Roles of Data Types:-

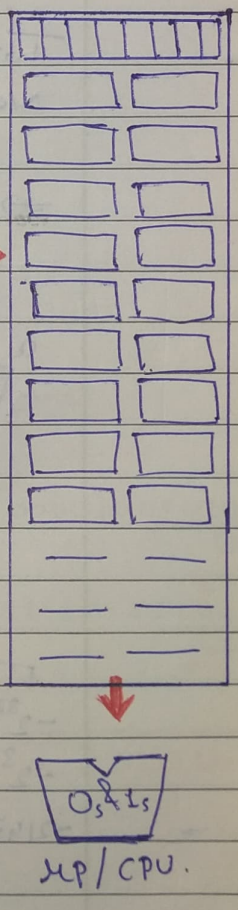
Page No: .....
Date: .....

1. Primary purpose of Datatype in Java is to convert your real world data into the Binary format.
2. Indicate the amount of memory to be allocated on the RAM
3. Specify the type of data stored in a variable.

Job Application (Java)

Photo :-	Image	CGPA :-	Real No
Name :-	String	Backlogs :-	Yes/No
Age :-	Integer	Expected CTC :-	Real No
DOB :-	Date	Interview Slot :-	Time
Gender :-	Character	Grade :-	Character
Email :-	String	Qualification :-	String
Mobile :-	Integer	Resume :-	Document
Address :-	String	Self-intro :-	Video
YOP :-	Date		

Real World Data	Data Types in Java	Formats Followed by Data Types
Primitive	**Primitive Data Types**	
→ Integer Type Data	byte, short, int, long	Base - 2
→ Real Number Type Data	float, double	IEEE 754
→ Character Type Data	char	ASCII/Unicode
→ Yes/No Type Data	boolean	JVM Dependent
Non-Primitive	**In built Classes**	
→ String Type Data	String Classes	ASCII/Unicode
→ Grouped Type Data	Array/Collection Classes	Data Structure Dependent
→ Date/Time Type Data	Date-Time Classes	dd/mm/yy | hh:mm:ss
→ Text Files Type Data	File-stream Classes	.txt/.doc/.pdf/.rtf/.....
→ Image Type Data	Image Classes	.jpeg/.png/.gif/.jpg
→ Video Type Data	Java Media Framework	.mp4/.avi/.mov/...
→ Audio Type Data	"	.mp3/.wav/...



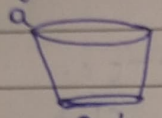
Integer Data Types

Base-2 Format: n-bits

-2^{n-1} to $+2^{n-1}-1$

⇒ byte

byte a;



1 Byte.

8-bits

-2^{8-1} to $+2^{8-1}-1$

-2^{7} to $+2^{7}-1$

-128 to +127

Underflow Overflow

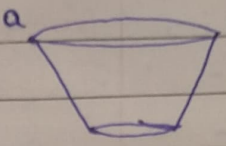
Total Allowed Values :- 256

- age of a person
- ⇒ years of experience
- ⇒ No. of Children

- ⇒ Salary of a fresher
- ⇒ price of a mobile.
- ⇒ max. speed car.

⇒ Short

short a;



2 Bytes.

16-bits.

-2^{16-1} to $+2^{16-1}-1$.

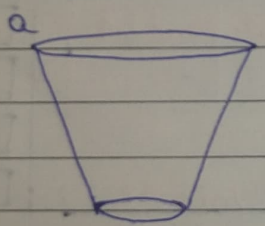
-2^{15} to $+2^{15}-1$.

-32768 to +32767.

65536.

⇒ int

int a;



4 Bytes

32-bits.

-2^{32-1} to $+2^{32-1}-1$.

-2^{31} to $+2^{31}-1$

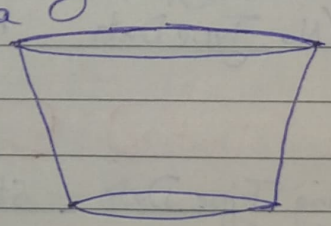
-2147483648 to +2147483647

4294967296

- ⇒ Price of a house.
- ⇒ distance b/w Countries.
- ⇒ Population of a country.

⇒ long

long a;



8 Bytes.

64-bits.

-2^{64-1} to $+2^{64-1}-1$.

-2^{63} to $+2^{63}-1$.

-9223372036854775808L to +9223372036854775807L.

