

LEARN **DSA** WITH C++

WEEK :: 01

LEARN **DSA**
WITH C++

PDF

COURSE INSTRUCTOR BY
ROHIT NEGI

[Check Profile](#)

MADE BY-
PRADUM SINGHA

[My profile](#)

LEARN DSA WITH C++

WEEK :: 01

DAY: 01

DATE: - 17-04-2023

Introduction To Programming

10 years ago, Once upon a time there was a man who had 5 pet animals such as 2 Goats, 2 Cows and one Dog. He was counting these animals using stone. He sees an animal and holds a stone. Thus he counted those animals.

Then time changed, people invented decimal numbers for calculations. Decimal numbers are 0 to 9 (0,1,2,3,4,5,6,7,8, 9). Now people can count from 0 to 1000. But that time was impossible.

Now consider one man doing business. He has four notebooks for calculation. And every notebook has 1000 pages. Now, if you want to calculate those notebooks one by one. Again it's impossible. If you calculate it's a high chance to make a mistake.

Then, Charles Babbage credited a computer called a mechanical computer in 1822. Basic meaning of Computer - **To Calculator**.

That time, A decimal number was used on the computer. Computer size was one room.

A few years later, the transistor was invented. A **transistor** is a semiconductor device used to amplify or switch electrical signals and power. Transistors provide two numbers 0 and 1 (Binary number). Generate 1 when the transistor is on and 0 when the transistor is off.

Decimal Calculation:-

$\begin{array}{r} 15 \\ + 34 \\ \hline 49 \end{array}$	$\begin{array}{r} 47 \\ + 26 \\ \hline 73 \end{array}$	$\begin{array}{r} 56 \\ + 32 \\ \hline 88 \end{array}$
--	--	--

Binary Calculation:-

$\begin{array}{r} 0 \\ + 0 \\ \hline 0 \end{array}$	$\begin{array}{r} 0 \\ + 1 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ + 0 \\ \hline 1 \end{array}$	$\begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$
---	---	---	--

Decimal To Binary :

0	1	2	3	4	5	6	7
0	1	10	11	100	101	110	111

Binary To Decimal: **Exp:- 1** (1 0 1 1 0)

1	0	1	1	0
1×2^4	0×2^3	1×2^2	1×2^1	0×2^0
16	0	4	2	0

Result = $16 + 0 + 4 + 2 + 0 = 22$

Exp:- 2 (1 1 0 1 0 1)

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

Result = $32 + 16 + 0 + 4 + 2 + 1 = 55$

Decimal To Binary : **Exp:- 1** [35]

2	35	
2	17	1
2	8	1
2	4	0
2	2	0
2	1	0
2	0	1

Result = 1 0 0 0 1 1 [**Note: Result take reverse.]

Exp:- 2 [67]

2	67	
2	33	1
2	16	1
2	8	0
2	4	0

2	2	0
2	1	0
2	0	1

Result = 1 0 0 0 0 1 1

Transistor ::

One transistor 2 value 1 either 0.

0	1
---	---

So, **One** memory space: we can put 2 different pieces of data.

1	0
---	---

Two memory spaces: we can put 4 different pieces of data.

0	0	0	1	1	0	1	1
---	---	---	---	---	---	---	---

Four memory spaces: we can put 16 different pieces of data.

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

Moore's Law : Moore's law is the observation that the number of transistors in an integrated circuit (IC) doubles about every two years.

Simple meaning: If there is a transistor in one place. After 2 years there should be 2 transistors in the same place.

50 transistor	After 2 Years	100 transistor
---------------	---------------	----------------

Communication with the computer:

At first we were using assembly language to communicate with the computer. And the assembler converts it to binary code. It was very complicate and code lenght so long.

Now, we use high level languages like C++, Java, and Python. And the compiler or interpreter converts the code into binary code.

Machine Language :: Machine language is a low-level language made up of binary numbers or bits that a computer can understand. It is also known as machine code.

Assembly Language:: An assembly language is a type of low-level programming language that is intended to communicate directly with a computer's hardware.

High Level Language:: A high-level language is any programming language that enables development of a program in a much more user-friendly programming context and is generally independent of the computer's hardware architecture.

Assembler: Assembler is a program for converting instructions written in low-level assembly code into relocatable machine code and generating long information for the loader.

Compiler: A compiler is a special program that translates a programming language's source code into machine code, bytecode or another programming language.

Interpreters : interpreter is a computer program that directly executes instructions written in a programming or scripting language, without requiring them previously to have been compiled into a machine language program.

Importance of Algorithms?

Using Algorithms, our code executes very fast and takes less space for store.

Time Complexity: Total time taken by the algorithms with respect to **input size**.

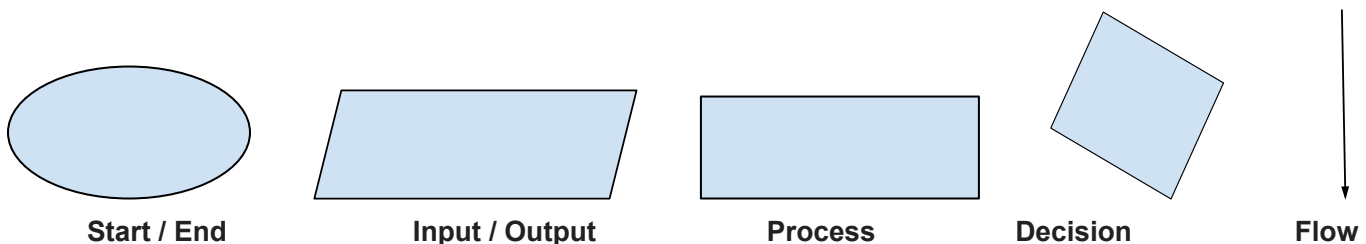
Space Complexity: the amount of memory space required to solve an instance of the computational problem as a function of characteristics of the input.

Importance of DSA?

