.txt");

    if (!outFile.is_open()) {
        cout << "File could not be opened!" << endl;
        return 1;
    }

    outFile << "Hello, File Handling in C++!" <<
endl;

    // Check if the write failed
    if (outFile.fail()) {
        cout << "Error writing to file!" << endl;
        return 1;
    }

    outFile.close(); // Close the file after writing
  
```

```

// Reading from the file
ifstream inFile("example.txt");

if (!inFile.is_open()) {
    cout << "File could not be opened!" << endl;
    return 1;
}

string line;
while (getline(inFile, line)) { // Read file line
    by line
        cout << line << endl; // Output each line
        to console
    }

    inFile.close(); // Close the file after reading

    return 0;
}
  
```

Explanation of File Handling Functions:

- `open()`: Opens the file for reading/writing in a specified mode.
- `close()`: Closes the file stream and releases resources.
- `is_open()`: Checks if the file is open successfully.
- `eof()`: Used to check if the end of the file has been reached.



- `fail()`: Checks if a file operation failed (such as reading a non-existent file).
- `tellg()` and `tellp()`: Returns the current position of the input/output file pointers.
- `seekg()` and `seekp()`: Moves the file pointers to specific positions in the file (useful for random access).
- `getline()`: Reads a complete line of text from a file.
- `read()`: Reads binary data from a file into memory.
- `write()`: Writes binary data to a file.
- `flush()`: Ensures that the data is actually written to the file by flushing the output buffer.

C++ - Introduction of Classes and Objects

- I. Class and Object is very important part for OOP.
- II. The concept of OOP is dependent on Objects and Classes.
- III. There can be many member functions and data members inside the class. Which is accessed through object.
- IV. The variables which are inside the class are called data members and the functions which are there are called member functions.
- V. for eg.
- VI. If there is a creature, then the behavior and properties of that creature such as its barking, walking, watching, the composition of its body are written through data members and member functions.
- VII. Each more than one creature is also made their objects by different names.

What is class?

- I. Class is like a structure. In which variable(data members) and function(member functions) are collected at one place.
- II. Data and functions are the members of the class.
- III. Works to hold class data.
- IV. Class is the layout of the object.

Defining Class

The class keyword is used to define the class. With this goes 'Classname'. The first character in the class name can be either uppercase or lowercase. But having the first letter of the class name in uppercase is called a 'Good Programming'.

There are some more important parts for the class. For example, Access Specifier

Access Specifier / Modifier is used to control access.

There are three types of Access Specifiers.

- I. private
- II. public
- III. protected

1. Private: In private the class members mean variables are written. Without private the members of private are written, that is the default private member. Private member works only for his class. These are not accessible outside the class.

2. Protected: The work of protected is like private. It is inherited. They are used in inheritance.

3. Public: Members of public are accessed both inside and outside the class. In this, data members are also written.

Syntax for Class

```
class Classname{
    data member(s);
    member function(s);
};
```

For Example

In the example below, the name of the class with the keyword 'class' is 'Number' and the curly brace is open({) . With this data members and member function have been defined. After this, the curly brace close(}) is followed by semicolon (;) at the end.