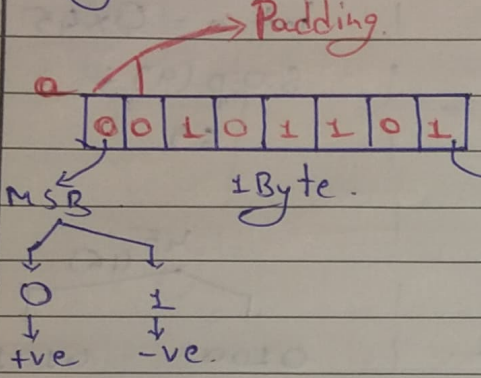
$1.8446744073^{*}10^{19}$

- ⇒ Mobile No.
- ⇒ dist. b/w galaxies
- ⇒ Population of the world.

Date:

#How Integer Data is Stored in Memory?

⇒ byte a = 45 .



+ Base - 2 Format

2	45
2	22 → 1
2	11 → 0
2	5 → 1
2	2 → 1
	1 → 0

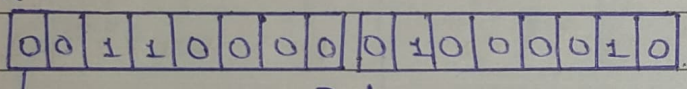
45 ⟺ 101101
(10) (2).

⇒ short a = 12354;

Base - 2 Format.

2	12354.
2	6177 → 0
2	3088 → 1
2	1544 → 0.
2	772 → 0
2	386 → 0
	⋮
	1 → 1.

a.

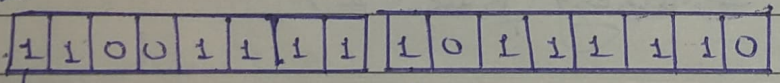


12354 ⟺ 11000001000010.
(10) (2).

⇒ Short a = -12354;

Base - 2 Format

a



12354 ⟺ 0011000001000010
(10) (2)

1's Complement ⟺ 1100111110111101
(2)
+1

∴ Binary Addition

2's Complement ⟺ 1100111110111110

Decimal (Base-10)

```
int a = 45;
S.o.p(a);
```

Output:- 45.

Input:- $45_{(10)}$

2	45	
2	22	→ 1
2	11	→ 0
2	5	→ 1
2	2	→ 1
	1	→ 0

Memory:- 101101 (2)

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$1\times2^5 + 1\times2^3 + 1\times2^2 + 1\times2^0$

$32 + 8 + 4 + 1 \Rightarrow 45$

Output:- $45_{(10)}$ //

Octal (Base-8)

```
int a = 045;
S.o.p(a);
```

37.

$45_{(8)}$ → 100 101.

Memory:- 100101 (2)

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$1\times2^5 + 1\times2^2 + 1\times2^0$

$32 + 4 + 1$

$\Rightarrow 37$.

Output:- $37_{(10)}$ //

Hexadecimal (Base-16)

```
int a = 0x45;
S.o.p(a);
```

69

$45_{(16)}$ → 0100 0101.

Memory:- 01000101 (2)

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$1\times2^6 + 1\times2^2 + 1\times2^0$

$64 + 4 + 1 \Rightarrow 69$.

Output:- $69_{(10)}$ //

Binary (Base-2)

```
int a = 0b101101;
S.o.p(a);
```

Output :- 45

Input :- $101101_{(2)}$.

Memory:- $101101_{(2)}$

$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$1\times2^5 + 1\times2^3 + 1\times2^2 + 1\times2^0$

$32 + 8 + 4 + 1 \Rightarrow 45$

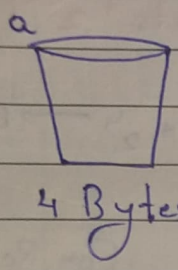
Output :- $45_{(10)}$ //.

Page No:
Date:

Real Number Data Types.

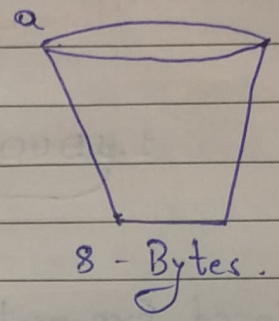
float.

float a;

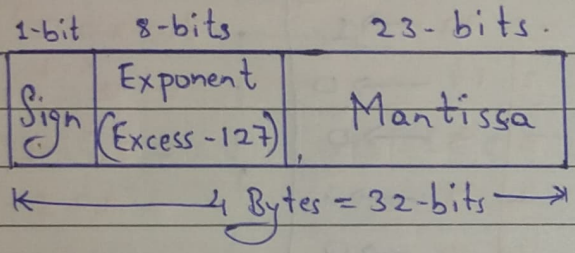


double

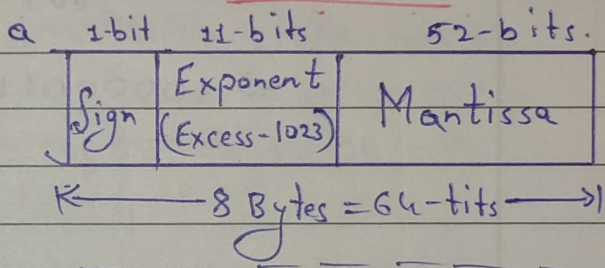
double a;



IEEE-754 Single-precision Format



IEEE-754 Double-precision Format



How Real Number Data is Stored in Memory?

float a = 24.17f;

suffix

(IEEE-754 Single-Precision Format)

Integral Part

Decimal Point

Fractional Part.