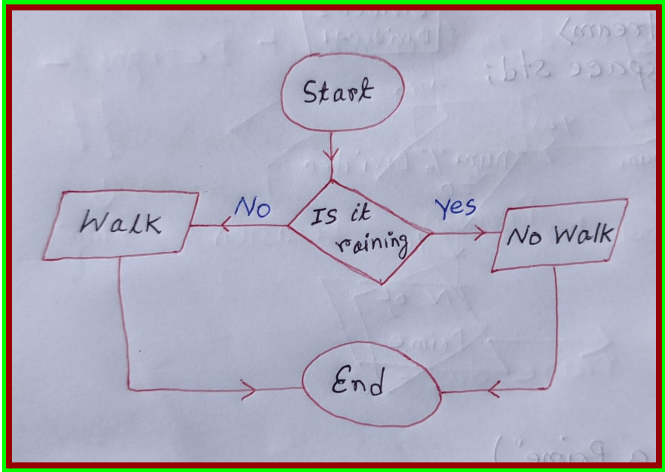
1. Take less Time
2. Take less space

FLOWCHART

FlowChar : A flowchart is a type of diagram that represents a workflow or process.

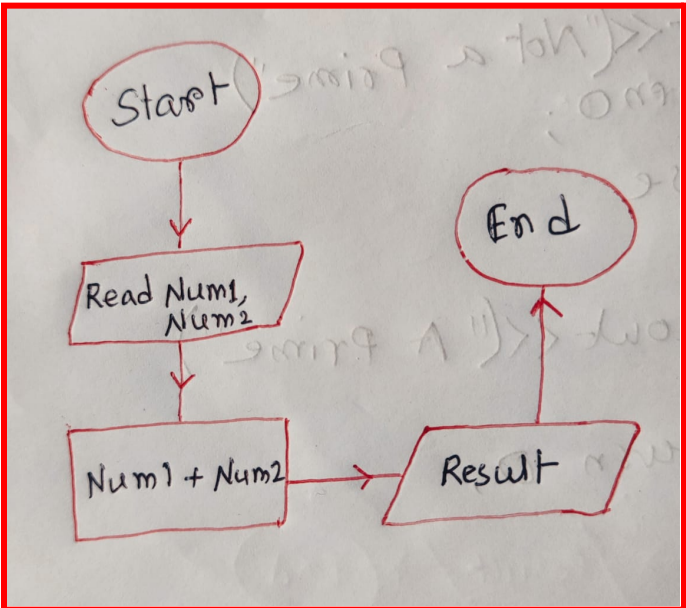


Exp:- 01 **How did you decide whether to go for a walk or stay alone on a rainy day**

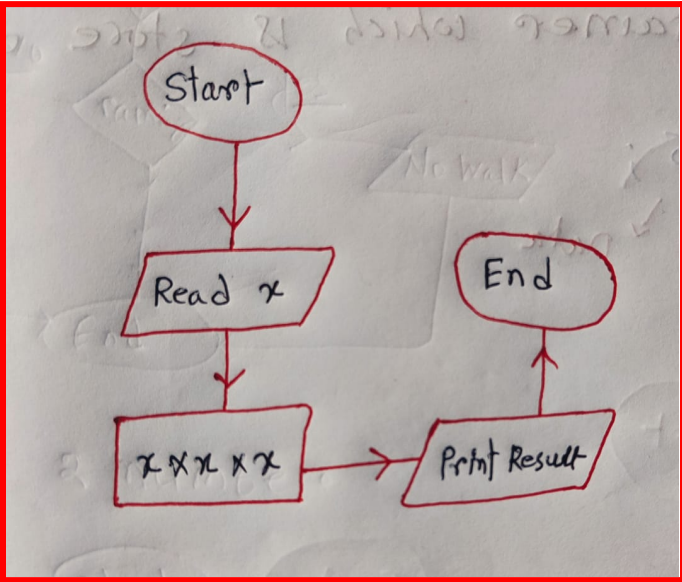
FLOWCHART	PSEUDO CODE
 <pre> graph TD Start([Start]) --> Decision{Is it raining?} Decision -- No --> Walk[Walk] Decision -- Yes --> NoWalk[No Walk] Walk --> End([End]) NoWalk --> End </pre>	<ol style="list-style-type: none"> 1. Know about whether 2. It is rainy or not 3. No rain, go for a walk, It's rainy so, don't go walking.

PSEUDO CODE : Pseudocode is a "text-based" detail (algorithmic) design tool.

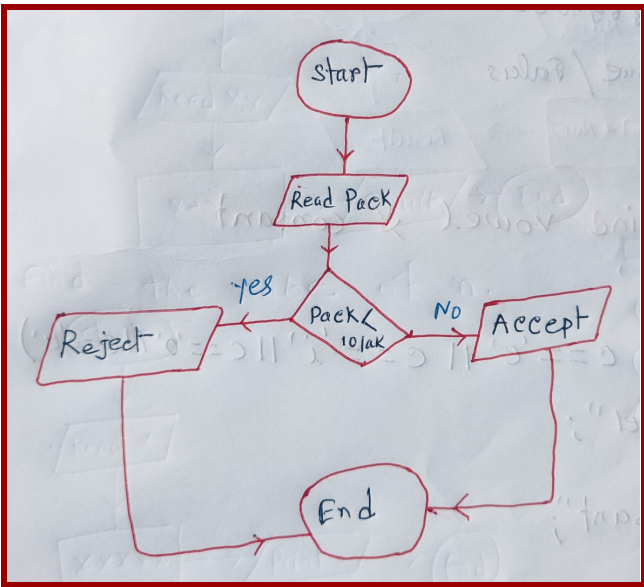
Exp:- 02 :: Add two Number

FLOWCHART	PSEUDO CODE
 <pre> graph TD Start([Start]) --> Read[Read Num1, Num2] Read --> Sum[Num1 + Num2] Sum --> Result[/Result/] Result --> End([End]) </pre>	<ol style="list-style-type: none"> 1. Read Two number 2. number + number 3. Output Sum