```
class Number
```

```
{
```

```
    private:
```

```
        int a; //data member
```

```
    public:
```

```
        getdata();    //member
```

```
    function
```

```
};
```



Java

A class is a user defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type.

Object- is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of:

1. **State** : It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior** : It is represented by methods of an object. It also reflects the response of an object with other objects.
3. **Identity** : It gives a unique name to an object and enables one object to interact with other objects.
4. **Method**: A method is a collection of statements that perform some specific task and return result to the caller. A method can perform some specific task without returning anything. Methods allow us to **reuse** the code without retyping the code. In Java, every method must be part of some class which is different from languages like C, C++ and Python.

Abstraction

Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essentials units are not displayed to the user.

Encapsulation

It is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield.

- Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared.
- As in encapsulation, the data in a class is hidden from other classes, so it is also known as **data-hiding**.
- Encapsulation can be achieved by Declaring all the variables in the class as private and writing public

methods in the class to set and get the values of variables.

Inheritance

I. Inheritance in Java is a concept that acquires the properties from one class to other classes; for example, the relationship between father and son.

II. In Java, a class can inherit attributes and methods from another class.

III. The class that inherits the properties is known as the sub-class or the child class.

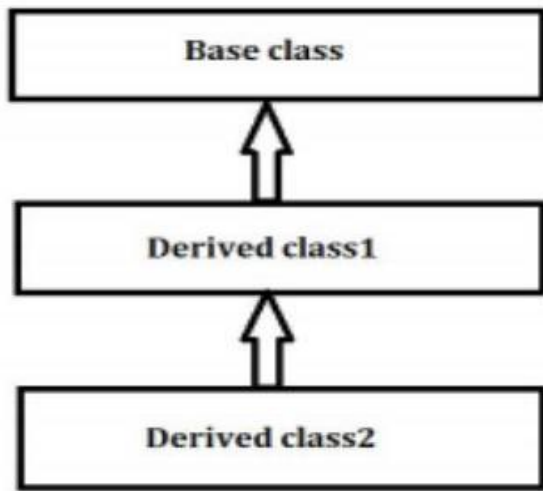
Let us discuss some of frequent used important terminologies:

- **Super Class**: The class whose features are inherited is known as superclass (or a base class or a parent class).
- **Sub Class**: The class that inherits the other class is known as subclass (or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.
- **Reusability**: Inheritance supports the concept of "reusability", i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

Types of Inheritance

There are three inheritance in Java.

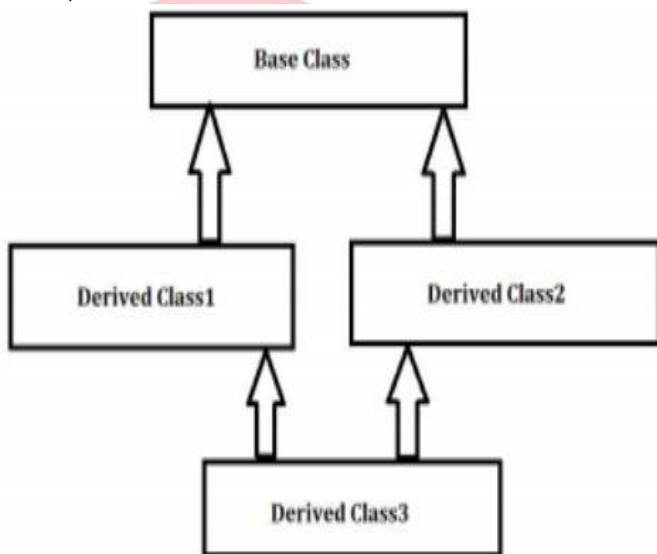
1. **Single Inheritance**-Creating subclasses from a single base class is called single inheritance.
2. **MultiLevel Inheritance**-
 1. Defining derived class from numerous base classes is known as 'Multiple Inheritance'. In this case, there is more than one superclass, and there can be one or more subclasses.
 - II. Multiple inheritances are available in object-oriented programming with C++, but it is not available in Java.
- III. Java developers want to use multiple inheritances in some cases. Fortunately, Java developers have interface concepts expecting the developers to achieve multiple inheritances by using multiple interfaces.



3. Hierarchical Inheritance-

- I. In Hierarchical Inheritance in Java, more than one derived class extends a single base class.
- II. In simple words, more than one child class extends a single parent class or a single parent class has more than one child class.

Examples -



1. Single Inheritance

Source Code :

```

//B.java
class A
{
    void disp()
    {
        System.out.println("Parent
Class");
    }
}
  
```

class B extends A

```

{
    void show()
    {
        System.out.println("Child
Class");
    }
    public static void
    main(String args[])
    {
        B obj = new B();
        obj.disp();
        obj.show();
    }
}
  
```

Output :-

Parent Class
Child Class

2. Multilevel Inheritance

Source Code :

```

//C.java
class A
{
    void disp()
    {
        System.out.println("Class
A");
    }
}
class B extends A
{
    void show()
    {
        System.out.println("Class
B");
    }
}
class C extends B
{
    void getdata()
    {
  
```



```

C");
    }
    public static void
main(String args[])
{
    C obj = new C();
    obj.disp();
    obj.show();
    obj.getdata();
}

```

Output :

Class A

Class B

Class C

3. Hierarchical Inheritance

Source Code :