2	24
2	12 → 0
2	6 → 0
2	3 → 0
	1 → 1

$24_{(10)} \Longleftrightarrow 11000_{(2)}$

.17 * 2 = 0.34 → 0.
.34 * 2 = 0.68 → 0
.68 * 2 = 1.36 → 1
.36 * 2 = 0.72 → 0
.72 * 2 = 1.44 → 1
.44 * 2 = 0.88 → 0
.88 * 2 = 1.76 → 1
.76 * 2 = 1.52 → 1
.52 * 2 = 1.04 → 1
.....
.....
.....

$.17_{(10)} \Leftrightarrow .00101011100001010\ldots_{(2)}$.

$24.17_{(10)} \Leftrightarrow 11000.00101011100001010\ldots_{(2)}$. {Standard form}

⇩

Normalized Form {1. Significant bits * 2^{exp}}

⇩

$1.1000001010111000001010\ldots * 2^{4}$ → Exponent.

Mantissa.

For Reference.

⇒ $5265.86_{(10)}$.

⇒ $5.26586 * 10^{3}$

Bias = $2^{n-1} - 1$

$= 2^{8-1} - 1$

$= 2^{7} - 1$

$= 128 - 1$

⇒ 127 //.

\# Biased Exponent = Exponent + Bias.

$= 4 + 127$.

$= 131_{(10)}$

2	131
2	65 → 1
2	32 → 1
2	16 → 0
2	8 → 0
2	4 → 0
2	2 → 0
	1 → 0

⇒ $10000011_{(2)}$

Exponent.

(Excess-127)

a

Sign	Exponent	Mantissa.
0	10000011	10000010101110000...
1-bit	8-bits	23-bits.

← 4 Bytes = 32-bits →

float a = -24.17f ;.

(Excess-127)

a

Sign	Exponent	Mantissa.
1	10000011	10000010101110000...
1-bit	8-bits.	23-bits.

← 4 Bytes = 32-bits →

Page No. .....
Date: .....

double a = 844.726;

[IEEE-754 Double-Precision Format].

2 | 844.		0.726 * 2 ⇒ 1.452 ⟶ 1.
2 | 422 ⟶ 0.		0.452 * 2 ⇒ 0.904 ⟶ 0.
2 | 211 ⟶ 0		0.904 * 2 ⇒ 1.808 ⟶ 1.
2 | 105 ⟶ 1		0.808 * 2 ⇒ 1.616 ⟶ 1.
2 | 52 ⟶ 1.		0.616 * 2 ⇒ 1.232 ⟶ 1.
2 | 26 ⟶ 0		0.232 * 2 ⇒ 0.464 ⟶ 0.
2 | 13 ⟶ 0.		0.464 * 2 ⇒ 0.928 ⟶ 0.
2 | 6 ⟶ 1.		0.928 * 2 ⇒ 1.856 ⟶ 1.
2 | 3 ⟶ 0.		0.856 * 2 ⇒ 1.712 ⟶ 1.
1 ⟶ 1.		0.712 * 2 ⇒ 1.424 ⟶ 1.
		0.424 * 2 ⇒ 0.848 ⟶ 0.
		0.848 * 2 ⇒ 1.696 ⟶ 1.

$844_{(10)} \Leftrightarrow 1101001100.$

$0.726_{(10)} \Leftrightarrow .101110011101\ldots_{(2)}$

$844.726_{(10)} \Leftrightarrow 1101001100.101110011101\ldots$ [Standard Form]

⇩

Normalized Form {1.Significant bits * 2^{exp}}

⇩

$1.10100110010111001110 1\ldots * 2^{9}$ ⟶ Exponent.

Mantissa.

Biased Exponent = Exponent + Bias.

⇒ 9 + 1023

⇒ 1032.

2 | 1032.	
2 | 516 ⟶ 0.	
2 | 258 ⟶ 0	
2 | 129 ⟶ 0.	
2 | 64 ⟶ 1.	
2 | 32 ⟶ 0	
2 | 16 ⟶ 0.	
2 | 8 ⟶ 0	
2 | 4 ⟶ 0	
2 | 2 ⟶ 0	
1 ⟶ 0.	

a

Sign	Exponent [Excess-1023]	Mantissa.
0	10000001000	10100110010111001110 1....
1-bit	11-bits	52-bits.