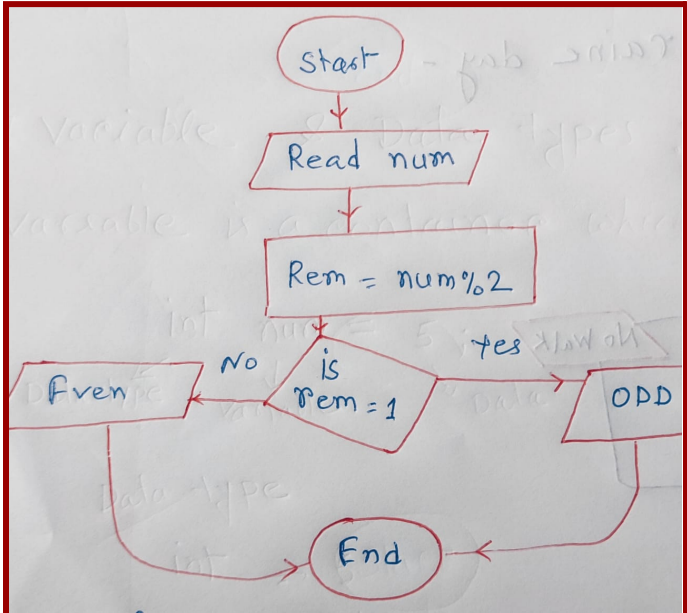
Exp:- 03 :: Find the Cube of Number?

FLOWCHART	PSEUDO CODE
 <pre> graph TD Start([Start]) --> ReadX[/Read x/] ReadX --> CalcCube[x * x * x] CalcCube --> PrintResult[/Print Result/] PrintResult --> End([End]) </pre>	<ol style="list-style-type: none"> 1. Read number 2. number * number * number 3. Output cube

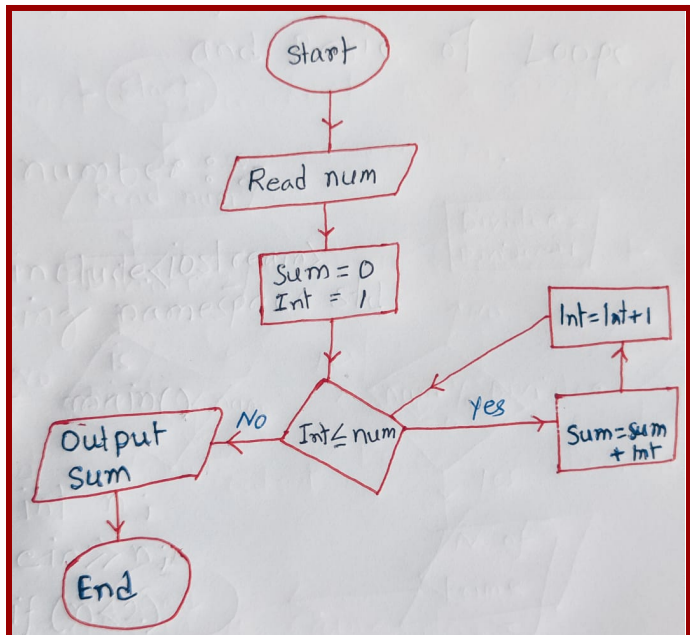
Exp:- 04 :: MNC offer letter?

FLOWCHART	PSEUDO CODE
 <pre> graph TD Start([Start]) --> ReadPack[/Read Pack/] ReadPack --> Decision{Pack < 10 Lak} Decision -- yes --> Reject[/Reject/] Decision -- No --> Accept[/Accept/] Reject --> End([End]) Accept --> End </pre>	<ol style="list-style-type: none"> 1. See Letter 2. Take decision according to package 3. Package suitable so accept or not satisfy don't go.

Exp:- 05 :: Check Number Even and Odd?

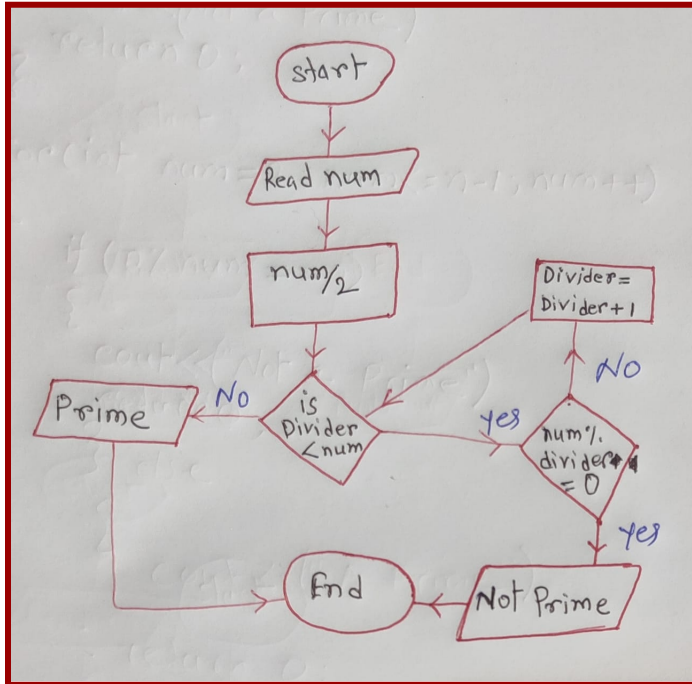
FLOWCHART	PSEUDO CODE
 <pre> graph TD Start([Start]) --> ReadNum[/Read num/] ReadNum --> RemCalc[Rem = num % 2] RemCalc --> IsRem1{is Rem = 1} IsRem1 -- Yes --> ODD[/ODD/] IsRem1 -- No --> Even[/Even/] ODD --> End([End]) Even --> End </pre>	<ol style="list-style-type: none"> 1. Read num 2. Check remainder 3. Remainder is 0 = Even 4. Remainder is not 0 = Odd 5. Print result.

Exp:- 06 :: Sum of n natural numbers?