```

//C.java
class A{

```

```

    void disp()
{
    System.out.println("Class
A");
}
class B extends A
{
    void show()
    System.out.println("Class
B");
}
class C extends A
{
    void getdata()
    System.out.println("Class
C");
}
public static void
main(String args[])
{
    C obj = new C();
    obj.disp();
}

```

```

obj.getdata();
B b = new B();
b.show();
}

```

```

}

```

Output :

Class A

Class C

Class B

4. Multiple Inheritance

- I. For Multiple Inheritance, Java has more than one base class and all base classes inherit the same derived class.
- II. Multiple Inheritance in Java; **Does not support**, but supports Multiple Inheritance in Java through interface.
- III. Multiple Inheritance; Why is it not supported in Java?
- IV. When Multiple Inheritance comes in Java, then Ambiguity problem comes. Even with extends in Java, there is no mention of Multiple Inheritance. But it can be used in interface.

Java cannot have more than one parent class. There are three classes in the program. C class; A and B inherit both these classes.

Constructor

- I. This is a special method of a class, initializes the object of the same class on which it is created.
- II. The class of which the constructor is made, if when the object of the same class is created, then it is automatically called.

Constructor is similar to its class name, but Constructor has no return type.

Characteristics of Constructor

There are the following characteristics of constructor in java. They are as follows:

- I. Constructor's name must be the same as the name of the class in which it is declared and defined.
- II. The constructor should not have any return type even void also because if there is a return type then JVM would consider as a method, not a constructor.
- III. Compiler and JVM differentiate constructor and method definitions on the basis of the return type.

- IV. Suppose you define the method and constructor with the same name as that of the class name then JVM would differentiate between them by using return type.
- V. Whenever we create an object/instance of a class, the constructor will be automatically called by the JVM (Java Virtual Machine).
- VI. If we don't define any constructor inside the class, Java compiler automatically creates a default constructor at compile-time and assigns default values for all variables declared in the class.

The default values for variables are as follows:

- a. Numeric variables are set to 0.
 - b. Strings are set to null.
 - c. Boolean variables are set to false.
- VII. Java constructor may or may not contain parameters. Parameters are local variables to receive value (data) from outside into a constructor.
 - VIII. A constructor is automatically called and executed by JVM at the time of object creation.
 - IX. JVM first allocates the memory for variables (objects) and then executes the constructor to initialize instance variables.
 - X. It is called and executed only once per object. This means that when an object of a class is created, constructor is called. When we create second object then the constructor is again called during the second time.

Syntax

```
class class_name
{
    -----
    class_name( )
{
    // Constructor
    //statements;
}
    -----
}
```

Example