⟵ 8 Bytes = 64-bits ⟶

Page No:
Date:

Precision of Real Number Data Types:

(N.F) 24.17 $\Leftrightarrow$ $1.10000010101110000101000111101011000\ldots \times 2^4$
(10)

Loss of Precision

float (4-Bytes) 23-bit Mantissa.

double (8 Bytes) 52-bit Mantissa.

Floating-Point Literals:

1. Standard Notation {American Notation}.
2. Scientific Notation {E}.

Standard Notation	Power of 10 Notation	Scientific Notation (E)
123.4	$1.234 * 10^2$	1.234E2 / 1.234F+2 / 1.234e2. (optionally)
72.543	7.2543×10	7.2543E1 (Case-insensitive)
400	$4.0 * 10^2$	4.0e2 / 4e2
0.354	$3.54 * 10^{-1}$	3.54E-1 (Mandatory)
-0.0003159	$-3.159 * 10^{-4}$	-3.159e-4

```
public class Main
{
    public static void main(String[] args)
    {
        double a = 123.4;
        double b = 1.234e2;
        System.out.println(a);
        System.out.println(b);
    }
}
```

Output:
123.4
123.4

Date: .....

Range of Real Number Data Types:-

float :- ± 3.4e38 to ± 3.4e-38.

double :- ± 1.7e308 to ± 1.7e-308.

IEEE-754.

---

double a = 45.75D; → Optional.
↳ Case insensitive.

45.75 → "double" → Java compiler.

~~float a = 45.75;~~ ✗ Error :- Incompatible Types.

① float a = 45.75f; → Mandatory.
↳ Case Insensitive.

② float a = (float) 45.75;
↳ Type Casting.

Character Data Type.

⇒ Character Encoding

Symbols	Binary Code	Symbols	Binary Code	Symbols	Binary Code	Symbols	Binary Code
A	0	A	00	A	000	A	0000
B	1	B	01	B	001	B	0001
2 Symbols ⇓		C	10	C	010	C	0010
2^1 Symbols ⇓		D	11	D	011	D	0011
1-bit binary code		4 Symbols ⇓		E	100	E	0100
		2^2 Symbols ⇓		F	101	F	0101
		2-bit binary code		G	110	G	0110
				H	111	H	0111
				8 Symbols ⇓		I	1000
				2^3 Symbols ⇓		J	1001
				3-bit binary code		K	1010
						L	1011
						M	1100
						N	1101
						O	1110
						P	1111
						16 Symbols ⇓	
						4-bit Binary Code ⇐ 2^4 Symbols	

[ANSI]
American National Standards Institute.

1960s

[ASCII]
American Standard Code for Information Interchange
128 Symbols ⇒ 2^7 Symbols ⇓
7-bit binary code

ASCII Charset

Code Point

Symbols	Decimal	Binary Code.
I Symbol.	0	0000000
II Symbol.	1	0000001.
III Symbol.	2	0000010.
⋮	⋮	⋮
A	65	1000001.
B	66	1000010.
C	67	1000011.
⋮	⋮	⋮
a	97	1100001
b	98	1100010
c	99.	1100011
⋮	⋮	⋮
Penultimate Symbol	126	1111110.
Ultimate Symbol.	127.	1111111.

A ⟺ 65

2	65
2	32 → 1
2	16 → 0
2	8 → 0
2	4 → 0
2	2 → 0
	1 → 0.

a ⟺ 97.

2	97
2	48 → 1
2	24 → 0
2	12 → 0
2	6 → 0
2	3 → 0
	1 → 1

C Language ⇒ ASCII ⇒ 7-bit Binary Code ⇒ Stored as 8-bits. [with 1 unused bit]

Data: RAJA [Real world Format] ⟷ US-ASCII

R

0	1010010

1 Byte.

A

0	1000001

1 Byte

J

0	1001010

1 Byte

A

0	1000001

1 Byte

R.

2	82.
2	41 → 0
2	20 → 1
2	10 → 0
2	5 → 0
2	2 → 1
	1 → 0

J

2	74
2	37 → 0
2	18 → 1
2	9 → 0
2	4 → 1
2	2 → 0
	1 → 0

A.

2	65
2	32 → 1
2	16 → 0
2	8 → 0
2	4 → 0
2	2 → 0
	1 → 0

∴ In C, the size of char data type is 1 Byte.

Page No: .....
Date: .....

Does Java follow ASCII format like C & C++?

ANSI.

1960s 128 symbols.

2^7 Symbols.

7-bit Binary Code with 1 unused bit

ASCII

ABCDEFGH..

+

128 Symbols

8-bits.

2^8 Symbols.

256 Symbols

"ASCII Extensions"

ÀÁÂÃÆÇÈÉ...

"ISO Latin 1"

"Lack of Standardization"

ISCII → India.

VSCII → Vietnam.