FLOWCHART	PSEUDO CODE
 <pre> graph TD Start([Start]) --> ReadNum[/Read num/] ReadNum --> Init[Sum = 0
Int = 1] Init --> IsIntLeNum{Int <= num} IsIntLeNum -- Yes --> SumAdd[Sum = Sum + Int] SumAdd --> IntInc[Int = Int + 1] IntInc --> IsIntLeNum IsIntLeNum -- No --> OutputSum[/Output Sum/] OutputSum --> End([End]) </pre>	<ol style="list-style-type: none"> 1. Read num 2. Take variable sum = 0, int = 1 3. Check condition int <= n 4. If break loop Print sum

Exp:- 07 :: Find it is prime or not?

FLOWCHART



PSEUDO CODE

1. Read num
2. Number divided by 2
3. Check condition $\text{num} < \text{divisor}$
4. Agine check condition $\text{num} \% \text{divisor} = 0$
5. It is 0= print Not Prime, 0 != divisor increase 1
6. When condition $\text{divisor} < \text{num}$ false print Prime

[Click Here:: Pradum Singha](#)

INTRODUCTION to C++ and Write First Code

WEEK :: 01

DAY: 02

DATE: - 18-04-2023

COMPUTER MEMORY UNIT ::

One Transistor represents one 1 bit. Every bit stores two values 1 or 0. Smallest unit of memory is a bit.

1 Lak Transistor	1 Lak bit memory
------------------	------------------

1 byte = 8 bit

0	1	0	1	1	0	0	1
---	---	---	---	---	---	---	---

Memory Unit:

Units of Computer Memory Measurements		
1 Bit	=	Binary Digit
8 Bit	=	1 Byte
1024 Bytes	=	1 KB (Kilo Byte)
1024 KB	=	1 MB (Mega Byte)
1024 MB	=	1 GB (Giga Byte)
1024 GB	=	1 TB (Terra Byte)
1024 TB	=	1 PB (Peta Byte)
1024 PB	=	1 EB (Exa Byte)
1024 EB	=	1 ZB (Zetta Byte)
1024 ZB	=	1 YB (Yotta Byte)
1024 YB	=	1 (Bronto Byte)
1024 Brontobyte	=	1 (Geop Byte)

1 Byte	1 KB	1 MB	1 GB
8 bit	2 ¹⁰ Byte	2 ¹⁰ KB	2 ¹⁰ MB

Binary number:

Computers understand only binary numbers.

Decimal - Binary - Octal - Hex – ASCII Conversion Chart

Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII	Decimal	Binary	Octal	Hex	ASCII
0	00000000	000	00	NUL	32	00100000	040	20	SP	64	01000000	100	40	@	96	01100000	140	60	`
1	00000001	001	01	SOH	33	00100001	041	21	!	65	01000001	101	41	A	97	01100001	141	61	a
2	00000010	002	02	STX	34	00100010	042	22	"	66	01000010	102	42	B	98	01100010	142	62	b
3	00000011	003	03	ETX	35	00100011	043	23	#	67	01000011	103	43	C	99	01100011	143	63	c
4	00000100	004	04	EOT	36	00100100	044	24	\$	68	01000100	104	44	D	100	01100100	144	64	d
5	00000101	005	05	ENQ	37	00100101	045	25	%	69	01000101	105	45	E	101	01100101	145	65	e
6	00000110	006	06	ACK	38	00100110	046	26	&	70	01000110	106	46	F	102	01100110	146	66	f
7	00000111	007	07	BEL	39	00100111	047	27	'	71	01000111	107	47	G	103	01100111	147	67	g
8	00001000	010	08	BS	40	00101000	050	28	(72	01001000	110	48	H	104	01101000	150	68	h
9	00001001	011	09	HT	41	00101001	051	29)	73	01001001	111	49	I	105	01101001	151	69	i
10	00001010	012	0A	LF	42	00101010	052	2A	*	74	01001010	112	4A	J	106	01101010	152	6A	j
11	00001011	013	0B	VT	43	00101011	053	2B	+	75	01001011	113	4B	K	107	01101011	153	6B	k
12	00001100	014	0C	FF	44	00101100	054	2C	,	76	01001100	114	4C	L	108	01101100	154	6C	l
13	00001101	015	0D	CR	45	00101101	055	2D	-	77	01001101	115	4D	M	109	01101101	155	6D	m
14	00001110	016	0E	SO	46	00101110	056	2E	.	78	01001110	116	4E	N	110	01101110	156	6E	n
15	00001111	017	0F	SI	47	00101111	057	2F	/	79	01001111	117	4F	O	111	01101111	157	6F	o
16	00010000	020	10	DLE	48	00110000	060	30	0	80	01010000	120	50	P	112	01110000	160	70	p
17	00010001	021	11	DC1	49	00110001	061	31	1	81	01010001	121	51	Q	113	01110001	161	71	q
18	00010010	022	12	DC2	50	00110010	062	32	2	82	01010010	122	52	R	114	01110010	162	72	r
19	00010011	023	13	DC3	51	00110011	063	33	3	83	01010011	123	53	S	115	01110011	163	73	s
20	00010100	024	14	DC4	52	00110100	064	34	4	84	01010100	124	54	T	116	01110100	164	74	t
21	00010101	025	15	NAK	53	00110101	065	35	5	85	01010101	125	55	U	117	01110101	165	75	u
22	00010110	026	16	SYN	54	00110110	066	36	6	86	01010110	126	56	V	118	01110110	166	76	v
23	00010111	027	17	ETB	55	00110111	067	37	7	87	01010111	127	57	W	119	01110111	167	77	w
24	00011000	030	18	CAN	56	00111000	070	38	8	88	01011000	130	58	X	120	01111000	170	78	x
25	00011001	031	19	EM	57	00111001	071	39	9	89	01011001	131	59	Y	121	01111001	171	79	y
26	00011010	032	1A	SUB	58	00111010	072	3A	:	90	01011010	132	5A	Z	122	01111010	172	7A	z
27	00011011	033	1B	ESC	59	00111011	073	3B	;	91	01011011	133	5B	[123	01111011	173	7B	{
28	00011100	034	1C	FS	60	00111100	074	3C	<	92	01011100	134	5C	\	124	01111100	174	7C	
29	00011101	035	1D	GS	61	00111101	075	3D	=	93	01011101	135	5D]	125	01111101	175	7D	}
30	00011110	036	1E	RS	62	00111110	076	3E	>	94	01011110	136	5E	^	126	01111110	176	7E	~
31	00011111	037	1F	US	63	00111111	077	3F	?	95	01011111	137	5F	_	127	01111111	177	7F	DEL

This work is licensed under the Creative Commons Attribution-ShareAlike License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/>

ASCII Conversion Chart.doc Copyright © 2008, 2012 Donald Weisman 22 March 2012

INTRODUCTION TO C++

C++ was developed as an extension of C, and both languages have almost the same syntax. The main difference between C and C++ is that C++ supports classes and objects, while C does not.