```
class A
{
    -----
    A () { // Constructor
    //statements;
}
```

}

Three Types of Constructor

1. Default Constructor
2. Parameterized Constructor
3. Constructor Overloading

Default Constructor

- I. Default constructor does not take any parameter or argument.
- II. The above program is of Default Constructor.
- III. There is a class named 'A' in the program and its constructor has been created. When the object of A class is created then the constructor will be called automatically.
- IV. As many times as the object of A class is created, the constructor is called.

Parameterized Constructor

- I. In Parameterized Constructor, parameters are passed to the constructor.
- II. In Parameterized Constructor, different arguments are given to the constructor. There is no limit to arguments in this.
- III. In Parameterized Constructor, the values of the parameters have to be given in the object of the class.
- IV. For the program addition given below, two values have been initialized and their addition has been done.

3. Constructor Overloading

- In Constructor Overloading, multiple constructor overloading can be done in the class, only their number of parameters and their type are different.
- Constructor overloading is similar to Function overloading.

Copy Constructor

There is no concept of copy constructor in Java. But in the constructor an object can be copied to another object.

Polymorphism

- I. Polymorphism is a very good feature of Object Oriented Programming.
- II. Having multiple forms in a single form is polymorphism.

III. **Polymorphism** -The word 'poly' and 'morph' have been formed by combining these two words.

IV. It refers to the ability of OOPs programming languages to differentiate between entities with the same name efficiently.

Real Life Example for Polymorphism

1. There are many companies of smartphones in real life. When a smartphone is used, then that mobile has many features like Camera, Calling, Music Player, Video Player.
- II. Here the name mobile is the same but it has many forms.

There are two types for Polymorphism in Java.

1. Compile-Time Polymorphism
2. Run-Time Polymorphism

I. **Compile-Time Polymorphism:**

1. Compile Time Polymorphism is also called Static Polymorphism.
- II. Method overloading happens here.
- III. In which the names of the same type of methods and their parameters and the types of those parameters are different.

Polymorphism in Java are mainly of 2 types:

1. Overloading

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor overloading in Java, that allows a class to have more than one constructor having different argument lists.

2. Overriding

Declaring a method in sub class which is already present in parent class is known as method overriding. Overriding is done so that a child class can give its own implementation to a method which is already provided by the parent class. In this case the method in parent class is called overridden method and the method in child class is called overriding method. In this guide, we will see what is method overriding in Java and why we use it.

2. Run-Time Polymorphism: In Run-Time Polymorphism the method is called by JVM at run time. Method Overriding is a typical example of Run-Time Polymorphism.

Binding

Static Binding

1. Static Binding is also called Early Binding. When the object is fixed by the compiler at compile-time, it is called Static Binding.
- II. Static Binding is there in the class which has private, static or final method.
- III. The method cannot be overridden at compile-time. These methods are accessed from their object. Example for Static Binding

Example

Dynamic Binding

1. Dynamic Binding is also called Late Binding.
- II. This does not happen at compile-time by the Binding Compiler.
- III. Here classes have the same methods.
- IV. Here Method Override happens. Here both the classes have the same signature methods.

Abstract Class and Method

1. **Abstract class:** is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
- II. **Abstract method:** can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

An abstract class can have both abstract and regular methods:

Access

Modifiers

Access Modifiers; Used for accessibility of Classes, Data members, Methods and Constructor.