The Char data type in Java

Format :- Unicode (UTF-16)

Size :- 2 Bytes (16-bits)

Symbols :- 65,535.

Range :- 0 to 65,535.

Late' 80s

Joe Becker. Xerox.

"XCCS"

"Universal Character Set"

1991

Unicode Consortium.

UNICODE [Unique, Unified, Universal Encoding]

→ UTF-8 [1/2/3/4].

→ UTF-16 [2/4] → JAVA

→ UTF-32 [4].

1991 - V1.0 7129 Symbols.

2023 - V15.1 149813 Symbols.

Modern Use < 2^{14} = 16,384 Symbols.

16-bits.

2^{16} Symbols.

65,536 Symbols

Page No: .....
Date: .....

Unicode Charset

Code Points

Symbols	Binary Code	Hexadecimal	Decimal
I Symbol	0000 0000 0000 0000	0000H	0
II Symbol	0000 0000 0000 0001	0001H	1
III Symbol	0000 0000 0000 0010	0002H	2
⋮	⋮	⋮	⋮
A	0000 0000 0100 0001	0041H	65
B	0000 0000 0100 0010	0042H	66
C	0000 0000 0100 0011	0043H	67
⋮	⋮	⋮	⋮
a	0000 0000 0110 0001	0061H	97
b	0000 0000 0110 0010	0062H	98
c	0000 0000 0110 0011	0063H	99
⋮	⋮	⋮	⋮
अ	0000 1001 0000 0101	0905H	2309
आ	0000 1001 0000 0110	0906H	2310
⋮	⋮	⋮	⋮
অ	0000 1100 1000 0101	0C85H	3205
আ	0000 1100 1000 0110	0C86H	3206
⋮	⋮	⋮	⋮
Penultimate Symbol	1111 1111 1111 1110	FFFEH	65534
Ultimate Symbol	1111 1111 1111 1111	FFFFH	65535

8421 ↓ ↓ 1001 ⇒ 9

8421 ↓ ↓ 1100 ⇒ 12

[In Java, the size of char data type is 2 Bytes.]

Data:- RAJA (Real World Format)

⇕ Unicode (Binary Format).

R		A	
00000000	01010010	00000000	01000001
2 Bytes.		2 Bytes.	

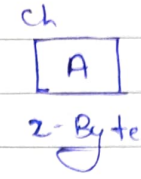
J		A	
00000000	01001010	00000000	01000001
2 Bytes		2 Bytes.	

Date:

Three ways to initialize a character literal to a character variable :-

(1) Directly using the character literal enclosed within single quotes :- 'A'.
⇒ Char ch = 'A';

(2) Using an integral value :-
⇒ Decimal : char ch = 65;
⇒ Octal : char ch = 0101;
⇒ Hexadecimal : char ch = 0x41;
⇒ Binary : char ch = 0b1000001;



(3) Using the Unicode representation :- '\uXXXX' where XXXX represents 4-digit Hexadecimal value of the Unicode character.
⇒ Char ch = '\u0041';
↳ Escape Sequence

Escape Sequences in Java (Control Sequences).

(1) \t :- inserts a tab space.
(2) \b :- inserts a back space.
(3) \n :- inserts a new line.
(4) \r :- inserts a carriage return.
(5) \f :- inserts a form feed.
(6) \' :- inserts a single quote.
(7) \" :- inserts a double quote.
(8) \\ :- inserts a backslash.
(9) \u :- unicode representation of a character.

Expected Output :- Joe's friend said, "Unicode is a Universal Charset".

Input :-
System.out.println("Joe's friend said, "Unicode is a Universal Charset.");

Output :- Compilation Error

Input :-

```
System.out.println("Joe's friend said, \"Unicode is a Universal Charset \".");
```

Output :-

Joe's friend said, "Unicode is a Universal Charset".

Is there any difference in the way how character data '6' is stored in the memory compared to integer data - 6 ?

```
int i = 6;
```

Base-2 Format

2 | 6
2 | 3 → 0
1 → 1

$6_{(10)} \Leftrightarrow 110_{(2)}$

i

00000000 | 00000000

00000000 | 00000110

4-Bytes

```
char c = '6';
```

Unicode Format

Symbol	Unicode Value.
6	54.

2 | 54
2 | 27 → 0
2 | 13 → 1
2 | 6 → 1
2 | 3 → 0
1 → 1.

$6_{(10)} \Leftrightarrow 110110_{(2)}$

c

00000000 | 00110110

2-Bytes.

Important ASCII / UNICODE Values.

A ⇔ 65 (32) 97 ⇔ a

B ⇔ 66 98 ⇔ b

C ⇔ 67 99 ⇔ c.

[Space] ⇔ 32

0 ⇔ 48

1 ⇔ 49

2 ⇔ 50.