First code::

```
int main()
{
    // code
    Return 0;
};
```

```
int main()
{
    cout<<" Hello Coder Army";
    return 0;
};
```

<< :- **Insertion operator**
>> :- **Extraction operator**
cout:- **Print Output**

```
#include<iostream>
using namespace std;

int main()
{
    cout<<"Hello CoderArmy";
    return 0;
};
```

#include<iostream>

the iostream library is an object-oriented library that provides input and output functionality using streams

using namespace std;

we can use names for objects and variables from the standard library. helps in avoiding the ambiguity that may occur when two identifiers have the same name.

Int main()

int main represents that the function returns some integer even '0' at the end of the program execution. Starting point of Program.

Return 0

'0' represents the successful execution of a program.

VARIABLE & DATA TYPES:

Data Types is a Type of data.

Number System = 1, 3, 5, 2.1, 3.1, 0.123, 1.0235

int(integer) = 1, 3, 5,

Float(Floating Point) = 2.1, 3.1

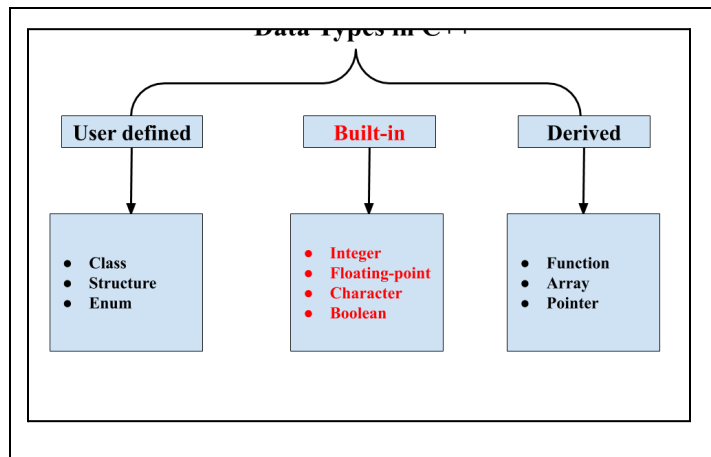
Double(Double Floating Point) = 0.123, 1.0235

Alphabet :- A,V,R,E,J

char(Character) = A,V,R,E,J

bool (Boolean) = Yes/no true/false

Differents types of Data Types:



Size of Data Types in Memory::

Integer data type	ILP32 size	ILP32 alignment	LP64 size	LP64 alignment
char	1 byte	1 byte	1 byte	1 byte
BOOL, bool	1 byte	1 byte	1 byte	1 byte
short	2 bytes	2 bytes	2 bytes	2 bytes
int	4 bytes	4 bytes	4 bytes	4 bytes
long	4 bytes	4 bytes	8 bytes	8 bytes
long long	8 bytes	4 bytes	8 bytes	8 bytes
pointer	4 bytes	4 bytes	8 bytes	8 bytes
size_t	4 bytes	4 bytes	8 bytes	8 bytes
time_t	4 bytes	4 bytes	8 bytes	8 bytes
NSInteger	4 bytes	4 bytes	8 bytes	8 bytes
CFIndex	4 bytes	4 bytes	8 bytes	8 bytes
fpos_t	8 bytes	4 bytes	8 bytes	8 bytes
off_t	8 bytes	4 bytes	8 bytes	8 bytes

Reserved Words in c++::

wchar_t	default	break	case	char	const	continue	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while
delete	dynamic_cast	const_cast	catch	class	namespace	mutable	inline
export	explicit	template	static_cast	reinterpret_cast	public	false	friend
protected	private	true	try	typeid	typename	using	virtual

VARIABLE ::

Variables are containers for storing data values.

int num = 500;

int = Data type

num = Variable name

500 = Value

Print ==

Print int	Print Char
<pre>#include<iostream> using namespace std; int main() { // integer int num = 500; cout<<num; return 0; };</pre>	<pre>#include<iostream> using namespace std; int main() { // Character char c = 'a'; cout<<c; return 0; };</pre>
Print double	Print float
<pre>#include <iostream> using namespace std; int main() { // double double num = 93.3265; cout << num; return 0; };</pre>	<pre>#include<iostream> using namespace std; int main() { // float float d = 1.32; cout<<d; return 0; };</pre>

Comment :- Comments can be used to explain code, and to make it more readable. Compiler is not execute the comment.

```
// This is a comment
```

Single line comment

```
/* The code below will print the words Hello World!
to the screen, and it is amazing */
```

Multiline Comment

How to store Negative Number(-) in memory :-

Num = 9 => 1001

= -9 => Binary ?

2's complement 1001

```

=> 0110 ← 1's Complement
=> 0110
   + 1
   ----
   0111 ← 2's Complement

```

Store this Binary digit for (-9).

sizeof() :- The **sizeof** operator can be used to get the **size of** classes, structures, unions and any other user defined data type.

<pre> int main() { double num = 93.3265; cout << sizeof(num); return 0; }; </pre>	<pre> int main() { int num = 93; cout << sizeof(num); return 0; }; </pre>	<pre> int main() { bool num = 1; cout << sizeof(num); return 0; }; </pre>
Size 8	Size 4	Size 1

Typecasting: A type cast is basically a conversion from one type to another.