There are four types of Access Modifiers.

1. default Access Modifier
2. private Access Modifier
3. public Access Modifier
4. protected Access Modifier

I. **default Access Modifier**

1. If no modifier is given then by default default is given.
- II. The default Modifier is only accessible within the package.

Dear Aspirants, here are the our results in differents exams

(Proof Video Link) ↓

RAS PRE. 2021 - <https://shorturl.at/qBJ18> (74 प्रश्न , 150 में से)

RAS Pre 2023 - <https://shorturl.at/tGHRT> (96 प्रश्न , 150 में से)

UP Police Constable 2024 - <http://surl.li/rbfyn> (98 प्रश्न , 150 में से)

Rajasthan CET Gradu. Level - <https://youtu.be/gPqDNlc6UR0>

Rajasthan CET 12th Level - <https://youtu.be/oCa-CoTFu4A>

RPSC EO / RO - <https://youtu.be/b9PKjl4nSxE>

VDO PRE. - <https://www.youtube.com/watch?v=gXdAk856Wl8&t=202s>

Patwari - <https://www.youtube.com/watch?v=X6mKGdtXyu4&t=2s>

PTI 3rd grade - https://www.youtube.com/watch?v=iA_MemKKgEk&t=5s

SSC GD - 2021 - <https://youtu.be/2gz2fJyt6vI>

EXAM (परीक्षा)	DATE	हमारे नोट्स में से आये हुए प्रश्नों की संख्या
MPPSC Prelims 2023	17 दिसम्बर	63 प्रश्न (100 में से)
RAS PRE. 2021	27 अक्तूबर	74 प्रश्न आये
RAS Mains 2021	October 2021	52% प्रश्न आये

whatsapp <https://wa.link/yoh401> 1 web.- <https://shorturl.at/mhX4l>





RAS Pre. 2023	01 अक्टूबर 2023	96 प्रश्न (150 में से)
SSC GD 2021	16 नवम्बर	68 (100 में से)
SSC GD 2021	08 दिसम्बर	67 (100 में से)
RPSC EO/RO	14 मई (1st Shift)	95 (120 में से)
राजस्थान S.I. 2021	14 सितम्बर	119 (200 में से)
राजस्थान S.I. 2021	15 सितम्बर	126 (200 में से)
RAJASTHAN PATWARI 2021	23 अक्टूबर (1st शिफ्ट)	79 (150 में से)
RAJASTHAN PATWARI 2021	23 अक्टूबर (2 nd शिफ्ट)	103 (150 में से)
RAJASTHAN PATWARI 2021	24 अक्टूबर (2 nd शिफ्ट)	91 (150 में से)
RAJASTHAN VDO 2021	27 दिसम्बर (1 st शिफ्ट)	59 (100 में से)
RAJASTHAN VDO 2021	27 दिसम्बर (2 nd शिफ्ट)	61 (100 में से)
RAJASTHAN VDO 2021	28 दिसम्बर (2 nd शिफ्ट)	57 (100 में से)
U.P. SI 2021	14 नवम्बर 2021 1 st शिफ्ट	91 (160 में से)
U.P. SI 2021	21 नवम्बर 2021 (1 st शिफ्ट)	89 (160 में से)
Raj. CET Graduation level	07 January 2023 (1 st शिफ्ट)	96 (150 में से)
Raj. CET 12th level	04 February 2023 (1 st शिफ्ट)	98 (150 में से)
UP Police Constable	17 February 2024 (1 st शिफ्ट)	98 (150 में से)

& Many More Exams like UPSC, SSC, Bank Etc.





whatsapp <https://wa.link/yoh401> 2 web.- <https://shorturl.at/mhX41>




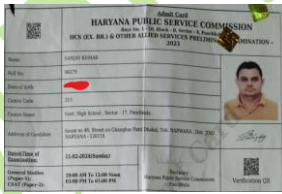
Our Selected Students

Approx. 563+ students selected in different exams. Some of them are given below -

Photo	Name	Exam	Roll no.	City
	Mohan Sharma S/O Kallu Ram	Railway Group - d	114195120370022	PratapNagar Jaipur
	Mahaveer singh	Reet Level- 1	1233893	Sardarpura Jodhpur
	Sonu Kumar Prajapati S/O Hammer shing prajapati	SSC CHSL tier- 1	2006018079	Teh.- Biramganj, Dis.- Raisen, MP
N.A	Mahender Singh	EO RO (81 Marks)	N.A.	teh nohar , dist Hanumang arh
	Lal singh	EO RO (88 Marks)	13373780	Hanumang arh
N.A	Mangilal Siyag	SSC MTS	N.A.	ramsar, bikaner

	MONU S/O KAMTA PRASAD	SSC MTS	3009078841	kaushambi (UP)
	Mukesh ji	RAS Pre	1562775	newai tonk
	Govind Singh S/O Sajjan Singh	RAS	1698443	UDAIPUR
	Govinda Jangir	RAS	1231450	Hanumang arh
N.A	Rohit sharma s/o shree Radhe Shyam sharma	RAS	N.A.	Churu
	DEEPAK SINGH	RAS	N.A.	Sirsi Road , Panchyawa la
N.A	LUCKY SALIWAL s/o GOPALLAL SALI WAL	RAS	N.A.	AKLERA , JHALAWAR
N.A	Ramchandra Pediwal	RAS	N.A.	diegana , Nagaur

	Monika jangir	RAS	N.A.	jhunjhunu
	Mahaveer	RAS	1616428	village- gudaram singh, teshil-sojat
N.A	OM PARKSH	RAS	N.A.	Teshil- mundwa Dis- Nagaur
N.A	Sikha Yadav	High court LDC	N.A.	Dis- Bundi
	Bhanu Pratap Patel s/o bansi lal patel	Rac batalian	729141135	Dis.- Bhilwara
N.A	mukesh kumar bairwa s/o ram avtar	3rd grade reet level 1	1266657	JHUNJHUN U
N.A	Rinku	EO/RO (105 Marks)	N.A.	District: Baran
N.A.	Rupnarayan Gurjar	EO/RO (103 Marks)	N.A.	sojat road pali
	Govind	SSB	4612039613	jhalawad

	Jagdish Jogi	EO/RO Marks)	(84 N.A.	tehsil bhinmal, jhalore.
	Vidhya dadhich	RAS Pre.	1158256	kota
	Sanjay	Haryana PCS	96379 	Jind (Haryana)

And many others.....

Click on the below link to purchase notes

WhatsApp करें -

<https://wa.link/yoh401>

Online Order करें -

<https://shorturl.at/mhX4l>

Call करें - **9887809083**