Z ⇔ 90 122 ⇔ z 9 ⇔ 57

Page No.:
Date:

Boolean Data Type

Yes / No Type of Data.

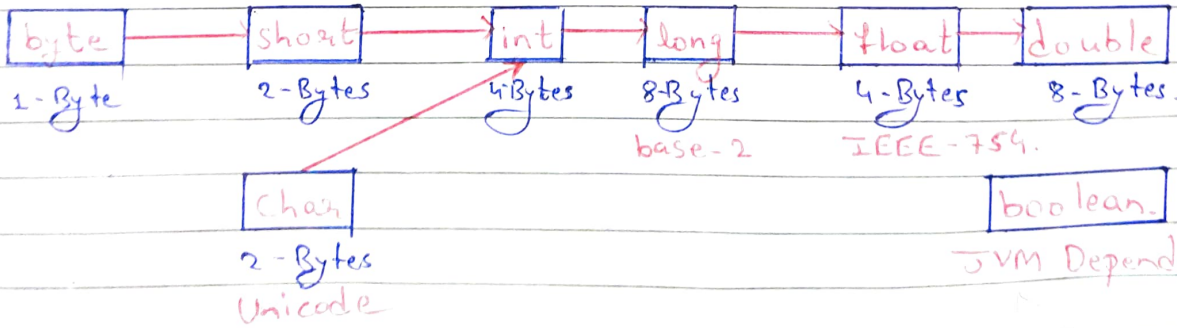
Java	C/C++
boolean.	bool.
boolean a;	bool a;
Literals :- true & false.	true & false. <OR> 0 & 1.
Size :- JVM Dependent. Format :-	1 Byte.
	true => 1. false => 0.

# Valid Syntax:	# Invalid Syntax:
boolean a=true; ✓	boolean a=0; ✗ boolean a="true"; ✗
boolean a=false; ✓	boolean a=True; ✗ boolean a='true'; ✗
	boolean a=TRUE; ✗ boolean a=On; ✗
	boolean a=yes; ✗

Type Casting / Type Conversion

Implicit Type Casting / Numeric Promotion / Type Promotion / Widening / Upcasting

Automatically done by the Compiler



Manually done by the Programmer

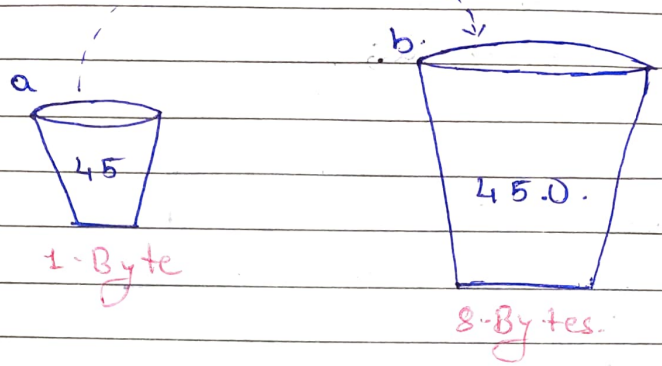
Explicit Type Casting / Narrowing / Downcasting.

Date: .....

Implicit Type Casting / Numeric Promotion / Type Promotion / Widening / Upcasting.

```
byte a = 45;
double b;
b = a;
S.o.p(a);
S.o.p(b);
```

Implicitly done by the Compiler



* Output :- 45
45.0.

Advantage :- No loss of information [data or precision]

Explicit Type Casting

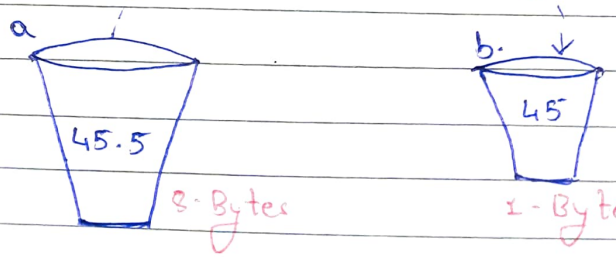
```
double a = 45.5;
byte b;
b = a;
S.o.p(a);
S.o.p(b);
```

XError :- Incompatible Types.

Explicit Type Casting / Narrowing / Down casting.

```
double a = 45.5;
byte b;
b = (byte)a;
S.o.p(a);
S.o.p(b);
```

Explicitly done by the programmer



Output :- 45.5
45

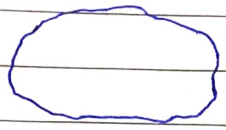
Disadvantage :- Possible loss of information (data or precision).

Date: .....