<pre> int main() { int num = 'A'; cout<<(num); return 0; }; </pre>	<pre> int main() { char num = 65; cout<<(num); return 0; }; </pre>
Output : 65	Output : A

Here, At first the compiler converts the code into binary numbers. Then, Binary numbers convert into output according to **ASCII TABLE**.

IF-ELSE Condition

The C++ if statement tests the condition. It is executed if the condition is true.

Comparison Operators:

Operator	Name	Example
==	Equal to	x==y
!=	Not equal	x!=y
<	Less than	x<y
>	Greater than	x>y

<=	Less than or equal to	x<=y
>=	Greater than or equal to	>=

Exp :: 01 Print big number. 20, 30

```
#include <iostream>
using namespace std;

int main()
{
    int first_num = 20;
    int second_num = 30;

    if(first_num>second_num)
    {
        cout<<first_num;
    }
    else
    {
        cout<<second_num;
    }
    return 0;
};
```

Exp :: 02 Check number ODD and EVEN.

```
#include <iostream>
using namespace std;

int main()
{
    int num = 25;

    if(num%2==0)
    {
        cout<<"EVEN";
    }
    else
    {
        cout<<"ODD";
    }
    return 0;
};
```


Exp :: 03 Check number Positive, Negative or Zero?

```
#include <iostream>
using namespace std;

int main()
{
    int num = 0;

    if(num>0)
    {
        cout<<"It is Positive Number";
    }
    else if(num<0)
    {
        cout<<"It is Negative Number";
    }
    else
    {
        cout<<"It is zero Number";
    }
    return 0;
};
```

Logical Operator::

&&	⇒ Logical AND	return True if both statements are true	$x < 5 \ \&\& \ x < 10$
	⇒ Logical OR	return True if one statements are true	$x < 5 \ \ x < 4$
!	⇒ Logical NOT	Reverse the result return false if the result is true	$!(x < 5 \ \&\& \ x < 10)$

Exp :: 04 Print big number? 20, 40, 60

```
#include <iostream>
using namespace std;

int main()
{
    int a= 20, b= 40, c= 60;

    if(a>=b && a>=c)
    cout<<a;

    if(b>=a && b>=c)
    cout<<b;
```

```
else  
cout<<c;  
return 0;  
};
```

HOMEWORK

Exp :: 04 Find the given year is leap year or not?

```
#include <iostream>  
using namespace std;  
  
int main()  
{  
    int year;  
    cout<<"Enter Your Year: ";  
    cin>>year;  
  
    if((year%400==0) || (year%4==0 && year%100!=0))  
        cout<<"Leap Year";  
    else  
        cout<<"Not a Leap Year";  
    return 0;  
};
```

If-Else Condition, Operators and Basic of Loops

WEEK :: 01

DAY: 03

DATE: - 19-04-2023

Exp :: 01 Print Vowel and Consonant?

```
#include <iostream>
using namespace std;

int main()
{
    char c = 'h';
    if(c=='a' || c=='e' || c=='i' || c=='o' || c=='u')
        cout<<"Vowel";
    else
        cout<<"Consonant";
    /*if(c=='a')
        cout<<"Vowel";
    else if(c=='e')
        cout<<"Vowel";
    else if(c=='i')
        cout<<"Vowel";
    else if(c=='o')
        cout<<"Vowel";
    else if(c=='u')
        cout<<"Vowel";
    else
        cout<<"Consonant"; */
    return 0;
};
```

Exp :: 02 Given a no. check if it is divisible by 3 & 5?

```
#include <iostream>
using namespace std;

int main()
{
    int num = 27;
    if(num%3==0 && num%5==0)
        cout<<"Perfect";
    else
        cout<<"Not Perfect";
    return 0;
};
```

TAKE INPUT FROM USER `cin>>"Your Value";`

Exp :: 03 :: Add two numbers and Take input from the user?

```
#include <iostream>
using namespace std;

int main()
{
    int num1, num2;
    cout<<"Enter num1: ";
    cin>>num1;
    cout<<"Enter num2: ";
    cin>>num2;

    int sum;
    sum=num1+num2;

    cout<<"Your Sum is :"<<sum;
    return 0;
};
```

LOOP

A loop statement allows us to execute a statement or group of statements multiple times.

For loop::

for(initialize; condition; operation)

{
 // code

false

};

initialize : This statement gets executed only once, at the beginning of the for loop.

Condition: This statement use for break the loop when condition

Operation: It is used for updating a number.(Num++ ← Increase)

Exp :: 04 :: Print 10 natural numbers?

```
#include <iostream>
using namespace std;

int main()
{
    for(int num=1; num<=10; num++)
    cout<<num<<" ";
    return 0;
};
```

num ++ —>> **Post Increment**

++num —>> **Pre increment**

Why use :: ++num, num++, --num, num-- ⇒ **less memory used.**

num++ :- ++num

num = 0	num = 0
sum = num ++	sum = ++num
[store old value then increment]	[store new value]
sum=0	sum = 1
(Store old value)	

Exp :: 05 :: Print all even numbers from 1 to 20?

```
#include <iostream>
using namespace std;

int main()
{
    for(int num=1; num<=20; num++)
    {
        if(num%2==0)
            cout<<num<<" ";
    }
    return 0;
};
```

HOMEWORK

Exp:6 :: Print Prime Number? (10)

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout << "Enter Number: ";
    cin >> n;

    if(n<2)
    {
        cout<<"Not a Prime";
        return 0;
    }

    for (int num = 2; num <= n - 1; num++)
    {
        if (n % num == 0)
        {
            cout << "Not a Prime";
            return 0;
        }
    };

    cout << "Prime Number";
    return 0;
};
```

Exp : 01 :: Print Table of 3?