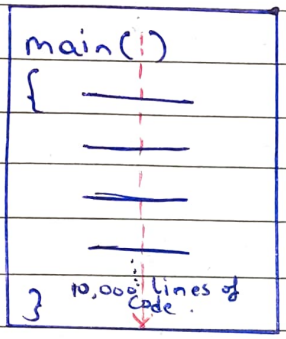
Functions in "C"

Need of Functions :-

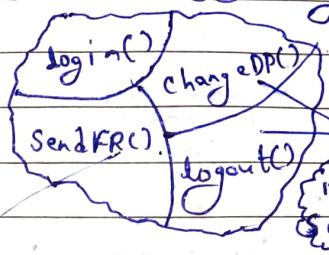
UnStructured Style



{ Unstructured Programming / Non-Structured Programming }

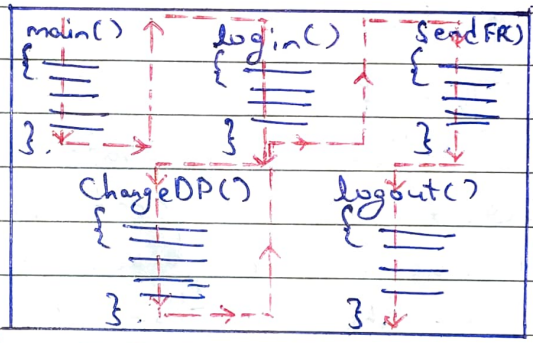


Structured Style



Structured Programming / Modular Programming / Procedure-Oriented Prog... / Procedural Prog... / Functional Prog....

- ⇒ Functions
- ⇒ Procedure
- ⇒ Sub-Program
- ⇒ Module
- ⇒ Sub-Module
- ⇒ Routine
- ⇒ Sub-Routine
- ⇒ Method.



Programming Paradigms :-

1. Unstructured Style.
2. Structured / Procedural / Functional Style.
3. Object - Oriented Style.

⇒ In Unstructured Style of programming, large or complex programs are ~~closed~~ coded as a single-continuous block of code. It makes the program code hardly-readable. Hence, it is difficult to modify & debug.

⇒ Supports limited basic data types.

⇒ Programs are executed sequentially & uses unstructured control statements such as goto.

⇒ Suitable only for small & simple projects.

Example :- Earlier versions of BASIC, COBOL & FORTRAN.

Structured Style of Programming:-

⇒ In Structured Style of Programming, large or complex programs are converted into smaller blocks of code so that managing the code becomes easy. Producing readable code.
⇒ Hence, it is easier to modify & debug.
Supports variety of data types.
⇒ Programs are executed sequentially and uses, structured control statements such as if/else/for/while, etc.
⇒ Suitable for large & complex projects.

⇒ These blocks of code are implemented as functions.
⇒ Therefore, a function is a block of code or collection of statements grouped together to perform a certain task or operation. However, the block of code only runs/executes when the functions is called.

Example :- C, C++, Visual Basic, etc.

Syntax of Functions

⇒ A Function must have the following 4 components :-
(1) Name of the function (2) Input to the function (Optional)
(3) Activity performed by the function (4) Output from the function

⇒ Function Definition :-

```
output name(input)
{
  //activity or task
}
```

⟺

```
return_type name(parameters)
{
  // body of the function
}
```

⇒ Function Invocation :-

```
name(arguments);
```

1. Modularity

Program to add & subtract 2 Numbers

⇒ UnStructured Style.

```
#include <stdio.h>
void main()
{
 int a,b;
 int sum,diff;

printf("Enter the two numbers to add:\n");
scanf("%d %d",&a,&b);
sum=a+b;
printf("The sum is:%d",sum);

printf("Enter the two numbers to subtract:\n");
scanf("%d %d",&a,&b);
diff = a-b;
printf("The difference is:%d",diff);
}
```

Start.

End.

⇒ 3 - Tasks Found :-

1. Addition - add()
2. Subtraction - sub()
3. Execution - main()

Page No: .....
Date: .....

⇒ Structured Style :-

```
#include <stdio.h>
int add(int x, int y)
{
    int res = x + y;
    return res;
}
int sub(int x, int y)
{
    int res = x - y;
    return res;
}
void main()
{
    int a, b;
    int sum, diff;

    printf("Enter the two numbers to add: \n");
    scanf("%d %d", &a, &b);
    sum = add(a,b);
    printf("The sum is: %d", sum);

    printf("Enter the two numbers to subtract: \n");
    scanf("%d %d", &a, &b);
    diff = sub(a,b);
    printf("The difference is: %d", diff);
}
```