```
#include <iostream>
using namespace std;

int main()
{
    int num=3;

    for(int i=1; i<=10; i++)
        cout<<num<<" * "<<i<<" = "<<num*i<<endl;

    return 0;
};
```

\n, endl ⇒ use for next line;

Exp : 02 :: Print Fibonacci Series? [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]**#Explanation:**

First_num = 0, **Second_num** = 1, **Current_num** = **First_num** + **Second_num** = 0 + 1 = 1

Fourth_num = New_First_num + New_Second_num	New_First_num = Second_num
= 1 + 1	New_Second_num = Current_num
= 2	

#Code:-

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout << "Enter Number: ";
    cin >> n;

    int first_num = 0;
    int second_num = 1;
    int current_num;

    if (n == 1)
    {
        cout << 0;
        return 0;
    }

    if (n == 2)
```

```
{
    cout << 1;
    return 0;
}

for (int i = 3; i <= n; i++)
{
    current_num = first_num + second_num;
    first_num = second_num;
    second_num = current_num;
}

cout << current_num;

return 0;
};
```

Exp : 03 :: Print Yes / No 3>2>1

#Explanation:

- 3>2 = yes= print 1
- 2>1= yes = print 1
- 1>1= No = print 0 ← output

#Code::

```
#include <iostream>
using namespace std;

int main()
{
    int n = 3 > 2 > 1;
    cout << n;

    return 0;
};
```

C++ Operator Precedence

Precedence	Operator	Associativity
1	a*b a/b a%b	Left-to-right →
2	a+b a-b	Left-to-right →
3	<< >>	“

4	< <= > >=	69
5	== !=	69
6	&&	69
7		69
#	More Information —>	CLICK HERE

Exp : 04 :: Calculate Multiple operations. [n = $3*2 - 10/5$]

#Explanation: $3*2 - 10/5$
 $= 6 - 10/5 = 6 - 2 = 4$ (**Output**)

#Code::

```
#include <iostream>
using namespace std;

int main()
{
    int n = 3*2 - 10/5;
    cout << n;
    return 0;
};
```

Nested Loop:: Nested loop means a loop statement inside another loop statement.

```
include <iostream>
using namespace std;

int main()
{
    int n;

    for (int row = 1; row <= 5; row++) // Outer loop
    {
        for (int col= 1; col <= 5; col++) // Nested Loop
        {
            cout << "1 ";
            cout << endl;
        }

        return 0;
    }
};
```

PATTERN PRINTING

Exp : 05 :: Print Pattern

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

#Explanation:

row =5; col = 5;
Print col;

#Code ::

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    for (int row = 1; row <= 5; row++)
    {
        for (int col = 1; col <= 5; col++)
            cout << col << " ";
        cout << endl;
    }
    return 0;
};
```

Exp : 06 :: Print Pattern

```
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
```

#Explanation:

Print col reverse.

#Code

```
#include <iostream>
using namespace std;

int main()
{
    int n;

    for (int row = 1; row <= 5; row++)
    {
        for (int col = 5; col >= 1; col--)
            cout << col << " ";
    }
}
```

```

        cout << endl;
    }
    return 0;
};

```

Exp : 07 :: Print Pattern

```

1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25

```

#Explanation: Take Counter and increment

#Code::

```

#include <iostream>
using namespace std;

int main()
{
    int n;
    int counter=1;

    for (int row=1; row<=5; row++)
    {
        for (int col=1; col<=5; col++)
        {

            cout<<counter<<" ";
            counter++;
        }
        cout<<endl;
    }
    return 0;
};

```

Exp : 08 :: Print Pattern

```

★
★ ★
★ ★ ★
★ ★ ★ ★

```

#Code::

```

#include <iostream>
using namespace std;

```

```

int main()
{
    int n;
    cout<<"Enter Star Row: ";
    cin>>n;

    for (int row=1; row<=n; row++)
    {
        for (int col=1; col<=row; col++)
        {

            cout<<"*"<<" ";

        }
        cout<<endl;
    }
    return 0;
};

```

Exp : 09 :: Print Pattern

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

#Code::

```

#include <iostream>
using namespace std;

int main()
{
    int n;
    cout<<"Enter Star Row: ";
    cin>>n;

    for (int row=1; row<=n; row++)
    {
        for (int col=1; col<=row; col++)
        {

            cout<<col<<" ";

        }
        cout<<endl;
    }
}

```

```
    return 0;
};
```

Exp : 10 :: Print Pattern

```
a a a a a
b b b b b
c c c c c
d d d d d    { Watch ASCII TABLE- Better Understand}
e e e e e
```

#Code

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout<<"Enter Star Row: ";
    cin>>n;

    for (int row=1; row<=n; row++)
    {
        char c ='a'+row-1;
        for (int col=1; col<=n; col++)
            cout<<c<<" ";

        cout<<endl;
    }
    return 0;
};
```

Exp : 11 :: Print Pattern

```
a
b b
c c c
d d d d
e e e e e
```

#Code

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout<<"Enter Star Row: ";
```

```

cin>>n;

for (int row=1; row<=n; row++)
{
    char c ='a'+row-1;
    for (int col=1; col<=row; col++)
        cout<<c<<" ";

    cout<<endl;
}
return 0;
};

```

HOMework

Exp : 12 :: Print Pattern

```

1 2 3 4 5
 2 3 4 5
  3 4 5
   4 5
    5

```

#Code::

```

#include <iostream>
using namespace std;

int main()
{
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < i; j++)
            cout << " ";

        for (int j = i; j < 5; j++)
        {
            cout << j + 1;
        }
        cout << endl;
    }

    return 0;
};

```

PATTERN PRINTING + WHILE LOOP + MATHS

WEEK :: 01

DAY: 05

DATE: - 21-04-2023

Exp: 01 Pattern Print

```
  *
 **
 ***
 ****
 *****
```

#Explanation:

row	1	print								
		Star	1	= 2 * row -1			★			space = Total row - row nu. = 4 - 1
	2		3	= 2 * row -1			★	★	★	space = 4 - 2
	3		5	= 2 * row -1		★	★	★	★	space = 4 - 3
	4		7	= 2 * row -1	★	★	★	★	★	space = 4 - 4

Code:-