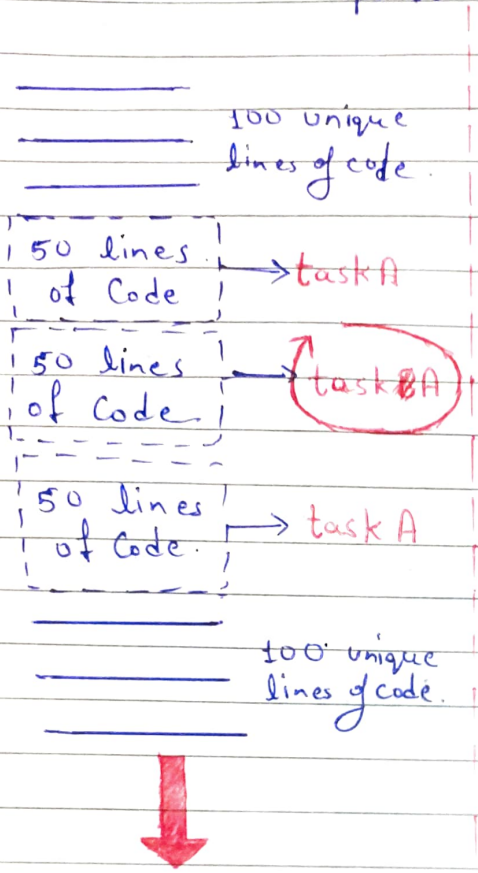
#Output :-

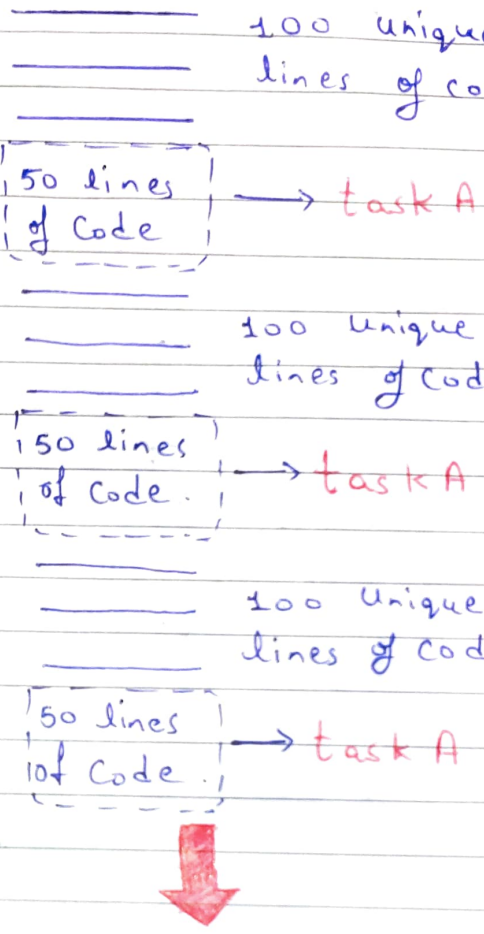
```
Enter the two numbers to add:
10
20
Sum is: 30
Enter the two numbers to subtract:
40
25
The difference is: 25
```

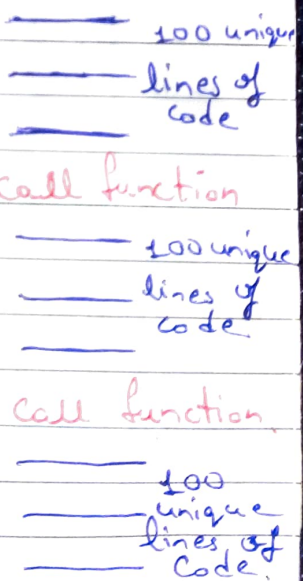
Date: .....

2. Reusability

Loops Vs Functions.







#A function can be defined once & can be called any number of times by passing different arguments.

Page No:
Date:

Check if a given character is Vowel or Consonant.

```
#include<stdio.h>
void main()
{
  ---------
  ---------
  ---------
Char ch='s';
switch(ch)
{
  case'a' :
  case'e' :
  Case'i' :
  Case'o' :
  Case'u' : printf("Vowel");
    break;
  default: printf("Consonant");
}
  ---------
  ---------
  ---------
ch='i';
switch(ch)
{ case'a':
  case'e':
  case'i':
  case'o':
  case'u': printf("Vowel");
  break;
  default: printf("Consonant");
}
  ---------
  ---------
  ---------
ch='m';
switch(ch)
{ case'a':
    :
  case'u': printf("vowel");
    break;
  default: printf("Consonant");
}}
```

```
#include<stdio.h>
Void checkAlphabet(charch)
{
  switch(ch)
  {
    Case'a':
    case'e':
    case'i':
    case'o':
    case'u': printf("Vowel");
      break;
    default: printf("Consonant");
  }
Void main()
{
  ---------
  ---------
  ---------
  ---------
  checkAlphabet('s');
  ---------
  ---------
  ---------
  checkAlphabet('i');
  ---------
  ---------
  ---------
  checkAlphabet('m');
}
```