```
#include <iostream>
using namespace std;

int main()
{
    int Total_rows;
    cout << "Enter star number: ";
    cin >> Total_rows;

    for (int row = 1; row <= Total_rows; row++)
    {

        // for space
        for (int col = 1; col <= Total_rows - row; col++)
            cout << " ";

        // For Star
        for (int col = 1; col <= 2 * row - 1; col++)
            cout << "*";

        cout << endl;
```

```

    }

    return 0;
};

```

Exp: 02 Pattern Print

				★				
			★		★			
		★		★		★		
	★		★		★		★	
★		★		★		★		★

#Explanation:

				★					row - 1 (Nrow)	star- 1 (Nrow)	space-4 (Trow-Nrow)
			★		★				2	2	3
		★		★		★			3	3	2
	★		★		★		★		4	4	1
★		★		★		★		★	5	5	0

Nrow = row num; Trow = Total row Inner Space = Cout<<"* space";

Code :-

```

#include <iostream>
using namespace std;

int main()
{
    int Totat_rows;
    cout << "Enter Row Number: ";
    cin >> Totat_rows;

    for (int row = 1; row <= Totat_rows; row++)
    {
        // for space
        for (int col = 1; col <= Totat_rows - row; col++)

```



```
        cout << " ";

        // For Star and Inner Space
        for (int col = 1; col <= row; col++)
            cout << "* ";

        cout << endl;
    }

    return 0;
};
```

Exp: 03 Pattern Print

★									★
★	★							★	★
★	★	★					★	★	★
★	★	★	★			★	★	★	★
★	★	★	★	★	★	★	★	★	★
★	★	★	★			★	★	★	★
★	★	★					★	★	★
★	★							★	★
★									★

#Explanation:

Upper Half Star:

[illegible]

Left side star:

row star

1 1

2 2

3 3

4 4

5 5

(star=row)

For space :

$2 * n -- 2*row$

(Total col - row star)

right side star:

same with left side

(depend on row)

(star=row)

Lower Half Star:

★	★	★	★			★	★	★	★
★	★	★					★	★	★
★	★							★	★
★									★

Left side star:

row star

1 4

2 3

3 2

4 1

(n-row)

For space :

$2 * n - (n-row)$

- (n-row) =

(2 row)

right side star:

same with left side

(n-row)

#Code:

```
#include<iostream>
using namespace std;

int main()
{
    int n;
    //Total_row = n
    cout<<"Enter The row Num: ";
    cin>>n;
    // upper side
    for(int row =1; row<=n; row++)
    {
        // star
        for(int col=1;col<=row;col++)
            cout<<"*";

        // space
        for(int col=1;col<=2*n-2*row;col++)
            cout<<" ";
```

```

        // star
        for(int col=1;col<=row;col++)
            cout<<"*";

        cout<<endl;

    }

    // lower side
    for(int row =1; row<=n-1; row++)
    {
        // star
        for(int col=1;col<= n - row; col++)
            cout<<"*";

        // space
        for(int col=1;col<=2*row; col++)
            cout<<" ";

        // star
        for(int col=1;col<= n - row; col++)
            cout<<"*";

        cout<<endl;

    }

return 0;
};

```

WHILE LOOP

Exp: 01 :: Find digit any number like(125, 220056, 65487);

#Explanation:

If We want to find digit 125. So, We can divide it by 10, 100, 1000, Until result 0.

So, 125 is divided by 1000 and the remainder 125. Thus We know the digit is 3 according to 1000.

Because 000 is equal to a digit.

Zero	Digit
0	1
00	2
00000	5

Code::

```
#include<iostream>
using namespace std;

int main()
{
    int n;
    cin>>n;

    int num=10;
    int digit =1;

    for(int i=1; i>0;i++)    // infinite condition
    {
        if(n/num==0)
        {
            cout<<digit;
            return 0;
        }
        num = num*10;
        digit++;
    }

    return 0;
};
```

#Explanation: **WHILE LOOP**

while(Condition) Condition is True code is executed. When the condition is False, the loop is broken.

```
{
    // #code
};
```

Same Q. Exp: 01 :: Find digit any number like(125, 220056, 65487);

#Explanation: Divided by 10 until we get 0 result.

#Code::

```
#include<iostream>
using namespace std;

int main()
{
    int n;
    cout<<"Enter Your Digit: ";
```

```

cin>>n;
int Digit = 0;

while (n)
{
    Digit++;
    n =n/10;
}
cout<<Digit;

return 0;
};

```

Exp: 02:: Reverse Number: 3 2 6 5 Output: 5 6 2 3

#Explanation:

1st part Get digit reverse one by one.

Divide by 10 and get remainder

$3265/10 \% 5$

$326/10 \% 6$

$32/10 \% 2$

$3/10 \% 3$

2nd part Multiply by 10 and add digits one by one.

$0 * 10 + 5 = 5$

$5 * 10 + 6 = 56$

$56 * 10 + 2 = 562$

$562 * 10 + 3 = 5623$

#Code::

```

#include<iostream>
using namespace std;

int main()
{
    int num;
    cout<<"Enter Number: ";
    cin>>num;
    int sum = 0;

    while(num)
    {
        int digit = num%10;
        sum = sum*10+digit;
        num = num/10;
    }
    cout<<sum;

    return 0;
};

```

Exp: 03:: Convert Decimal to Binary. 37 Output: (100101)

#Explanation:

1st Part-

Divide by 2 and get remainder (%)
One by one

2nd Part-

According to Ex:2
 $\text{bin} \times 10 + \text{bin}$

3rd Part-

According to Ex:2
Reverse

#Code::

```
#include<iostream>
using namespace std;

int main()
{
    int num, sum=0, mul=1;
    cout<<"Enter Your Numbers: ";
    cin>>num;

    while(num>0)
    {
        int digit = num%2;
        sum = sum + digit *mul;
        num= num/2;
        mul = mul*10;
    };
    cout<<sum;

    return 0;
};
```

NOTE**

When input long values like (1345664). It is overflow (32bit to 64bit). So, use a **long long** data type.